ALIBABA CLOUD

Alibaba Cloud

Apsara Stack Enterprise

DataWorks

User Guide

Product Version: v3.16.2

Document Version: 20220916

Alibaba Cloud

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.

2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.

3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.

4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).

5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.

6. Please directly contact Alibaba Cloud for any errors of this document.

Document conventions

Style	Description	Example
Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	Danger: Resetting will result in the loss of user configuration data.
Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	Warning: Restarting will cause business interruption. About 10 minutes are required to restart an instance.
Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	Notice: If the weight is set to 0, the server no longer receives new requests.
Note	A note indicates supplemental instructions, best practices, tips, and other content.	Note: You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click **Settings**> **Network**> **Set network type**.
Bold	Bold formatting is used for buttons , menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

Table of Contents

1.Create a synchronization node

This topic describes how to create a synchronization node to synchronize data from MaxCompute to MySQL.

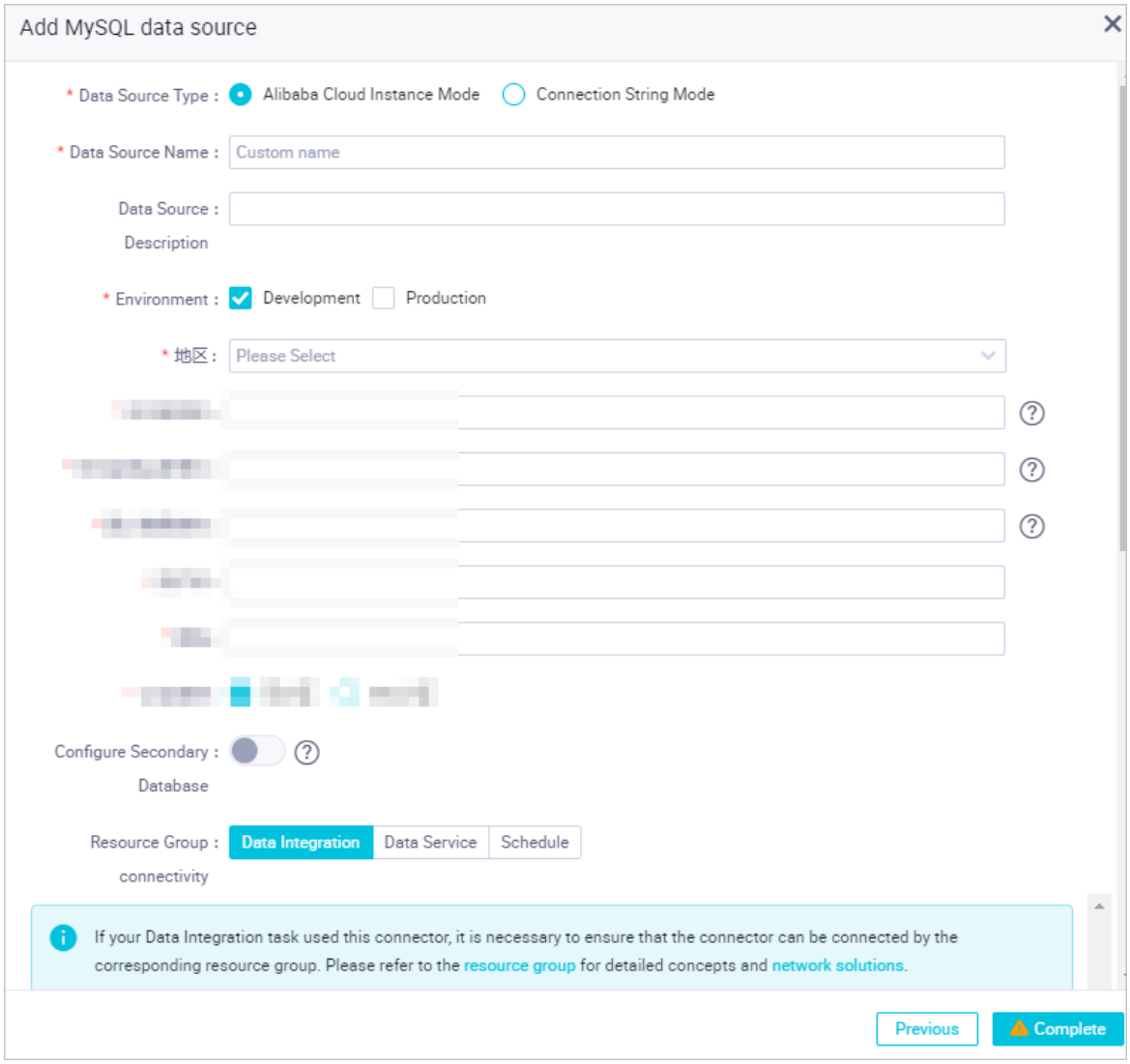
Background information

You can use Data Integration to periodically synchronize the business data generated in a business system to a DataWorks workspace. You can create SQL nodes to compute the data and use Data Integration to periodically synchronize the computing results to your specified data source for further display or use.

Add a data source

> **Note** Only the workspace administrator can add data sources. Members of other roles can only view data sources.

1. Log on to the DataWorks console. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Integration**.
2. In the left-side navigation pane, choose **Data Source** > **Data Sources**. On the page that appears, click **Add data source** in the upper-right corner.
3. In the **Add data source** dialog box, click **MySQL** in the Relational Databases section.
4. In the **Add MySQL data source** dialog box, set the parameters. In this example, a MySQL data source is added by using the **connection string mode**.



Parameter	Description
Data source type	The mode in which the data source is added. Set this parameter to **Connection string mode**.
Data Source Name	The name of the data source. The name can contain letters, digits, and underscores (_) and must start with a letter.
Data source description	The description of the data source. The description can be up to 80 characters in length.
Environment	The environment in which the data source is used. Valid values: **Development** and **Production**. **Note** This parameter is displayed only when the workspace is in standard mode.

Parameter	Description
JDBC URL	The Java Database Connectivity (JDBC) URL of the database. Specify a value for this parameter in the format of `jdbc:mysql://ServerIP:Port/Database`.
User name	The username that is used to connect to the database. **Note** You must enter the information of your MySQL database.
Password	The password that is used to connect to the database.

5. Click **Test connectivity**.
6. If the connectivity test is passed, click **Complete**.

Create a table in the destination MySQL database

The odps_result table is created in the ApsaraDB RDS for MySQL database to which the data is synchronized. You can create the table by executing the following statement:

```
CREATE TABLE `ODPS_RESULT` (
`education`  varchar(255) NULL ,
`num`  int(10) NULL
);
```

After the table is created, execute the `desc odps_result;` statement to view the table details.

Create and configure a batch synchronization node

This section describes how to create and configure a batch synchronization node named write_result and use the node to synchronize data in the result_table table to your MySQL data source. Perform the following steps:

1. On the **Scheduled Workflow** page, create a batch synchronization node named write_result.
2. Configure the insert_data node as the ancestor node of the write_result node.
3. Select **odps_first** of the **ODPS** type as the source and select the result_table table as the source table.
4. Select the odps_result table that you create in the ApsaraDB RDS for MySQL database as the destination table.
5. Configure field mappings between the source and destination tables in the Mappings section.

   Fields in the source on the left have a one-to-one mapping with fields in the destination on the right. You can click **Add** to add a field. You can also click the icon to manually add or remove fields.
6. In the Channel section, set the parameters as required.

Parameter	Description
Expected Maximum Concurrency	The maximum number of parallel threads that the synchronization node can use to read data from the source or write data to the destination. You can configure the parallelism for the synchronization node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable throttling. You can enable throttling and specify a maximum transmission rate to prevent heavy read workloads on the source. We recommend that you enable throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

7. Preview and save the configuration.

   After the node is configured, you can scroll up and down to view the node configuration. After you confirm the settings, click the icon.

Commit the synchronization node

Return to the workflow after you save the synchronization node. Click the icon in the toolbar to commit the synchronization node to the scheduling system. The scheduling system automatically and periodically runs the node from the next day based on the properties configured for the node.

What to do next

You have learned how to create a synchronization node to synchronize data to a specific data source. You can proceed with the next tutorial. In the next tutorial, you will learn how to configure properties and dependencies for a synchronization node. For more information, see Configure recurrence and dependencies for a node.

2.Quick Start

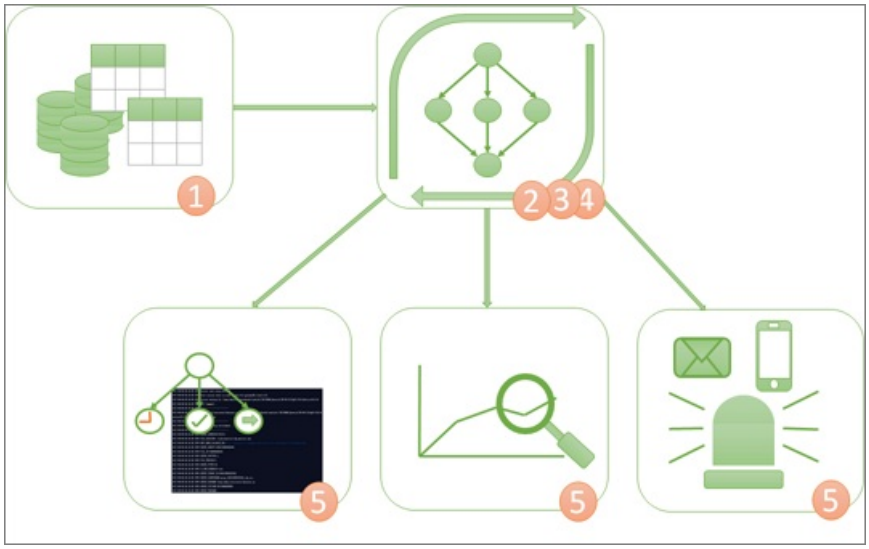
2.1. Overview

Quick Start guides you through a complete process of data analytics and O&M.

Generally, you can complete the following data analytics and O&M operations in a workspace of DataWorks:

1. Create tables and import data.
2. Create a workflow.
3. Create a sync node.
4. Configure recurrence and dependencies for a node.
5. Run a node and troubleshoot errors.

The following figure shows the basic process of data analytics and O&M.



2.2. Create tables and import data

This topic uses the bank_data and result_table sample tables to describe how to create tables and import data in the DataWorks console.

> **Note** The bank_data table stores business data, and the result_table table stores data analytics results.

Create the bank_data table

1. Log on to the DataWorks console.
2. In the left-side navigation pane, click the icon. On the **Workspace Tables** page, click the icon.
3. In the **Create Table** dialog box, set the **Please select an Engine type** parameter to MaxCompute, set the **Table Name** parameter to **bank_data**, and then click **Create**.

4. On the configuration tab of the created table, click **DDL Statement**.
5. In the **DDL Statement** dialog box, enter the table creation statement and click **Generate Table Schema**. In the Confirm dialog box, click **OK**.

   In this example, the following CREATE TABLE statement is used to create a MaxCompute table named bank_data:

```
CREATE TABLE IF NOT EXISTS bank_data
(
 age             BIGINT COMMENT 'Age',
 job             STRING COMMENT 'Job type',
 marital         STRING COMMENT 'Marital status',
 education       STRING COMMENT 'Education level',
 default         STRING COMMENT 'Credit card',
 housing         STRING COMMENT 'Mortgage',
 loan            STRING COMMENT 'Loan',
 contact         STRING COMMENT 'Contact information',
 month           STRING COMMENT 'Month',
 day_of_week     STRING COMMENT 'Day of the week',
 duration        STRING COMMENT 'Duration',
 campaign        BIGINT COMMENT 'Number of contacts during the campaign',
 pdays           DOUBLE COMMENT 'Interval from the last contact',
 previous        DOUBLE COMMENT 'Number of contacts with the customer',
 poutcome        STRING COMMENT 'Result of the previous marketing campaign',
 emp_var_rate    DOUBLE COMMENT 'Employment change rate',
 cons_price_idx  DOUBLE COMMENT 'Consumer price index',
 cons_conf_idx   DOUBLE COMMENT 'Consumer confidence index',
 euribor3m       DOUBLE COMMENT 'Euro deposit rate',
 nr_employed     DOUBLE COMMENT 'Number of employees',
 y               BIGINT COMMENT 'Time deposit available or not'
);
```

6. After the schema is generated, enter the display name of the table and click **Commit to Development Environment** or **Commit to Production Environment**.

   > **Note** If you use a workspace of the basic mode, click **Commit to Production Environment**.

7. In the left-side navigation pane, click **Workspace Tables**. On the Workspace Tables page, enter the table name in the search box to search for the created table. After you find the table, double-click the table name to view the table information.

Create the result_table table

1. In the left-side navigation pane, click the icon. On the **Workspace Tables** page, click the icon.
2. In the **Create Table** dialog box, set the **Please select an Engine type** parameter to MaxCompute, set the **Table Name** parameter to **result_table**, and then click **Create**.
3. On the configuration tab of the created table, click **DDL Statement**.
4. In the **DDL Statement** dialog box, enter the table creation statement and click **Generate Table Schema**. In the Confirm dialog box, click **OK**.

In this example, the following CREATE TABLE statement is used to create a MaxCompute table named result_table:

```
CREATE TABLE IF NOT EXISTS result_table
(
 education   STRING COMMENT 'Education level',
 num         BIGINT COMMENT 'Number of persons'
);
```

5. After the schema is generated, enter the display name of the table and click **Commit to Development Environment** or **Commit to Production Environment**.
6. In the left-side navigation pane, click **Workspace Tables**. On the Workspace Tables page, enter the table name in the search box to search for the created table. After you find the table, double-click the table name to view the table information.

Upload an on-premises file to import its data to the bank_data table

You can perform the following operations in the DataWorks console:

- Upload an on-premises text file to import its data to a table in a workspace.
- Use Data Integration to import business data from different data sources to a workspace.

> **Note** In this topic, an on-premises file is used as the source of data. Comply with the following rules when you upload an on-premises file:
>
> - File format: The file must be in the .txt, .csv, or .log format.
> - File size: The size of the file cannot exceed 10 MB.
> - Destination object: The destination object can be a partitioned table or a non-partitioned table. The partition key value cannot be in Chinese.

To upload the on-premises file banking.txt to DataWorks, perform the following steps:

1. On the **Scheduled Workflow** page, click the **Import** icon.
2. In the **Import local data into the development table** dialog box, select the table to which you want to import data and click **Next**.
3. Set the **Select Data Import Method** parameter to **Upload Local File** and click **Browse**. In the dialog box that appears, select the on-premises file that you want to upload. Set import-related parameters based on your business requirements.

Parameter	Description
Select Data Import Method	The method that you use to import data. Select **Upload Local File**.
Select File	The file from which you want to import data. Click **Browse** and select the on-premises file to upload.
Select Delimiter	The delimiter used in the file. Valid values: **Comma (,)**, **Tab**, **Semicolon (;)**, **Space**, **|**, **#**, and **&**. In this example, **Comma (,)** is selected.

Parameter	Description
Original Character Set	The character set of the file. Valid values: **GBK**, **UTF-8**, **CP936**, and **ISO-8859**. In this example, **GBK** is selected.
Import First Row	The row from which data is to be imported. In this example, **1** is selected.
First Row as Field Names	Specifies whether to use the first line as the header line.
Preview	The preview of the data to be imported. **Note** If the data volume is large, only the data in the first 100 lines and 50 columns appears.

4. Click **Next**.
5. Select a matching mode for the fields in the source file and destination table. In this example, **By Location** is selected.
6. Click **Import Data**.

Data import methods

- Create a synchronization node

  This method is used to import data from various data sources, such as Relational Database Service (RDS), MySQL, SQL Server, PostgreSQL, MaxCompute, Open Cache Service (OCS), Distributed Relational Database Service (DRDS), Object Storage Service (OSS), Oracle, FTP, Dameng (DM), Hadoop Distributed File System (HDFS), and MongoDB.

- Upload an on-premises file

  This method is used to upload .txt and .csv files whose size does not exceed 10 MB. The destination object can be a partitioned table or a non-partitioned table. The partition key value cannot be in Chinese.

- Run Tunnel commands to upload a file

  This method is used to upload on-premises files and other resource files of any size.

What to do next

Now you have learned how to create tables and import data. You can proceed with the next tutorial. In the next tutorial, you will learn how to create a workflow and how to compute and analyze data in a workspace. For more information, see Create a workflow.

2.3. Create a workflow

This topic describes how to create a workflow, create nodes in the workflow, and configure the dependencies among the nodes. After you create a workflow, you can use the DataStudio service to compute and analyze data in the workspace.

Prerequisites

The bank_data table for storing business data and the result_table table for storing data analytics results are created in the workspace. Data is imported to the bank_data table. For more information, see Create tables and import data.

Context

The DataStudio service in DataWorks allows you to configure node dependencies by dragging lines between nodes in a workflow. You can process data and configure node dependencies based on the workflow.

Create a workflow

1. Log on to the DataWorks console.
2. On the **Scheduled Workflow** page, move the pointer over the **Create** icon and click **Workflow**.
3. In the **Create Workflow** dialog box, set the **Workflow Name** and **Description** parameters.
4. Click **Create**.

Create nodes and configure dependencies among the nodes

This section describes how to create a zero load node named start and an ODPS SQL node named insert_data in the workflow, and configure the insert_data node to depend on the start node.

> **Note** Take note of the following items when you use a zero load node:
>
> - A zero load node is a control node that is used to maintain and control its descendant nodes in a workflow. A zero load node does not generate data.
> - If other nodes depend on a zero load node and the zero load node is set to Failed by O&M engineers, the pending descendant nodes cannot run. During the O&M process, a zero load node can be disabled to prevent incorrect data of ancestor nodes from being obtained by their descendant nodes.
> - In most cases, the root node of the workspace is used as the ancestor node of a zero load node in a workflow. The root node of a workspace is named in the format of `Workspace name_root`.

When you design a workflow, we recommend that you create a zero load node as the root node of the workflow to control the entire workflow.

1. On the left side of the Scheduled Workflow page, double-click the name of the workflow that you created below Business Flow. On the configuration tab that appears, choose General > **Zero-Load Node**. You can also drag **Zero-Load Node** to the canvas on the right side to go to the Create Node dialog box.
2. In the **Create Node** dialog box, set the **Name** parameter to start and click **Commit**.
3. Repeat the preceding steps to create an **ODPS SQL** node named insert_data.
4. Draw a line to connect the nodes and set the start node as the ancestor node of the insert_data node.

Configure the ancestor node of the zero load node

In a workflow, a zero load node is often used to control the entire workflow and serves as the ancestor node of all nodes in the workflow. In most cases, the zero load node in a workflow depends on the root node of the workspace.

1. Double-click the name of the zero load node. On the page that appears, click the **Properties** tab in the right-side navigation pane.
2. In the **Dependencies** section, click **Add Root Node** and set the root node of the workspace as the ancestor node of the zero load node.
3. After the configuration is complete, click the icon in the upper-left corner.

Edit code in the ODPS SQL node

This section provides a sample SQL statement used to query and save the number of singles with different education levels who loan to buy houses in the ODPS SQL node insert_data. The queried data can be analyzed by and presented in descendant nodes of insert_data.

The following content shows the SQL statement:

```
INSERT OVERWRITE TABLE result_table  -- Insert data into the result_table table.
SELECT education
    , COUNT(marital) AS num
FROM bank_data
WHERE housing = 'yes'
    AND marital = 'single'
GROUP BY education
```

Run and debug the ODPS SQL node

1. After the SQL statement is entered in the insert_data node, click the **Save** icon.
2. Click the **Run** icon to view the run logs and results.

Commit the workflow

1. After you run and debug the ODPS SQL node insert_data, return to the workflow editing page and click the **Submit** icon.
2. In the **Commit** dialog box, select the node to be committed, select a reviewer from the **Specify code reviewer(Required)** drop-down list, enter a description in the **Change Description** text box, and then select **Ignore I/O Inconsistency Alerts**.
3. Click **Commit**.

> **Notice** **Specify code reviewer(Required)** is optional. You cannot commit a special node for code review. For example, you cannot commit a combined node such as a resource node, do-while node, or for-each node for code review. If you specify a code reviewer, the system generates a code review ticket. If the forcible code review feature is disabled for the workspace, you do not need to commit your node for code review. If the forcible code review feature is enabled, you must commit your node for code review. You can deploy the node only after the specified reviewer approves the node code.

What to do next

Now you have learned how to create and commit a workflow. You can proceed with the next tutorial. In the next tutorial, you will learn how to create a synchronization node to export data to different types of data sources. For more information, see Create a sync node.

2.4. Create a synchronization node

This topic describes how to create a synchronization node to synchronize data from MaxCompute to MySQL.

Background information

You can use Data Integration to periodically synchronize the business data generated in a business system to a DataWorks workspace. You can create SQL nodes to compute the data and use Data Integration to periodically synchronize the computing results to your specified data source for further display or use.

Add a data source

> **Note** Only the workspace administrator can add data sources. Members of other roles can only view data sources.

1. Log on to the DataWorks console. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Integration**.
2. In the left-side navigation pane, choose **Data Source > Data Sources**. On the page that appears, click **Add data source** in the upper-right corner.
3. In the **Add data source** dialog box, click **MySQL** in the Relational Databases section.
4. In the **Add MySQL data source** dialog box, set the parameters. In this example, a MySQL data source is added by using the **connection string mode**.

Parameter	Description
Data source type	The mode in which the data source is added. Set this parameter to **Connection string mode**.
Data Source Name	The name of the data source. The name can contain letters, digits, and underscores (_) and must start with a letter.
Data source description	The description of the data source. The description can be up to 80 characters in length.
Environment	The environment in which the data source is used. Valid values: **Development** and **Production**. > **Note** This parameter is displayed only when the workspace is in standard mode.
JDBC URL	The Java Database Connectivity (JDBC) URL of the database. Specify a value for this parameter in the format of `jdbc:mysql://ServerIP:Port/Database`.

Parameter	Description
User name	The username that is used to connect to the database. **Note** You must enter the information of your MySQL database.
Password	The password that is used to connect to the database.

5. Click **Test connectivity**.
6. If the connectivity test is passed, click **Complete**.

Create a table in the destination MySQL database

The odps_result table is created in the ApsaraDB RDS for MySQL database to which the data is synchronized. You can create the table by executing the following statement:

```
CREATE TABLE `ODPS_RESULT` (
`education`  varchar(255) NULL ,
`num`  int(10) NULL
);
```

After the table is created, execute the `desc odps_result;` statement to view the table details.

Create and configure a batch synchronization node

This section describes how to create and configure a batch synchronization node named write_result and use the node to synchronize data in the result_table table to your MySQL data source. Perform the following steps:

1. On the **Scheduled Workflow** page, create a batch synchronization node named write_result.
2. Configure the insert_data node as the ancestor node of the write_result node.
3. Select **odps_first** of the **ODPS** type as the source and select the result_table table as the source table.
4. Select the odps_result table that you create in the ApsaraDB RDS for MySQL database as the destination table.
5. Configure field mappings between the source and destination tables in the Mappings section.

   Fields in the source on the left have a one-to-one mapping with fields in the destination on the right. You can click **Add** to add a field. You can also click the icon to manually add or remove fields.
6. In the Channel section, set the parameters as required.

Parameter	Description
Expected Maximum Concurrency	The maximum number of parallel threads that the synchronization node can use to read data from the source or write data to the destination. You can configure the parallelism for the synchronization node on the codeless UI.

Parameter	Description
Bandwidth Throttling	Specifies whether to enable throttling. You can enable throttling and specify a maximum transmission rate to prevent heavy read workloads on the source. We recommend that you enable throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

7. Preview and save the configuration.

   After the node is configured, you can scroll up and down to view the node configuration. After you confirm the settings, click the icon.

Commit the synchronization node

Return to the workflow after you save the synchronization node. Click the icon in the toolbar to commit the synchronization node to the scheduling system. The scheduling system automatically and periodically runs the node from the next day based on the properties configured for the node.

What to do next

You have learned how to create a synchronization node to synchronize data to a specific data source. You can proceed with the next tutorial. In the next tutorial, you will learn how to configure properties and dependencies for a synchronization node. For more information, see Configure recurrence and dependencies for a node.

2.5. Configure recurrence and dependencies for a node

This topic describes how to configure recurrence and dependencies for a node in the DataWorks console.

> **Note** In this topic, the batch synchronization node write_result described in the "Create a synchronization node" topic is used, and the recurrence is set to weekly.

DataWorks has a powerful scheduling engine to trigger nodes based on the recurrence and dependencies of the nodes. DataWorks guarantees that tens of millions of nodes run accurately and punctually per day based on directed acyclic graphs (DAGs). In the DataWorks console, you can set the recurrence to minutely, hourly, daily, weekly, or monthly.

Configure recurrence for the batch synchronization node

1. After the synchronization node write_result is created, double-click the node to configure it.
2. On the configuration tab of the write_result node, click the **Properties** tab in the right-side navigation pane.
3. In the **Schedule** section, set the parameters as required.

Parameter	Description
Recurrence	The mode in which the node is run. Valid values: **Normal**, **Skip Execution**, and **Dry Run**.
Rerun	Specifies whether to allow the node to be rerun. Valid values: **Allow Regardless of Running Status**, **Allow upon Failure Only**, and **Disallow Regardless of Running Status**.
Auto Rerun upon Error	Specifies whether to rerun the node upon an error.
Validity Period	The date from which the node is effective.
Scheduling Cycle	The recurrence of the node. Valid values: Minute, Hour, Day, Week, and Month. In this example, Week is selected.
Run At	The specific day and time when the node is run. For example, you can configure the node to run at 02:00 every Tuesday.
Cron Expression	The CRON expression of the time that you specified. The expression cannot be changed.
Timeout Definition	The timeout period for the node. You can select **Default**, or select **Specify Timeout Period** and specify a custom timeout period.
Dependency on Last Cycle	Specifies whether the node depends on the result of the last cycle.

Configure dependencies for the batch synchronization node

After you configure recurrence for the batch synchronization node write_result, you can configure dependencies for the node.

You can configure the parent node on which the synchronization node depends. After that, the scheduling system triggers the synchronization node only after the instance of the parent node is run.

For example, the instance of the synchronization node is not triggered until the instance of its parent node insert_data is run.

By default, the scheduling system creates a node named in the format of Workspace name_root for each workspace as the root node. If no parent node is configured for the synchronization node, the synchronization node depends on the root node.

Save and commit the node.

1. Save the node: On the configuration tab of the write_result node, click the  icon in the toolbar.
2. Commit the node: Click the  icon in the toolbar. In the **Commit Node** dialog box, enter your comments in the **Change description** field and click **OK**.

> **Notice** You can commit the node only after you set the **Rerun** and **Parent Nodes** parameters.

A node must be committed to the scheduling system so that the scheduling system can automatically generate and run instances for the node. The scheduling system runs these instances at the specified time from the next day based on the recurrence settings.

> **Note** If you commit a node after 23:30, the scheduling system automatically generates and runs instances for the node from the third day.

What to do next

Now you have learned how to configure recurrence and dependencies for a synchronization node. You can proceed with the next tutorial. In the next tutorial, you will learn how to perform O&M operations on the committed node and troubleshoot errors based on the operational logs. For more information, see Run a node and troubleshoot errors.

2.6. Run a node and troubleshoot errors

This topic describes how to run and manage a node, and troubleshoot errors based on logs.

In the preceding step of Configure scheduling properties and dependencies for a node, you have configured the synchronization node write_result to run at 02:00 every Tuesday. After you commit this node, you have to wait until the next day to view the automatic scheduling result of this node. DataWorks allows you to run nodes in the following modes: test run, data backfill, and periodic run. This helps you confirm the scheduled time of each node instance, dependencies among node instances, and whether generated data meets your expectation.

- Test run: Nodes are manually triggered. We recommend that you use this mode if you need to check the scheduled time and running of a single node.
- Data backfill: Nodes are manually triggered. We recommend that you use this mode if you need to check the scheduled time of multiple nodes and dependencies among them, or if you need to reperform data analysis and computing from a specific root node.
- Periodic run: Nodes are automatically triggered. After you commit a node, the scheduling system automatically generates and runs instances for the node from 00:00 the next day. When the scheduled time of each instance arrives, the scheduling system checks whether the ancestor instances of the instance have been run. If all the ancestor instances have been run, the scheduling system automatically triggers the instance without manual intervention.

> **Note** The scheduling system generates instances for manually triggered nodes and auto triggered nodes based on the same rules.
>
> - The scheduling system generates an instance for each recurrence, which can occur by the day, hour, minute, month, or week.
> - The scheduling system runs the instances generated for the specified dates only when the scheduled time arrives and generates run logs for the instances.
> - The scheduling system does not run the instances generated for other dates. Instead, it directly changes the states of the instances to successful when the running conditions are met. In this case, the scheduling system does not generate run logs.

Test run

1. Log on to the DataWorks console.
2. On the DataStudio page, click the More icon in the upper-left corner and choose **All Products > Operation Center** to go to the **Operation Center** page.
3. In the left-side navigation pane, choose **Cycle Task MaintenanceCycle Task**. On the page that appears, find the node that you want to run and click **Test** in the Actions column.
4. In the **Test** dialog box, set the **Test Name** and **Data Timestamp** parameters, and click **OK**.
5. On the **Test Instance** page, click the name of the generated instance. The directed acyclic graph (DAG) of the instance appears on the right.

   Right-click the instance in the DAG to view its dependencies and details, or manage this instance. For example, you can rerun this instance.

> Note
>
> - In test run mode, a node is manually triggered. When the scheduled time arrives, the scheduling system runs the corresponding instance immediately, no matter whether the ancestor instances have been run.
> - The synchronization node write_result is configured to run at 02:00 every Tuesday. Based on the instance generation rules described earlier in this topic, if the data timestamp, which is one day before the run date, is set to Monday for a test run, the scheduling system runs the instance generated for the synchronization node write_result at 02:00 on Tuesday. If the data timestamp is not set to Monday for the test run, the scheduling system changes the state of the instance to successful at 02:00 on Tuesday with no run logs generated.

Data backfill

We recommend that you backfill data for a node if you need to check the scheduled time of multiple nodes and dependencies among them, or if you need to reperform data analysis and computing from a specific root node.

1. On the **Operation Center** page, choose **Cycle Task Maintenance > Cycle Task** in the left-side navigation pane.
2. On the page that appears, find the node that you want to run and choose **Patch Data > Current Node Retroactively** in the Actions column.
3. In the **Patch Data** dialog box, set the parameters that are described in the following table and click **OK**.

Parameter	Description
Retroactive Instance Name	The name of the data backfill instance.
Data Timestamp	The data timestamp of the data backfill instance. The data backfill instance is run on the next day of the specified data timestamp.
Node	The node for which you want to backfill data. By default, the current node is selected, which cannot be changed.

Parameter	Description
Parallelism	Specifies whether to concurrently run the node with other nodes. Clear the check box, or select the check box and specify several nodes to run concurrently.
Order	The order for nodes to run. Valid values: **Ascending by Business Date** and **Descending by Business Date**.

4. On the **Patch Data** page, click the name of the generated data backfill instance to view the DAG of the instance.

   Right-click the instance in the DAG to view its dependencies and details, or manage this instance. For example, you can rerun this instance.

   > Note
   > - In data backfill mode, the running of an instance depends on the instance generated for the previous day. For example, in the scenario in which you configure data backfill instances to run from September 15, 2017 to September 18, 2017, if the instance on September 15 fails to run, the instance on September 16 cannot be run.
   > - The synchronization node write_result is configured to run at 02:00 every Tuesday. Based on the instance generation rules described earlier in this topic, if the data timestamp, which is one day before the run date, is set to Monday for data backfill, the scheduling system runs the instance generated for the synchronization node write_result at 02:00 on Tuesday. If the data timestamp is not set to Monday for the data backfill, the scheduling system changes the state of the instance to successful at 02:00 on Tuesday with no run logs generated.

Periodic run

In periodic run mode, the scheduling system automatically triggers instances for all nodes based on the scheduling configuration. No menu item is provided for you to control the periodic run on the Operation Center page. You can view the instance information and run logs of a node, such as the synchronization node write_result, by using one of the following methods:

- On the **Operation Center** page, choose **Cycle Task Maintenance > Cycle Instance** in the left-side navigation pane. On the page that appears, set parameters such as the data timestamp or run date to search for a specific instance of the node. Then, right-click the instance in the DAG to view the instance information and run logs.
- On the **Cycle Instance** page, click an instance of the node. The DAG of the instance appears.

  Right-click the instance in the DAG to view its dependencies and details, or manage this instance. For example, you can rerun this instance.

Note

- If an ancestor node has not been run, its descendant nodes are not run either.
- If the initial state of an instance is pending, the scheduling system checks whether all its ancestor instances have been run when the scheduled time arrives.
- The instance can be triggered and run only after all its ancestor instances have been run and when the scheduled time arrives.
- If an instance is pending, check whether all its ancestor instances have been run and whether the scheduled time arrives.

3.Data Integration

3.1. Overview

Data Integration is a stable, efficient, and scalable data synchronization service. It is designed to migrate and synchronize data between a wide range of heterogeneous data stores fast and stably in complex network environments.

Limits

- Data Integration can synchronize structured, semi-structured, and unstructured data. Structured data stores include Relational Database Service (RDS) and Distributed Relational Database Service (DRDS). Unstructured data, such as Object Storage Service (OSS) objects and text files, must be capable of being converted to structured data. Data Integration can only synchronize data that can be abstracted to two-dimensional logical tables to MaxCompute. It cannot synchronize unstructured data that cannot be converted to structured data, such as MP3 files stored in OSS, to MaxCompute.
- Data Integration supports data synchronization and exchange in one region or between regions.

  Data can be transmitted between regions over the classic network, but the network connectivity is not guaranteed. If the transmission fails over the classic network, we recommend that you use an Internet connection.
- Data Integration supports only data synchronization but not data consumption.

Batch data synchronization

Data Integration can be used to synchronize large amounts of data. Data Integration facilitates data transmission between diverse structured and semi-structured data stores. It provides readers and writers for the supported data stores and defines a transmission channel between the source and destination data stores and datasets, based on simplified data types.

Supported data stores

- Relational databases: MySQL, SQL Server, PostgreSQL, Oracle, Dameng, DRDS, PolarDB, HybridDB for MySQL, AnalyticDB for PostgreSQL, AnalyticDB for MySQL 2.0, and AnalyticDB for MySQL 3.0
- Big data storage: MaxCompute, DataHub, and Data Lake Analytics (DLA)
- Semi-structured storage: OSS, Hadoop Distributed File System (HDFS), and FTP
- NoSQL: MongoDB, Memcache, Redis, and Tablestore
- Message queue: LogHub
- Graph compute engine: GraphCompute

For more information, see Supported data sources.

> **Note** The connection configurations for data stores vary greatly. You can view the specific parameters that need to be set when you configure connections and sync nodes for data stores.

Development modes of sync nodes

You can develop sync nodes in one of the following modes:

- **Codeless UI**: Data Integration provides step-by-step instructions to help you configure a sync node. This mode is easy to use but provides only limited features.

- **Code editor**: You can write a JSON script to create a sync node. This mode supports advanced features to facilitate flexible configuration. It is suitable for experienced users and increases the cost of learning.

> **Note**
> - The code generated for a sync node on the codeless user interface (UI) can be converted to a script. This conversion is irreversible.
> - Before you write code, you must configure a connection and create the destination table.

Network types

A data store can reside on the classic network or in a virtual private cloud (VPC). The user-created IDC network type has been planned and will be supported soon.

- Classic network: a network deployed by Alibaba Cloud, which is shared with other tenants. This network is easy to use.
- VPC: a network created on Alibaba Cloud, which is available to only one Apsara Stack account. You have full control over your VPC, including customizing the IP address range, dividing the VPC to multiple subnets, and configuring routing tables and gateways.

  A VPC is an isolated network for which you can customize a wide range of parameters, such as the IP address range, subnets, and gateways. Based on wide deployment of VPCs, Data Integration provides the feature to automatically detect the reverse proxy for some data stores, including ApsaraDB RDS for MySQL, ApsaraDB RDS for PostgreSQL, ApsaraDB RDS for SQL Server, PolarDB, DRDS, HybridDB for MySQL, AnalyticDB for PostgreSQL, and AnalyticDB for MySQL 3.0. By using this feature, you do not need to purchase an extra Elastic Compute Service (ECS) instance in your VPC to configure sync nodes for these data stores. Instead, Data Integration automatically uses this feature to provide network connectivity to these data stores.

  When you configure sync nodes for other Alibaba Cloud data stores in a VPC, such as PPAS, ApsaraDB for OceanBase, ApsaraDB for Redis, ApsaraDB for MongoDB, ApsaraDB for Memcache, Tablestore, and ApsaraDB for HBase, you must purchase an ECS instance in the same VPC. This ECS instance is used to access the data stores.
- User-created IDC network: an IDC network deployed by yourself, which is isolated from the Alibaba Cloud network.

> **Note** You can access data stores over the Internet. However, the access speed depends on the Internet bandwidth, and additional network access expenses are required. We recommend that you do not use Internet connections.

Terms

- Concurrency

  Concurrency indicates the maximum number of concurrent threads to read data from or write data to data storage within a single sync node.
- Bandwidth throttling

  Bandwidth throttling indicates that a maximum transmission rate is specified for a sync node of Data Integration.
- Dirty data

Dirty data indicates meaningless data and data that does not match the specified data type. For example, you want to write data of the VARCHAR type in the source table to an INT-type field in the destination table. A data conversion error occurs and the data cannot be written to the destination table. In this case, the data is dirty.

- Connection

  A connection in DataWorks is used for accessing a data store, which can be a database or a data warehouse. DataWorks supports various types of data stores, and supports data synchronization between data stores of different types.

3.2. Homepage

The Data Integration homepage provides entries for you to create sync nodes, manage connections, maintain sync nodes, and view help documents.

Log on to the DataWorks console, click ☰ in the upper-left corner, and choose **All Products > Data Aggregation > Data Integration**. The homepage of Data Integration appears by default.

On this page, you can perform the following operations:

- **New Task**: Click here to go to the **Data Analytics** page, where you can create sync nodes. For more information, see Create a sync node.
- **Connection**: Click here to go to the **Data Source** page, where you can view created connections and add a connection or multiple connections at a time.
- **Workbench**: Click here to go to the **Operation Center > Dashboard** page, where you can view the running status of created nodes. For more information, see *View the statistics on the Overview page*.

3.3. Connectivity testing

This topic describes the FAQ about connectivity testing on connections.

When configuring a security group for a connection hosted on an Elastic Compute Service (ECS) instance, add the IP address of the scheduling cluster to the inbound and outbound rules of the security group. If the security group is not properly configured, data synchronization fails due to a connection failure.

To set a wide port range for a security group rule, call relevant API operations, instead of using the console.

Common scenarios of connectivity test failures

When a connection fails the connectivity test, check whether the region, network type, whitelist, database name, and username are properly configured for the connection. The following errors may occur during connectivity testing:

- The database password is incorrect.
- The network connection fails.
- A network error occurs during data synchronization.

  Check the log and determine which resource group is used. Check whether the resource group is a custom one.

  For a Relational Database Service (RDS) connection or a MongoDB connection, if a custom resource group is used, check whether its IP addresses are added to the whitelist of the connection.

Check whether both the source and destination connections pass the connectivity test. For an RDS connection or a MongoDB connection, check whether all relevant IP addresses are added to the whitelist of the connection. If the IP address of a server is not added to the whitelist, the sync node fails when it runs on this server. However, the sync node succeeds when it runs on another server whose IP address is added to the whitelist.

- The result shows that a sync node is run but the log contains a disconnection error in port 8000.

  This issue occurs because a custom resource group is used and no inbound rule is configured for the corresponding IP address and port 8000 in the security group. To resolve the issue, add the IP address and port to the inbound rule of the security group and run the node again.

Examples of connectivity test failures

Example 1

- Symptom

  A connection failed the connectivity test. The database connection failed. The following information is involved: Database URL: jdbc:mysql://xx.xx.xx.x:xxxx/t_uoer_bradef. Username: xxxx_test. Error message: Access denied for user 'xxxx_test'@'%' to database 'yyyy_demo'.

- Troubleshooting

  i. Check whether the configuration of the connection is correct.

  ii. Check whether the database password is correct, the whitelist is properly configured, and your account has the permission to access the database. You can grant the required permissions in the RDS console.

- Example 2
  - Symptom

    A connection failed the connectivity test. The following error message is returned:

    ```
    error message: Timed out after 5000 ms while waiting for a server that matches ReadPref
    erenceServerSelector{readPreference=primary}. Client view of cluster state is {type=UNK
    NOWN, servers=[(xxxxxxxxxx), type=UNKNOWN, state=CONNECTING, exception={com.mongodb.Mon
    goSocketReadException: Prematurely reached end of stream}}]
    ```

  - Troubleshooting

    Before testing the connectivity to a MongoDB connection that is not deployed in a Virtual Private Cloud (VPC), add relevant IP addresses to the whitelist of the connection.

3.4. Authentication file management

3.4.1. Upload and reference an authentication file

The Data Integration service of DataWorks supports third-party identity authentication mechanisms. Before you use an authentication mechanism to perform identity authentication, you must upload the required authentication files on the Authentication File Management page of the DataWorks console. Then, you must enable third-party identity authentication when you add a data source. This way, only trusted applications and services can access the data source. This topic describes how to upload and reference an authentication file.

Context

Third-party identity authentication mechanisms are used to perform strict identity authentication on users and services. These mechanisms prevent untrusted applications or services from accessing data and improve the stability of data access during data synchronization. The Authentication File Management page of the DataWorks console allows you to manage authentication files in a centralized manner. You can upload an authentication file and view the data sources that reference an authentication file on this page.

Limits

Only Kerberos authentication is supported. Other authentication mechanisms will be available in the future. For more information about Kerberos authentication, see Configure Kerberos authentication.

Upload an authentication file

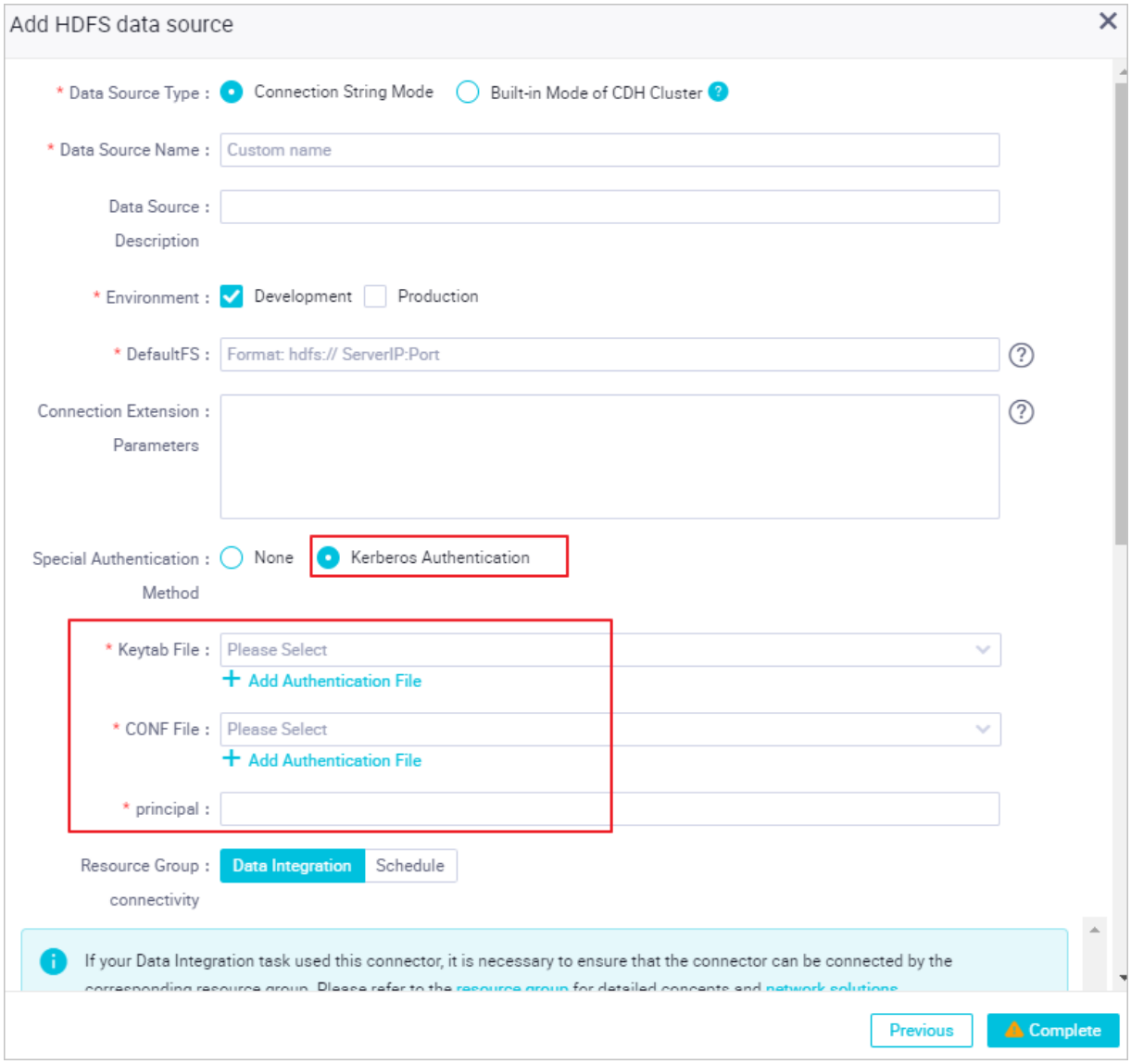
Before you use an identity authentication mechanism, you must upload the required authentication files on the Authentication File Management page.

1. Log on to the DataWorks console.
2. On the DataStudio page, click the ☰ icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
3. In the left-side navigation pane of the Data Integration page, choose **Data Source > Authentication File Management**.
4. On the **Authentication File Management** page, click **Upload Authentication File** in the upper-right corner.
5. In the **Upload Authentication File** dialog box, click **Upload File**, select a file, and then enter a description in the **File Description** field.
6. Click **OK**.

Reference an authentication file

If you want to use third-party identity authentication, you must enable special identity authentication, set relevant parameters, and then reference the uploaded authentication files when you add a data source. DataWorks supports only Kerberos authentication. For more information, see Configure Kerberos authentication.

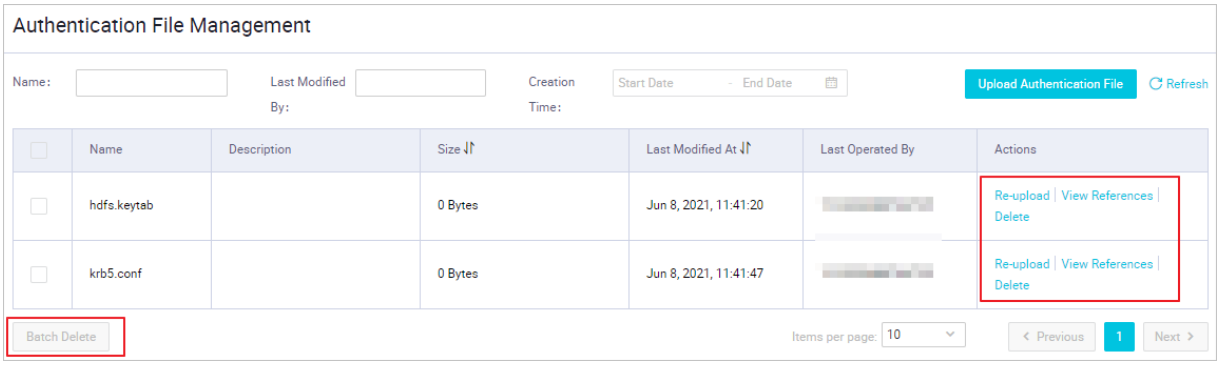
The following table describes the parameters that you must set after you set the Special Authentication Method parameter to Kerberos Authentication when you add an HDFS data source. For more information about how to add a data source, see Supported data stores and plug-ins.



Parameter	Description
Special Authentication Method	Set this parameter to **Kerberos Authentication**.
Keytab File	Select an uploaded keytab file from the Keytab File drop-down list. If you want to upload an authentication file, click **Add Authentication File**.
CONF File	Select a CONF file from the CONF File drop-down list. If you want to upload an authentication file, click **Add Authentication File**.
principal	The Kerberos principal that consists of the principal name, instance name, and domain name. Specify this parameter in the format of `Principal name/Instance name@Domain name`, such as `****/hadoopclient@**.***`.

Other operations

You can upload an authentication file and view the data sources that reference an uploaded authentication file on the **Authentication File Management** page. You can also remove multiple authentication files at a time on this page.



3.4.2. Configure Kerberos authentication

The Data Integration service of DataWorks supports only Kerberos authentication. After you configure Kerberos authentication, identity authentication is performed only on trusted applications and services. This way, only the applications and services that pass the authentication can access data. This topic describes how Kerberos authentication works.

Context

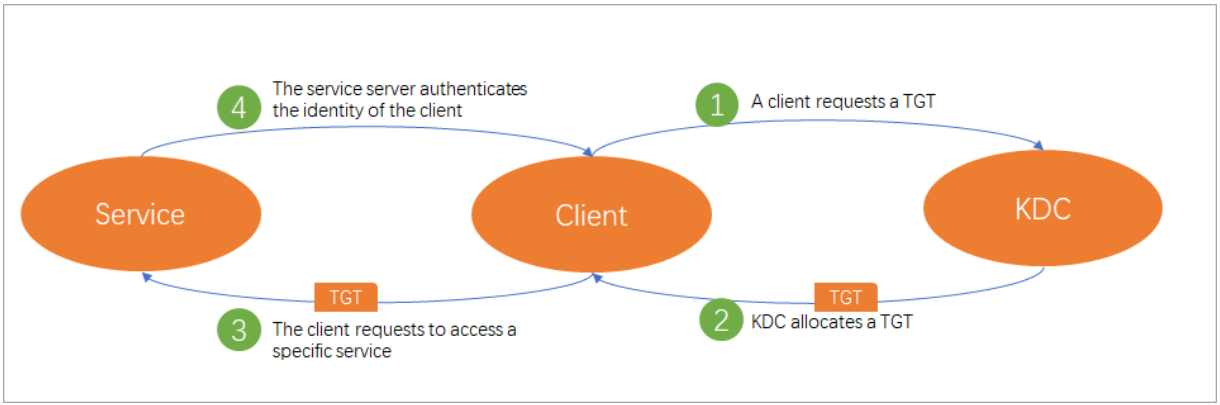
Kerberos is a computer network security protocol for authentication. It enables users to obtain service tickets that can be used to access multiple services by providing identity authentication information only once to achieve single sign-on (SSO). Kerberos provides high security. When you use Kerberos, a shared key is created between each client and service. Clients communicate with services by using keys. This way, untrusted services or applications cannot access data.

Limits

- Only CDH V6.X clusters support Kerberos authentication. CDH clusters of other versions or self-managed clusters for which Kerberos authentication tests are not performed may fail the authentication.
- The Kerberos authentication feature supports only HBase, HDFS, and Hive data sources.

How Kerberos authentication works

Kerberos is a third-party authentication protocol that is based on symmetric keys. Clients and services use Key Distribution Center (KDC) to perform identity authentication. KDC is a server program of Kerberos and can distribute Ticket Granting Tickets (TGT).



The preceding figure shows the four stages that are contained during the Kerberos authentication on DataWorks.

1. A client requests a TGT: When a client (principal) accesses a data source for which Kerberos authentication is enabled, the client requests a TGT from Authentication Server (AS) in KDC. Then, the client uses the obtained TGT to request another TGT for a specific service from Ticket Granting Server (TGS) in KDC.
2. KDC grants a TGT: After KDC receives a request from the client, KDC authenticates the identity of the client. If the client passes the authentication, KDC grants an encrypted TGT that has a specific validity period to the client.
3. The client requests to access a specific service: After the client obtains the TGT, the client requests to access specific service resources from the service server based on the service name.
4. The service server authenticates the identity of the client: After the service server receives the request from the client, the server authenticates the identity of the client. If the client passes the authentication, the client can access the service resources.

A keytab authentication file and a krb5.conf configuration file are required for Kerberos authentication. The krb5.conf configuration file is used to store the configurations of KDC servers. The keytab file is used to store the identity authentication tickets of resource principals, including principals and encrypted principal keys. Before you perform Kerberos authentication, you must upload the keytab authentication file and keb5.conf configuration file on the Authentication File Management page of the DataWorks console, reference the uploaded files, and then configure a principal when you add a data source. For more information about how to upload a keytab authentication file and the parameters of Kerberos authentication for different types of data sources, see Upload and reference an authentication file and Data sources that support Kerberos authentication.

Data sources that support Kerberos authentication

The following table lists the data source types that support Kerberos authentication and the configuration guide of Kerberos authentication for these types of data sources.

Data source type	References
HBase	Add an HBase data source

Data source type	References
Hadoop Distributed File System (HDFS)	Add an HDFS data source
Hive	Add a Hive data source
Kudu	Add a Kudu data source

3.5. Data sources

3.5.1. Supported data sources

Data Integration is a stable, efficient, and scalable data synchronization service. It provides transmission channels for offline and batch data in big data computing services of Alibaba Cloud such as MaxCompute, AnalyticDB for PostgreSQL, and Hologres.

The following table describes the data source types and plug-ins that are supported by Data Integration.

Data source type	Reader	Writer
ApsaraDB for OceanBase	ApsaraDB for OceanBase Reader	ApsaraDB for OceanBase Writer
DataHub	DataHub Reader	DataHub Writer
Db2	Db2 Reader	Db2 Writer
Dameng (DM)	RDBMS Reader	RDBMS Writer
DRDS	DRDS Reader	DRDS Writer
Elasticsearch	Elasticsearch Reader	Elasticsearch Writer
FTP	FTP Reader	FTP Writer
GBase 8a	Supported	GBase 8a Writer
HBase	HBase Reader	• HBase Writer • HBase11xsql Writer
Hadoop Distributed File System (HDFS)	HDFS Reader	HDFS Writer
Hive	Hive Reader	Hive Writer
Hologres	Supported	Supported
HybridDB for MySQL	Supported	Supported
LogHub	LogHub Reader	LogHub Writer

Data source type	Reader	Writer
MaxCompute	MaxCompute Reader	MaxCompute Writer
Memcache	Not supported	Memcache Writer
MongoDB	MongoDB Reader	MongoDB Writer
MySQL	MySQL Reader	MySQL Writer
Oracle	Oracle Reader	Oracle Writer
Object Storage Service (OSS)	OSS Reader	OSS Writer
PolarDB	Supported	Supported
PostgreSQL	PostgreSQL Reader	PostgreSQL Writer
Relational database management system (RDBMS)	RDBMS Reader	RDBMS Writer
Redis	Not supported	Redis Writer
Stream	Stream Reader	Stream Writer
SQL Server	SQL Server Reader	SQL Server Writer
Tablestore	Tablestore Reader	Tablestore Writer
Vertica	Vertica Reader	Vertica Writer

3.5.2. Data source isolation

DataWorks provides the data source isolation feature for workspaces in standard mode. This way, data sources in the development environment can be isolated from data sources in the production environment.

If a data source is configured in both the development and production environments, you can use the data source isolation feature to isolate the data source in the development environment from that in the production environment.

> **Note** Only workspaces in standard mode support the data source isolation feature.

When you configure a synchronization node, the data source in the development environment is used. After you commit the synchronization node to the production environment for running, the data source in the production environment is used. To commit a synchronization node to the production environment for running, you must configure data sources in both the development and production environments. The data sources must have the same name in the development and production environments.

The data source isolation feature has the following impacts on workspaces:

- Workspaces in basic mode: The features and configuration dialog boxes of data sources are the

same as those before the data source isolation feature is added.

- Workspaces in standard mode: The Environment parameter is added to the configuration dialog boxes of data sources.
- Workspaces upgraded from the basic mode to the standard mode: When you upgrade the mode, you are prompted to upgrade data sources. After you upgrade the mode, the data sources in the development environment are isolated from those in the production environment.

3.5.3. Sync data monitoring

The Sync Data Monitoring page displays the total number of sync node instances for different connections and the instance details based on the selected workspace and time range.

The cut-off time of data to be displayed is 0 minutes 0 seconds of the current hour. For example, if the current time is 2019-04-04 10:10:00, the page displays the data generated before 2019-04-04 10:00:00.

1. Log on to the DataWorks console and select a workspace.
2. Click the DataWorks icon in the upper-left corner and choose **All Products > Data Integration** to go to the **Data Integration** page.
3. In the left-side navigation pane, click **Sync Data Monitoring**. On the page that appears, view the total number of sync node instances for different connections and the instance details.
   - View summary data by connection type

     The Source and Target sections display the summary data of source connections and that of destination connections, respectively. Take source connections as an example. If the Source section displays MaxCompute with the value 1, a sync node instance whose source connection is MaxCompute is run in the selected time range.
   - View instance details

     The Sync Instances section displays the details of all sync node instances that are run in the selected time range. You can also perform the following operations:
     - Click a node in the **Node Name** column to go to the node configuration page.
     - Search for instances by condition, such as the ID, committer, node name, source connection type, and destination connection type. Sort search results based on the number of synchronized data entries or the size of synchronized data.

3.5.4. Manage connection permissions

DataWorks allows you to share connections among workspaces by managing permissions on the connections. After connections are shared, you can view the shared connections in the target workspaces. This topic describes how to manage permissions on connections and view shared connections.

Context

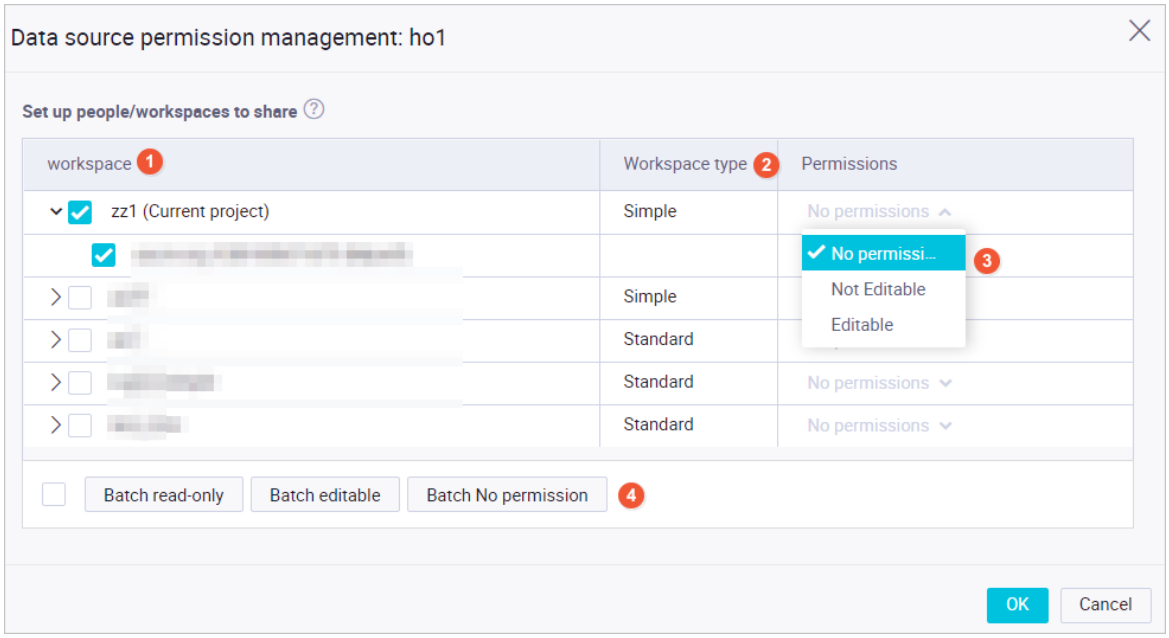
The configurations of a connection include sensitive information such as the endpoint of the data store, username, and password. Common developers only need to reference the connection to access the data store. Disclosing too much sensitive information or allowing everyone to modify the configurations of the connection may cause security risks. If multiple users modify the configurations of a connection, the data store may fail to be connected. In this way, the nodes that reference the

connection may fail.

Data Integration provides strict permission control. Only connection creators can manage the permissions on connections. They can grant permissions on connections to a specified workspace or user.

Go to the Data Source page

1. Log on to the DataWorks console.
2. On the DataStudio page, click in the upper-left corner and choose **All Products > Data Integration**.
3. On the page that appears, click **Data Source** in the left-side navigation pane.
4. On the page that appears, find the target connection and click **Modify Permission** in the **Actions** column.
5. In the **Data source permission management** dialog box, set the parameters as described in the following table.



No.	Parameter	Description

No.	Parameter	Description
1	**Workspace**	All workspaces that the current user joins and all members in each workspace. You can share the connection with several or all members in a workspace. ◦ If no permission is set for a connection, the connection inherits the permissions from the connection that is created earlier than the current one. ◦ When you configure the permissions on a connection for a workspace, the permissions apply to all members in the workspace. Members that join the workspace after the permission configuration also have the specified permissions. After you configure the permissions for a workspace, you can still configure the permissions for a specific user in the workspace. For example, after you set the permission on a connection to **No permission** for a workspace, you can still set the permission of a specific user in the workspace to **Editable**. ◦ You can configure the permissions on a connection for members in the current workspace. ◦ Only the creator of a connection can modify and share the connection. Other users including the workspace administrator cannot modify the connection. ◦ A workspace administrator can use a connection only after the workspace administrator is granted the required permission.
2	**Workspace type**	The type of each workspace. Valid values: **Simple** and **Standard**.
3	**Permissions**	The permission of a workspace or a member on the connection. Valid values: ◦ **No permission**: The workspace or member has no permission on the connection. ◦ **Not Editable**: The workspace or member can use the connection but cannot modify or view the configurations of connection. ◦ **Editable**: The workspace or member can use and modify the connection. **Note** If you grant the Editable permission with a workspace or member, the workspace or member can modify the connection. Exercise caution when you grant the Editable permission.
4	Batch operations	The operations that you can perform on the selected workspaces or members at a time. Valid values: **Batch read-only**, **Batch editable**, and **Batch No permission**.

6. Click **OK**.

You can share connections across workspaces based on the following rules:

- Between workspaces in simple mode:
  - If the source workspace is upgraded to the standard mode, connections in the production environment are shared.
  - If the target workspace is upgraded to the standard mode, a connection is shared to both the development environment and production environment with the same content.
- From a workspace in simple mode to a workspace in standard mode: A connection is shared to both the development environment and production environment with the same content.
- Between workspaces in standard mode: Connections in the development environment and production environment are shared to the corresponding environment separately.
- From a workspace in standard mode to a workspace in simple mode:
  - You can share connections in both the production environment and development environment. Only connections in the production environment or development environment exist in the target workspace. If you share a connection in both environments, the newly shared one overrides the existing one in the target workspace.
  - If the target workspace is upgraded to the standard mode, the shared connection exists in both the development environment and production environment with the same content.

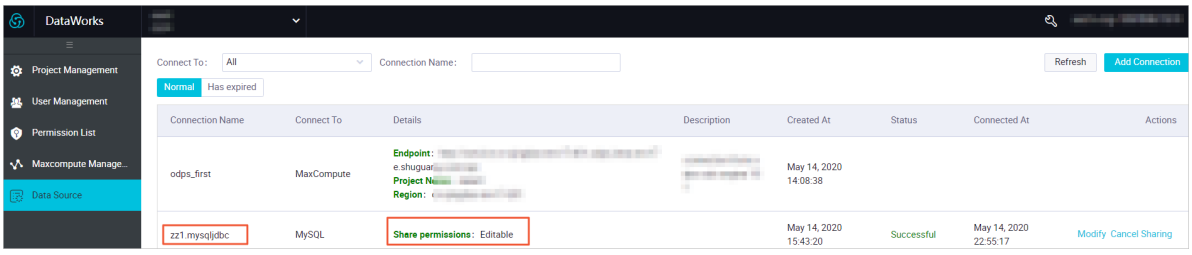
View shared connections

In the top navigation bar, select a workspace with connections shared from other workspaces from the drop-down list in the upper-left corner. The **Data Source** page of the selected workspace appears. On this page, you can view shared connections on the **Normal** and **Has expired** tabs.

- **Normal tab**

  On the Normal tab, you can view the information about each connection, including the connection name, connection type, permission details, connection description, creation time, connection status, and the time when the data store was last connected.

  The permission information appears in the **Details** column of the target connection. A shared connection is named in the Name of the workspace that shares the connection.Connection name format.

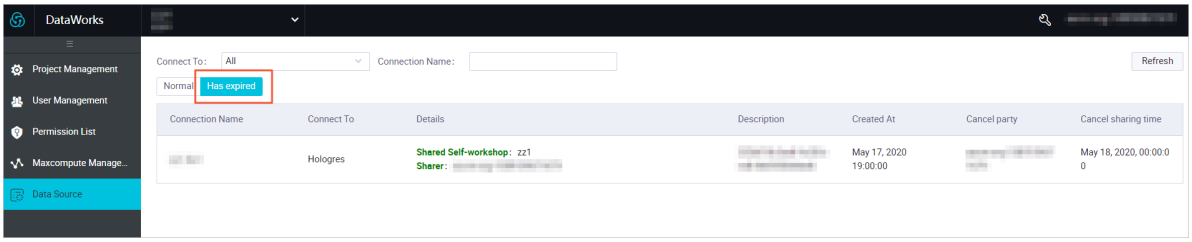


  If the current user has the Editable permission on the connection, **Modify** appears in the Actions column.

- **Has expired tab**

  On the **Has expired** tab, you can view the connections for which your permissions have expired.

In the **Cancel party** column, you can view the member who revoked the permissions. In the **Created at** column, you can view the time when the permissions were revoked. The information helps you locate the cause of connection failures.



3.5.5. Configure a MySQL data source

DataWorks provides MySQL Reader and MySQL Writer for you to read data from and write data to MySQL data sources. You can use the codeless user interface (UI) or code editor to configure synchronization nodes for MySQL data sources.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, choose **Data Source > Data Sources** in the left-side navigation pane. The **Data Source** page under **Workspace Management** appears.
2. Click **Add data source** in the upper-right corner.
3. In the **Add data source** dialog box, click **MySQL** in the Relational Databases section.
4. In the **Add MySQL data source** dialog box, set the parameters.

   You can use one of the following modes to add a MySQL data source: Alibaba Cloud instance mode and connection string mode.

   - The following table describes the parameters that you must set if you add a MySQL data source by using the Alibaba Cloud instance mode.

Parameter	Description
Data source type	The mode in which the data source is added. Set this parameter to **Alibaba Cloud instance mode**.
Data Source Name	The name of the data source. The name can contain letters, digits, and underscores (_), and must start with a letter.
Data source description	The description of the data source. The description can be up to 80 characters in length.

Parameter	Description
Environment	The environment in which the data source is used. Valid values: **Development** and **Production**. **Note** This parameter is displayed only when the workspace is in standard mode.
RDS instance ID	The ID of the ApsaraDB RDS for MySQL instance. You can log on to the ApsaraDB RDS console to view the ID.
RDS instance account ID	The ID of the Apsara Stack tenant account that is used to purchase the ApsaraDB RDS for MySQL instance.
Default Database Name	The name of the database that you created in the ApsaraDB RDS for MySQL console.
User name	The username that is used to connect to the database.
Password	The password that is used to connect to the database.

- The following table describes the parameters that you must set if you add a MySQL data source by using the connection string mode.

Parameter	Description
Data source type	The mode in which the data source is added. Set this parameter to **Connection string mode**.
Data Source Name	The name of the data source. The name can contain letters, digits, and underscores (_), and must start with a letter.
Data source description	The description of the data source. The description can be up to 80 characters in length.
Environment	The environment in which the data source is used. Valid values: **Development** and **Production**. **Note** This parameter is displayed only when the workspace is in standard mode.
JDBC URL	The Java Database Connectivity (JDBC) URL of the database. Specify a value for this parameter in the format of `jdbc:mysql://ServerIP:Port/Database`.
User name	The username that is used to connect to the database.
Password	The password that is used to connect to the database.

Parameter	Description
Whether the data source is within the VPC	Specifies whether to connect to the data source by using a virtual private cloud (VPC). If you cannot connect to the Elastic Compute Service (ECS) instance on which the data source resides but can connect to the VPC to which the data source belongs, select the check box.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

What's next

You have learned how to configure a MySQL data source. You can proceed to subsequent tutorials. The subsequent tutorials describe how to configure MySQL Reader or MySQL Writer. For more information, see Configure the MySQL reader or Configure MySQL Writer.

3.5.6. Configure an SQL Server data source

DataWorks provides SQL Server Reader and SQL Server Writer for you to read data from and write data to SQL Server data sources. You can use the codeless user interface (UI) or code editor to configure synchronization nodes for SQL Server data sources.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, choose **Data Source > Data Sources** in the left-side navigation pane. The **Data Source** page under **Workspace Management** appears.
2. Click **Add data source** in the upper-right corner.
3. In the **Add data source** dialog box, click **SQLServer** in the Relational Databases section.
4. In the **Add SQLServer data source** dialog box, configure the parameters.

   You can set the Data source type parameter to **Alibaba Cloud instance mode** or **Connection string mode** for an SQL Server data source.

   - The following table describes the parameters that appear after you set **Data source type** to **Alibaba Cloud instance mode**.

Parameter	Description
Data source type	The type of the data source. Set the parameter to **Alibaba Cloud instance mode**.
Data Source Name	The name of the data source. The name must contain letters, digits, and underscores (_) and start with a letter.
Description	The description of the data source. The description can be a maximum of 80 characters in length.

Parameter	Description
Environment	The environment in which the data source is used. Valid values: **Development** and **Production**. **Note** This parameter is displayed only when the workspace is in standard mode.
RDS instance ID	The ID of the ApsaraDB RDS for SQL Server instance. You can view the ID in the ApsaraDB RDS console.
RDS instance account ID	The ID of the Apsara Stack tenant account that is used to purchase the ApsaraDB RDS for SQL Server instance.
Database name	The name of the ApsaraDB RDS for SQL Server database.
User name	The username that is used to connect to the database.
Password	The password that is used to connect to the database.

- The following table describes the parameters that appear after you set **Data source type** to **Connection string mode**.

Parameter	Description
Data source type	The type of the data source. Set the parameter to **Connection string mode**.
Data Source Name	The name of the data source. The name must contain letters, digits, and underscores (_) and start with a letter.
Description	The description of the data source. The description can be a maximum of 80 characters in length.
Environment	The environment in which the data source is used. Valid values: **Development** and **Production**. **Note** This parameter is displayed only when the workspace is in standard mode.
JDBC URL	The Java Database Connectivity (JDBC) URL of the database, in the format of `jdbc:sqlserver://ServerIP:Port;DatabaseName=Database`.
User name	The username that is used to connect to the database.
Password	The password that is used to connect to the database.

Parameter	Description
Whether the data source is in a VPC	Specifies whether to connect to the data source by using a VPC. If you cannot connect to the ECS instance where the data source is located but can connect to the VPC to which the data source belongs, select the check box.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

What's next

You have learned how to configure an SQL Server data source. You can proceed to subsequent tutorials. The subsequent tutorials describe how to configure SQL Server Reader or SQL Server Writer. For more information, see Configure SQL Server Reader or Configure SQL Server Writer.

3.5.7. Configure a PostgreSQL connection

A PostgreSQL connection allows you to read data from and write data to PostgreSQL by using PostgreSQL Reader and Writer. You can configure sync nodes for PostgreSQL by using the codeless UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, choose **Data Source > Data Sources** in the left-side navigation pane. The **Data Source** page under **Workspace Management** appears.
2. Click **Add data source** in the upper-right corner.
3. In the **Add Connection** dialog box, click **PostgreSQL** in the Relational Databases section.
4. In the **Add PostgreSQL Connection** dialog box, set the required parameters.

   You can set the Connect To parameter to **ApsaraDB for RDS** or **Connection Mode** for a PostgreSQL connection.

   - The following table describes the parameters that appear after you set the **Connect To** parameter to **ApsaraDB for RDS**.

Parameter	Description
Connect To	The type of the connection. In this example, set the value to **ApsaraDB for RDS**.
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.

Parameter	Description
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
RDS Instance ID	The ID of the ApsaraDB RDS for PostgreSQL instance. You can view the ID in the ApsaraDB for RDS console.
RDS Instance Account ID	The ID of the Apsara Stack tenant account that is used to purchase the ApsaraDB RDS for PostgreSQL instance. You can view your account ID on the **Security Settings** page.
Database Name	The name of the database.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.

- The following table describes the parameters that appear after you set the **Connect To** parameter to **Connection Mode**.

Parameter	Description
Connect To	The type of the connection. In this example, set the value to **Connection Mode**.
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
JDBC URL	The JDBC URL of the database, in the format of `jdbc:postgresql://ServerIP:Port/Database`.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.
Enable reverse VPC access	Specifies whether to enable reverse VPC access. Select the Enable check box if you cannot directly access the data store on an ECS instance but can access it by using a VPC.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

3.5.8. Configure an Oracle connection

An Oracle connection allows you to read data from and write data to Oracle by using Oracle Reader and Writer. You can configure sync nodes for Oracle by using the codeless UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, choose **Data Source > Data Sources** in the left-side navigation pane. The **Data Source** page under **Workspace Management** appears.
2. Click **Add data source** in the upper-right corner.
3. In the **Add Connection** dialog box, click **Oracle** in the Relational Databases section.
4. In the **Add Oracle Connection** dialog box, set the parameters as required.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
JDBC URL	The JDBC URL of the database, in the format of `jdbc:oracle:thin:@ServerIP:Port:Database`.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.
Enable reverse VPC access	Specifies whether to enable reverse VPC access. Select the Enable check box if you cannot directly access the data store on an ECS instance but can access it by using a VPC.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

3.5.9. Configure a Dameng connection

A Dameng connection allows you to read data from and write data to Dameng by using Dameng Reader and Writer. You can configure sync nodes for Dameng by using the codeless UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the ☰ icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, choose **Data Source > Data Sources** in the left-side navigation pane. The **Data Source** page under **Workspace Management** appears.
2. Click **Add data source** in the upper-right corner.
3. In the **Add Connection** dialog box, click **DM** in the Relational Databases section.
4. In the **Add DM Connection** dialog box, set the parameters as required.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
JDBC URL	The JDBC URL of the database, in the format of `jdbc:dm://ServerIP:Port/Database`.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.
Enable reverse VPC access	Specifies whether to enable reverse VPC access. Select the Enable check box if you cannot directly access the data store on an ECS instance but can access it by using a VPC.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

3.5.10. Configure a DRDS connection

A DRDS connection allows you to read data from and write data to DRDS by using DRDS Reader and Writer. You can configure sync nodes for DRDS by using the codeless UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, choose **Data Source > Data Sources** in the left-side navigation pane. The **Data Source** page under **Workspace Management** appears.
2. Click **Add data source** in the upper-right corner.
3. In the **Add Connection** dialog box, click **DRDS** in the Relational Databases section.
4. In the **Add DRDS Connection** dialog box, set the parameters as required.

   You can set the Connect To parameter to **ApsaraDB for DRDS** or **Connection Mode** for a DRDS connection.

   - The following table describes the parameters that appear after you set the Connect To parameter to **ApsaraDB for DRDS**.

Parameter	Description
Connect To	The type of the connection. In this example, set the value to **ApsaraDB for DRDS**.
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
Instance ID	The ID of the DRDS instance. You can view the ID in the DRDS console.
Tenant Account ID	The ID of the Apsara Stack tenant account that is used to purchase the DRDS instance. You can view your account ID on the **Security Settings** page.
Database Name	The name of the database.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.

   - The following table describes the parameters that appear after you set the Connect To

parameter to **Connection Mode**.

Parameter	Description
Connect To	The type of the connection. In this example, set the value to **Connection Mode**.
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
JDBC URL	The JDBC URL of the database, in the format of `jdbc:mysql://ServerIP:Port/Database`.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

What's next

Now you have learned how to configure a DRDS connection. You can proceed with the next tutorial. In the next tutorial, you will learn how to configure DRDS Reader and Writer. For more information, see Configure the DRDS reader.

3.5.11. Configure a PolarDB connection

A PolarDB connection allows you to read data from and write data to PolarDB by using PolarDB Reader and Writer. You can configure sync nodes for PolarDB by using the codeless UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, choose **Data Source > Data Sources** in the left-side navigation pane. The **Data Source** page under **Workspace Management** appears.
2. Click **Add data source** in the upper-right corner.
3. In the **Add Connection** dialog box, click **POLARDB** in the Relational Databases section.

4. In the **Add POLARDB Connection** dialog box, set the parameters as required.

Parameter	Description
Connect To	The type of the connection. In this example, set the value to **Connection Mode**.
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
Database Type	The type of the database. Valid values: **MySql** and **Postgresql**.
JDBC URL	The JDBC URL of the database, in the format of `jdbc:mysql://ServerIP:Port/Database`.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

3.5.12. Configure a HybridDB for MySQL connection

A HybridDB for MySQL connection allows you to read data from and write data to HybridDB for MySQL by using HybridDB for MySQL Reader and Writer. You can configure sync nodes for HybridDB for MySQL by using the codeless UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, choose **Data Source > Data Sources** in the left-side navigation pane. The **Data Source** page under **Workspace Management** appears.
2. Click **Add data source** in the upper-right corner.
3. In the **Add Connection** dialog box, click **HybridDB for MySQL** in the Relational Databases section.

4. In the **Add HybridDB for MySQL Connection** dialog box, set the parameters as required.

Parameter	Description
Connect To	The type of the connection. In this example, set the value to **ApsaraDB for AnalyticDB**.
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
Instance ID	The ID of the HybridDB for MySQL instance. You can view the ID in the HybridDB for MySQL console.
Tenant Account ID	The ID of the Apsara Stack tenant account that is used to purchase the HybridDB for MySQL instance.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

3.5.13. Configure a HybridDB for PostgreSQL connection

A HybridDB for PostgreSQL connection allows you to read data from and write data to HybridDB for PostgreSQL by using HybridDB for PostgreSQL Reader and Writer. You can configure sync nodes for HybridDB for PostgreSQL by using the codeless UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, choose **Data Source > Data Sources** in the left-side navigation pane. The **Data Source** page under **Workspace Management** appears.
2. Click **Add data source** in the upper-right corner.
3. In the **Add Connection** dialog box, click **HybridDB for PostgreSQL** in the Relational Databases

section.

4. In the **Add HybridDB for PostgreSQL Connection** dialog box, set the parameters as required.

Parameter	Description
Connect To	The type of the connection. In this example, set the value to **ApsaraDB for AnalyticDB**.
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
Instance ID	The ID of the HybridDB for PostgreSQL instance. You can view the ID in the HybridDB for PostgreSQL console.
Tenant Account ID	The ID of the Apsara Stack tenant account that is used to purchase the HybridDB for PostgreSQL instance. You can view your account ID on the **Security Settings** page.
Database Name	The name of the database.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

3.5.14. Configure an ApsaraDB for OceanBase connection

An ApsaraDB for OceanBase connection allows you to read data from and write data to ApsaraDB for OceanBase by using ApsaraDB for OceanBase Reader and Writer. You can configure sync nodes for ApsaraDB for OceanBase by using the codeless UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.

iii. On the page that appears, choose **Data Source > Data Sources** in the left-side navigation pane. The **Data Source** page under **Workspace Management** appears.

2. Click **Add data source** in the upper-right corner.
3. In the **Add Connection** dialog box, click **ApsaraDB for OceanBase** in the Big Data Storage Systems section.
4. In the **Add ApsaraDB for OceanBase Connection** dialog box, set the parameters as required.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
JDBC URL	The JDBC URL of the ApsaraDB for OceanBase database, in the format `jdbc:oceanbase://ip:port/database`.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

3.5.15. Configure a MaxCompute connection

A MaxCompute connection allows you to read data from and write data to MaxCompute by using MaxCompute Reader and Writer.

Context

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, choose **Data Source > Data Sources** in the left-side navigation pane. The **Data Source** page under **Workspace Management** appears.
2. Click **Add data source** in the upper-right corner.

3. In the **Add Connection** dialog box, click **MaxCompute** in the Big Data Storage Systems section.
4. In the **Add MaxCompute Connection** dialog box, set the parameters as required.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
ODPS Endpoint	The endpoint of the MaxCompute project. This parameter is read-only, and the value is automatically obtained from system configurations.
Tunnel Endpoint	The endpoint of the MaxCompute Tunnel service.
MaxCompute Project Name	The name of the MaxCompute project.
AccessKey ID	The AccessKey ID for connecting to the MaxCompute project.
AccessKey Secret	The AccessKey secret for connecting to the MaxCompute project.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

3.5.16. Configure a DataHub connection

DataHub offers a comprehensive data import scheme to support fast computing for large amounts of data.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the ☰ icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, choose **Data Source > Data Sources** in the left-side navigation pane. The **Data Source** page under **Workspace Management** appears.
2. Click **Add data source** in the upper-right corner.
3. In the **Add Connection** dialog box, click **DataHub** in the Big Data Storage Systems section.
4. In the **Add DataHub Connection** dialog box, set the parameters as required.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
DataHub Endpoint	The endpoint of DataHub. This parameter is read-only, and the value is automatically obtained from system configurations.
DataHub Project	The ID of the DataHub project.
AccessKey ID	The AccessKey ID for connecting to the DataHub project. You can view the AccessKey ID on the **User Info** page.
AceessKey Secret	The AccessKey secret for connecting to the DataHub project.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

3.5.17. Configure an AnalyticDB for MySQL connection

An AnalyticDB for MySQL connection allows you to write data to AnalyticDB for MySQL by using AnalyticDB for MySQL Writer. You can configure sync nodes for AnalyticDB for MySQL by using the codeless UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, choose **Data Source > Data Sources** in the left-side navigation pane. The **Data Source** page under **Workspace Management** appears.
2. Click **Add data source** in the upper-right corner.
3. In the **Add Connection** dialog box, click **ADS** in the Big Data Storage Systems section.
4. In the **Add ADS Connection** dialog box, set the parameters as required.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
Connection URL	The connection URL of AnalyticDB for MySQL, in the format of `Address:Port`.
Database	The name of the database.
AccessKey ID	The AccessKey ID for connecting to the AnalyticDB for MySQL database.
AccessKey Secret	The AccessKey secret for connecting to the AnalyticDB for MySQL database.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

3.5.18. Configure a Vertica connection

A Vertica connection allows you to read data from and write data to Vertica by using Vertica Reader and Writer. You can configure sync nodes for Vertica by using the UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, choose **Data Source > Data Sources** in the left-side navigation pane. The **Data Source** page under **Workspace Management** appears.
2. Click **Add data source** in the upper-right corner.
3. In the **Add Connection** dialog box, click **Vertica** in the Big Data Storage Systems section.
4. In the **Add Vertica Connection** dialog box, set the parameters as required.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.

Parameter	Description
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
JDBC URL	The JDBC URL of the Vertica database, in the format of `jdbc:vertica://Server IP:Port/Database`.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.
Enable reverse VPC access	Specifies whether to enable reverse VPC access. Select the Enable check box if you cannot directly access the data store on an ECS instance but can access it by using a VPC.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

3.5.19. Configure a GBase8a connection

A GBase8a connection allows you to read data from and write data to GBase8a by using GBase8a Reader and Writer. You can configure sync nodes for GBase8a by using the codeless UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, choose **Data Source > Data Sources** in the left-side navigation pane. The **Data Source** page under **Workspace Management** appears.
2. Click **Add data source** in the upper-right corner.
3. In the **Add Connection** dialog box, click **GBase8a** in the Big Data Storage Systems section.
4. In the **Add GBase8a Connection** dialog box, set the parameters as required.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.

Parameter	Description
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
JDBC URL	The JDBC URL of the database, in the format of `jdbc:mysql://ServerIP:Port/Database`.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.
Enable reverse VPC access	Specifies whether to enable reverse VPC access. Select the Enable check box if you cannot directly access the data store on an ECS instance but can access it by using a VPC.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

3.5.20. Configure a Lightning connection

MaxCompute Lightning is an interactive query service that MaxCompute provides. MaxCompute Lightning complies with the PostgreSQL standards and syntax and allows you to use common tools and standard SQL to query and analyze data in MaxCompute projects.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, choose **Data Source > Data Sources** in the left-side navigation pane. The **Data Source** page under **Workspace Management** appears.
2. Click **Add data source** in the upper-right corner.
3. In the **Add Connection** dialog box, click **Lightning** in the Big Data Storage Systems section.
4. In the **Add Lightning Connection** dialog box, set the parameters as required.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.

Parameter	Description
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
Host	The endpoint of the MaxCompute Lightning server. Default value: `seahawks.aliyun-inc.com`.
Port	The port number of the MaxCompute Lightning server. Default value: 8099.
Database Name	The name of the database.
Username and **Password**	The username and password that you can use to connect to the database.
ODPS Endpoint	The endpoint of MaxCompute.
MaxCompute Project Name	The name of the MaxCompute project.
AccessKey ID	The AccessKey ID for connecting to the MaxCompute Lightning server.
AccessKey Secret	The AccessKey secret for connecting to the MaxCompute Lightning server.
JDBC Extension Parameters	The extension parameters used to establish a JDBC connection to MaxCompute Lightning. In this field, `prepareThreshold=0` is added by default and cannot be deleted. Otherwise, you cannot connect to MaxCompute Lightning.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

3.5.21. Configure an HBase data source

DataWorks provides HBase Reader and HBase Writer for you to read data from and write data to HBase data sources. You can use the codeless user interface (UI) or code editor to configure synchronization nodes for HBase data sources. This topic describes how to configure an HBase data source.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.

ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.

iii. On the page that appears, choose **Data Source > Data Sources** in the left-side navigation pane. The **Data Source** page under **Workspace Management** appears.

2. Click **Add data source** in the upper-right corner.
3. In the **Add data source** dialog box, click **HBase** in the Big Data Storage Systems section.
4. In the **Add HBase data source** dialog box, set the parameters.

Parameter	Description
Data Source Name	The name of the data source. The name can contain letters, digits, and underscores (_), and must start with a letter.
Data source description	The description of the data source. The description can be up to 80 characters in length.
Environment	The environment in which the data source is used. Valid values: **Development** and **Production**. **Note** This parameter is displayed only when the workspace is in standard mode.
Configuration information	The configuration information of the HBase cluster. You can convert the hbase-site.xml file to the JSON format. Then, add HBase client properties, such as cache and batch for scan, to optimize the interaction between the cluster and the client. You must configure different information based on the edition of HBase in use. ◦ If you are using ApsaraDB for HBase Standard Edition or a less advanced edition, the default configuration information is used. You need to enter only the corresponding ZooKeeper information. ◦ If you are using a more advanced edition than ApsaraDB for HBase Standard Edition, the endpoint parameter specific to advanced editions is used for connection, and the zookeeper.quorum parameter is not used. The following code provides an example of the configuration information if you use ApsaraDB for HBase Performance-enhanced Edition (Lindorm): `"hbaseConfig": {` `"hbase.client.connection.impl" :` `"com.alibaba.hbase.client.AliHBaseUEConnection",` `"hbase.client.endpoint" : "host:30020",` `"hbase.client.username" : "root",` `"hbase.client.password" : "root"` `}`

Parameter	Description
Special Authentication Method	Specifies whether to enable identity authentication. Default value: **None**. You can alternatively set this parameter to **Kerberos Authentication**. For more information about Kerberos authentication, see Configure Kerberos authentication.
Keytab File	If you set the **Special Authentication Method** parameter to **Kerberos Authentication**, you must select the specified keytab file from the Keytab File drop-down list. If no keytab file is available, you can click **Add Authentication File** to upload a keytab file.
CONF File	If you set the **Special Authentication Method** parameter to **Kerberos Authentication**, you must select the specified CONF file from the CONF File drop-down list. If no CONF file is available, you can click **Add Authentication File** to upload a CONF file.
principal	The Kerberos principal. Specify this parameter in the format of `Principal name/Instance name@Domain name`, such as `****/hadoopclient@**.***`.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

3.5.22. Configure a Hologres connection

A Hologres connection allows you to read data from and write data to Hologres by using Hologres Reader and Writer. You can configure sync nodes for Hologres by using the codeless UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, choose **Data Source > Data Sources** in the left-side navigation pane. The **Data Source** page under **Workspace Management** appears.
2. Click **Add data source** in the upper-right corner.
3. In the **Add Connection** dialog box, click **Hologres** in the Big Data Storage Systems section.
4. In the **Add Hologres Connection** dialog box, set the parameters as required.

Parameter	Description
Connect To	The type of the connection. Default value: **ApsaraDB for RDS**.
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
Instance ID	The ID of the Hologres instance.
Database Name	The name of the database in the Hologres instance.
AccessKey ID	The AccessKey ID for connecting to the Hologres database.
AccessKey Secret	The AccessKey secret for connecting to the Hologres database.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

3.5.23. Add a Hive data source

This topic describes how to configure a Hive data source. A Hive data source allows you to read data from and write data to Hive. You can configure synchronization nodes by using the codeless user interface (UI) or code editor.

Context

Workspaces in standard mode support the data source isolation feature. You can add data sources separately for the development and production environments to isolate the data sources. This helps keep your data secure. For more information about the feature, see Connection isolation.

If you use Object Storage Service (OSS) as the underlying storage, you must take note of the following items:

- The value of the defaultFS parameter must start with oss://. For example, the value can be `oss://IP:PORT` or `oss://nameservice`.
- You must configure the parameters that are required for connecting to OSS in the advanced parameters. The following sample code provides an example:

```
{
        "hiveConfig":{
            "fs.oss.accessKeyId":"<yourAccessKeyId>",
                "fs.oss.accessKeySecret":"<yourAccessKeySecret>",
                "fs.oss.endpoint":"oss-cn-<yourRegion>-internal.aliyuncs.com"
        }
    }
```

Limits

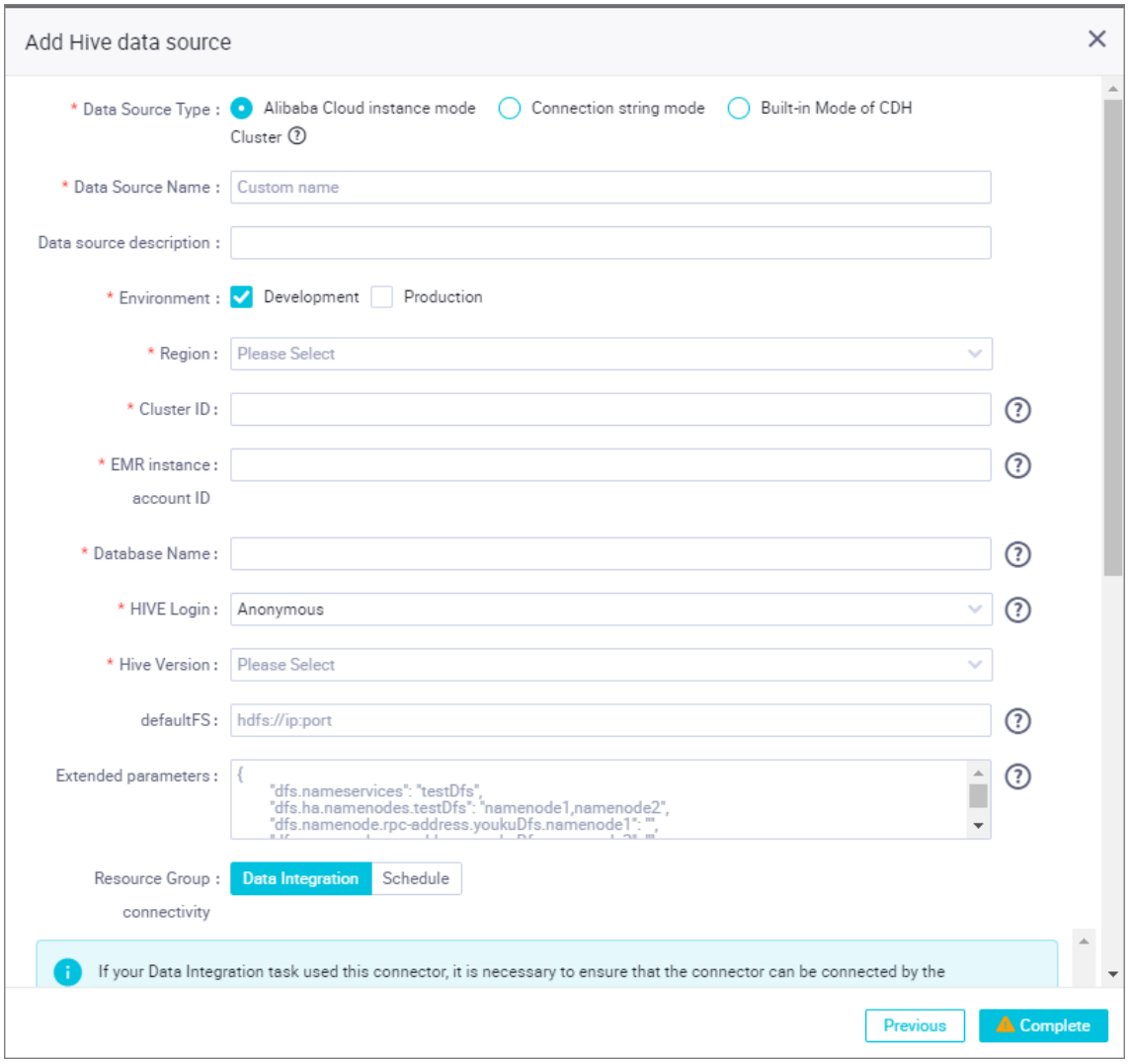
- DataWorks supports only Hive 2.3.3 and Hive 2.3.5.
- Hive data sources support only Kerberos authentication.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, choose **Data Source > Data Sources** in the left-side navigation pane. The **Data Source** page under **Workspace Management** appears.
2. Click **Add data source** in the upper-right corner.
3. In the **Add data source** dialog box, click **Hive** in the Big Data Storage section.
4. In the **Add Hive data source** dialog box, set the parameters as required.

   You can use one of the following modes to add a Hive data source: **Alibaba Cloud instance mode**, **Connection string mode**, and **Built-in Mode of CDH Cluster**.

   - The following table describes the parameters for adding a Hive data source in **Alibaba Cloud instance mode**.

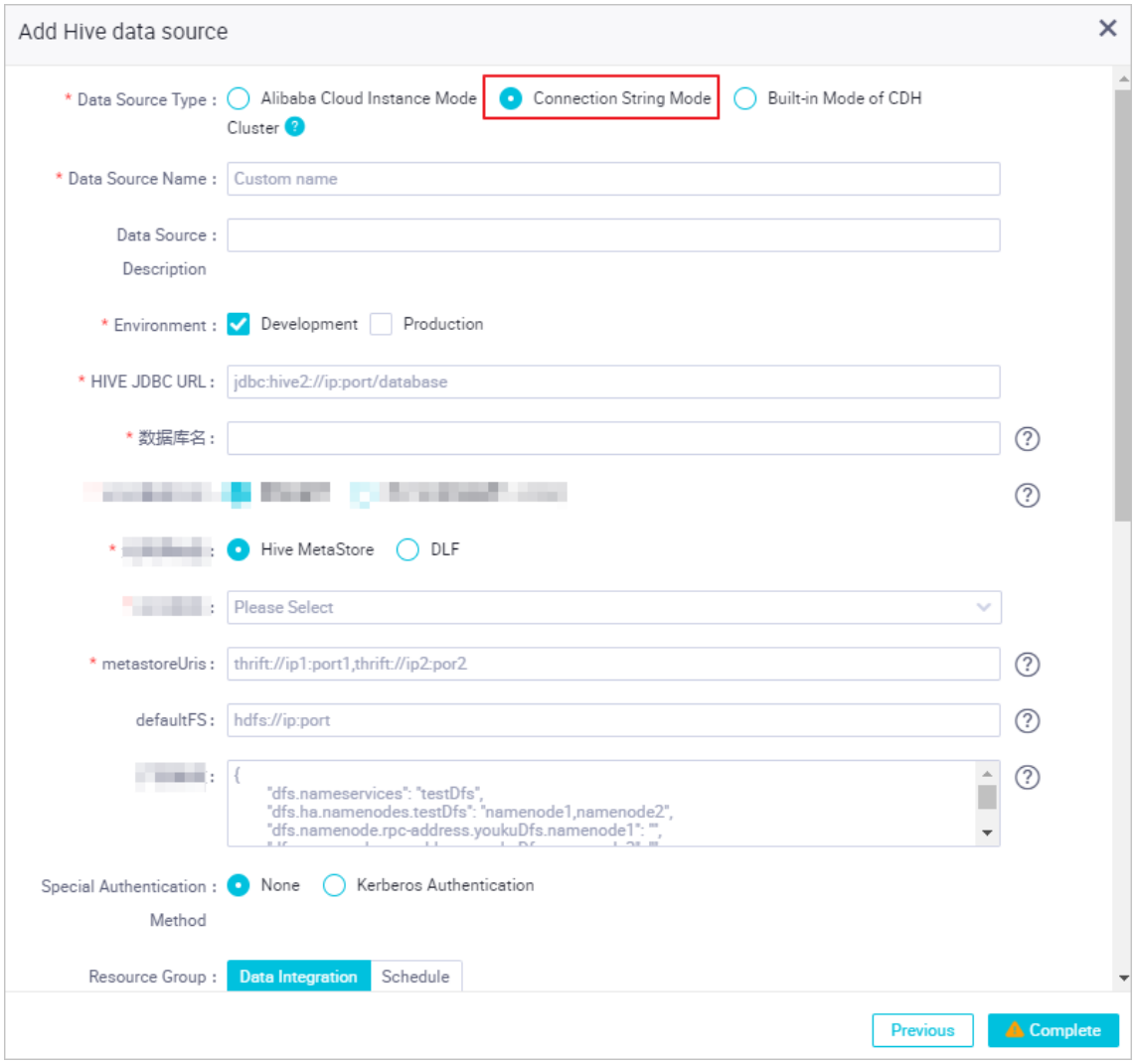


Parameter	Description
Data source type	The mode in which the data source is added. Set this parameter to **Alibaba Cloud instance mode**.
Data Source Name	The name of the data source. The name can contain letters, digits, and underscores (_) and must start with a letter.
Data source description	The description of the data source. The description can be up to 80 characters in length.
Environment	The environment in which the data source is used. Valid values: **Development** and **Production**. **Note** This parameter is displayed only when the workspace is in standard mode.
Cluster ID	You can log on to the EMR console to obtain the ID of the EMR cluster.

Parameter	Description
EMR instance account ID	The ID of the Alibaba Cloud account that is used to purchase the EMR cluster. You can view the ID of the account on the **Security Settings** page.
Database Name	The name of the Hive database that you want to access.
HIVE Login	The mode that is used to connect to the Hive database. Valid values: **Login with username and password** and **Anonymous**. If you select **Login with username and password**, enter the **username** and **password** that you can use to connect to the Hive database.
Hive Version	The Hive version that you want to use.
defaultFS	The address of the NameNode node in the Active state in Hadoop Distributed File System (HDFS), in the format of `hdfs://ip:port`.
Extended parameters	The advanced parameters of Hive, such as those related to high availability (HA). The following sample code provides an example: `"hadoopConfig":{` `"dfs.nameservices": "testDfs",` `"dfs.ha.namenodes.testDfs": "namenode1,namenode2",` `"dfs.namenode.rpc-address.youkuDfs.namenode1": "",` `"dfs.namenode.rpc-address.youkuDfs.namenode2": "",` `"dfs.client.failover.proxy.provider.testDfs` `"org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider"` `}`
Special Authentication Method	Specifies whether to enable identity authentication. Default value: **None**. You can alternatively set this parameter to **Kerberos Authentication**. For more information about Kerberos authentication, see Configure Kerberos authentication.

Parameter	Description
Keytab File	If you set the **Special Authentication Method** parameter to **Kerberos Authentication**, you must select the specified keytab file from the Keytab File drop-down list. If no keytab file is available, you can click **Add Authentication File** to upload a keytab file.
CONF File	If you set the **Special Authentication Method** parameter to **Kerberos Authentication**, you must select the specified CONF file from the CONF File drop-down list. If no CONF file is available, you can click **Add Authentication File** to upload a CONF file.
principal	The Kerberos principal. Specify this parameter in the format of Principal name/Instance name@Domain name, such as ****/hadoopclient@**.***.

- The following table describes the parameters for adding a Hive data source in **Connection string mode**.

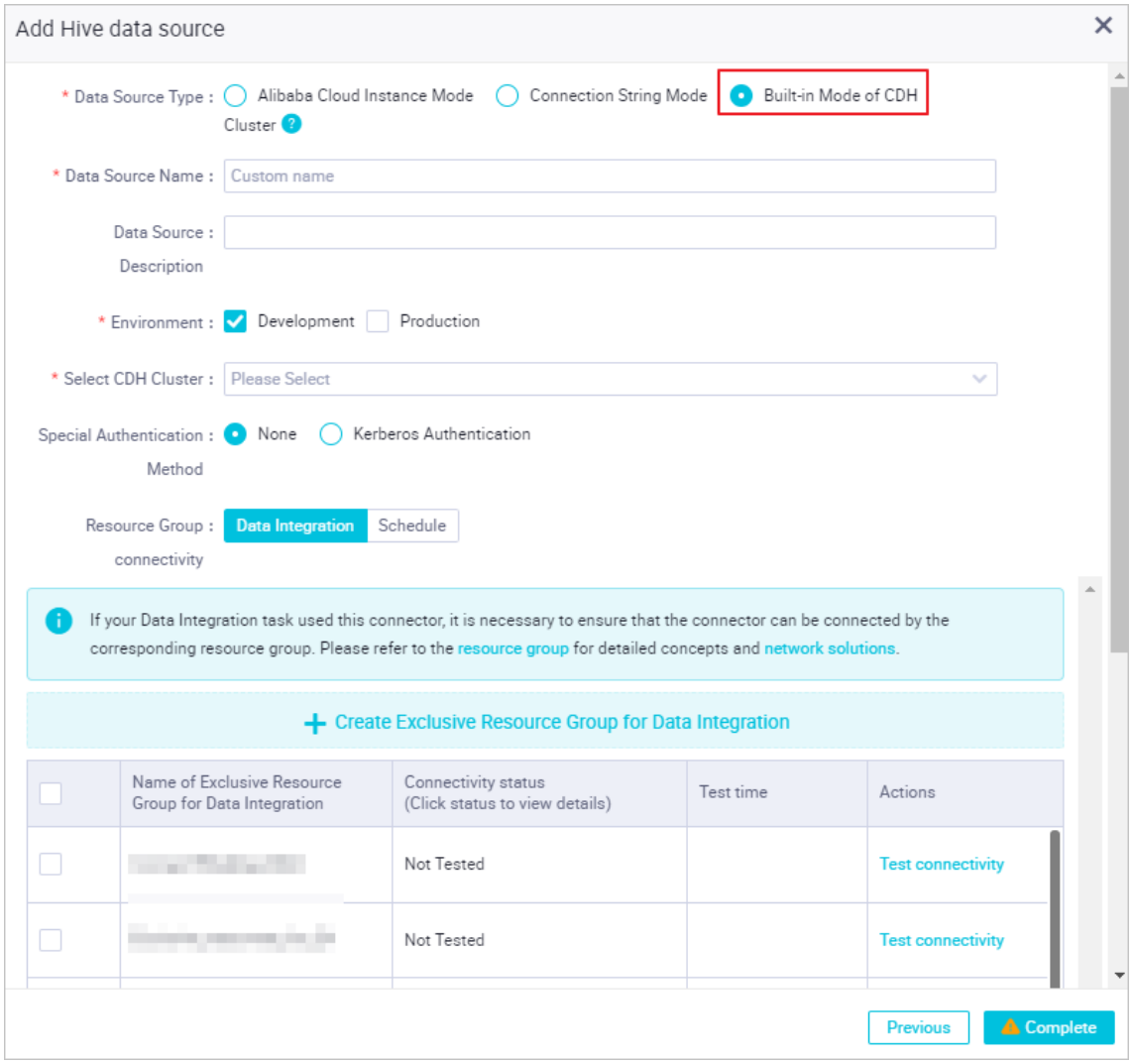


Parameter	Description
Data source type	The mode in which the data source is added. Set this parameter to **Connection string mode**.
Data Source Name	The name of the data source. The name can contain letters, digits, and underscores (_) and must start with a letter.
Data source description	The description of the data source. The description can be up to 80 characters in length.
Environment	The environment in which the data source is used. Valid values: **Development** and **Production**. **Note** This parameter is displayed only when the workspace is in standard mode.
HIVE JDBC URL	The Java Database Connectivity (JDBC) URL of the Hive metadatabase.

Parameter	Description
Database name	The name of the Hive database that you want to access. You can run the `show databases` command on the Hive client to query the created databases.
HIVE Login	The mode that is used to connect to the Hive database. Valid values: **Login with username and password** and **Anonymous**. If you select **Login with username and password**, enter the **username** and **password** that you can use to connect to the Hive database.
Hive Version	The Hive version that you want to use.
defaultFS	The address of the NameNode node in the Active state in Hadoop Distributed File System (HDFS), in the format of `hdfs://ip:port`.
Extended parameters	The advanced parameters of Hive, such as those related to HA. The following sample code provides an example: `"hadoopConfig":{` `"dfs.nameservices": "testDfs",` `"dfs.ha.namenodes.testDfs": "namenode1,namenode2",` `"dfs.namenode.rpc-address.youkuDfs.namenode1": "",` `"dfs.namenode.rpc-address.youkuDfs.namenode2": "",` `"dfs.client.failover.proxy.provider.testDfs` `"org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider"` `}`
Special Authentication Method	Specifies whether to enable identity authentication. Default value: **None**. You can alternatively set this parameter to **Kerberos Authentication**. For more information about Kerberos authentication, see Configure Kerberos authentication.
Keytab File	If you set the **Special Authentication Method** parameter to **Kerberos Authentication**, you must select the specified keytab file from the Keytab File drop-down list. If no keytab file is available, you can click **Add Authentication File** to upload a keytab file.
CONF File	If you set the **Special Authentication Method** parameter to **Kerberos Authentication**, you must select the specified CONF file from the CONF File drop-down list. If no CONF file is available, you can click **Add Authentication File** to upload a CONF file.

Parameter	Description
principal	The Kerberos principal. Specify this parameter in the format of Principal name/Instance name@Domain name, such as ****/hadoopclient@**.***.

- The following table describes the parameters for adding a Hive data source in **Built-in Mode of CDH Cluster**.



Parameter	Description
Data source type	The type of the data source. Set this parameter to **Built-in Mode of CDH Cluster**.
Data Source Name	The name of the data source. The name can contain letters, digits, and underscores (_) and must start with a letter.
Data source description	The description of the data source. The description can be up to 80 characters in length.

Parameter	Description
Environment	The environment in which the data source is used. Valid values: **Development** and **Production**. **Note** This parameter is displayed only when the workspace is in standard mode.
Select CDH Cluster	The CDH cluster that you want to use.
Special Authentication Method	Specifies whether to enable identity authentication. Default value: **None**. You can alternatively set this parameter to **Kerberos Authentication**. For more information about Kerberos authentication, see Configure Kerberos authentication.
Keytab File	If you set the **Special Authentication Method** parameter to **Kerberos Authentication**, you must select the specified keytab file from the Keytab File drop-down list. If no keytab file is available, you can click **Add Authentication File** to upload a keytab file.
CONF File	If you set the **Special Authentication Method** parameter to **Kerberos Authentication**, you must select the specified CONF file from the CONF File drop-down list. If no CONF file is available, you can click **Add Authentication File** to upload a CONF file.
principal	The Kerberos principal. Specify this parameter in the format of Principal name/Instance name@Domain name, such as ****/hadoopclient@**.***.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

3.5.24. Configure an OSS connection

Alibaba Cloud OSS is a secure and reliable service that allows you to store large amounts of objects.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, choose **Data Source > Data Sources** in the left-side navigation pane. The **Data Source** page under **Workspace Management** appears.

2. Click **Add data source** in the upper-right corner.
3. In the **Add Connection** dialog box, click **OSS** in the Semi-Structured Storage Systems section.
4. In the **Add OSS Connection** dialog box, set the parameters as required.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
Endpoint	The OSS endpoint, in the format of `http://oss.aliyuncs.com`. The OSS endpoint varies with the region. **Note** If you add the bucket name before the domain name, for example, `http://xxx.oss.aliyuncs.com`, the connection can pass the connectivity test but data synchronization will fail.
Bucket	The name of the OSS bucket. A bucket is a storage space that serves as a container for storing objects. You can create one or more buckets and add one or more objects to each bucket. DataWorks can search for objects only in the bucket specified here during data synchronization.
AccessKey ID	The AccessKey ID for connecting to the OSS bucket.
AceessKey Secret	The AccessKey secret for connecting to the OSS bucket.

Notice When data in OSS is stored as CSV files, they must comply with the standard CSV format. For example, if the data in a column of a CSV file contains a double quotation mark ("), you must replace the double quotation mark with a pair of double quotation marks (""). Otherwise, the data in the CVS file may be incorrectly parsed.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

3.5.25. Add an HDFS data source

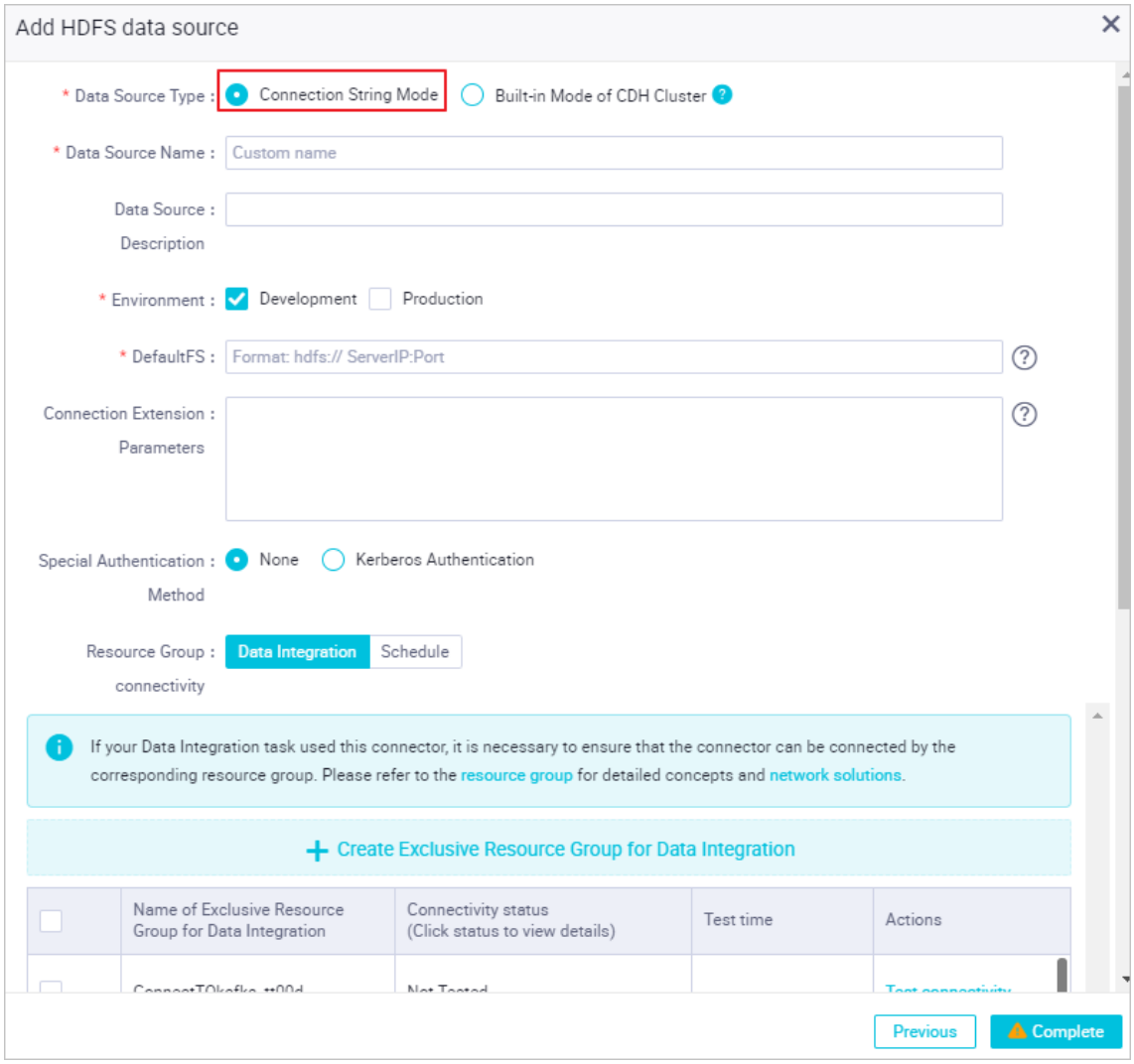
This topic describes how to configure a Hadoop Distributed File System (HDFS) data source. A HDFS data source allows you to read data from and write data to HDFS. You can configure synchronization nodes by using the code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, choose **Data Source > Data Sources** in the left-side navigation pane. The **Data Source** page under **Workspace Management** appears.
2. Click **Add data source** in the upper-right corner.
3. In the **Add data source** dialog box, click **HDFS** in the Semi-Structured Storage Systems section.
4. In the **Add HDFS data source** dialog box, set the parameters as required.

   You can use one of the following modes to add an HDFS data source: **Connection string mode** and **Built-in Mode of CDH Cluster**.

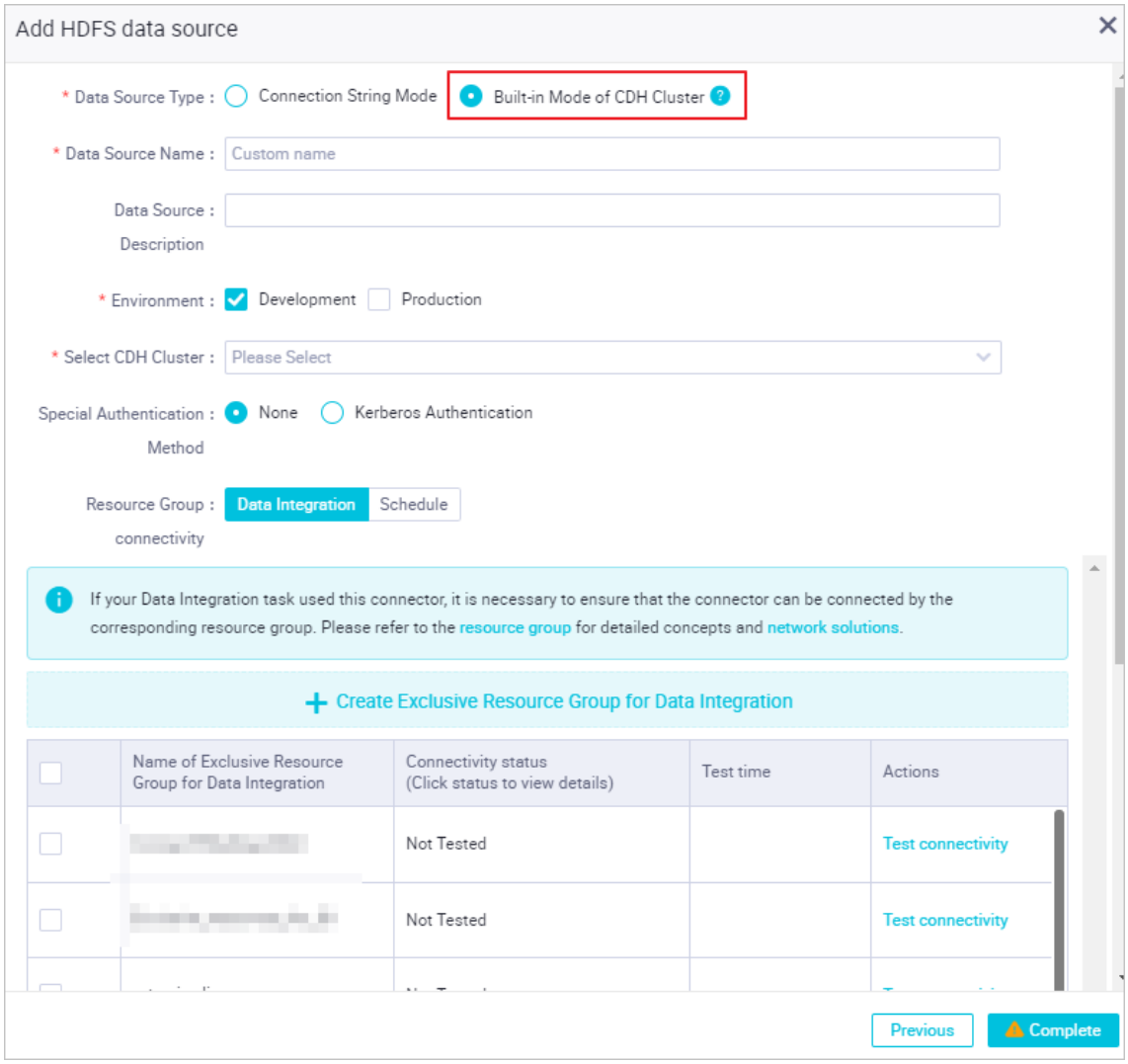
   - The following table describes the parameters for adding an HDFS data source in **Connection string mode**.



Parameter	Description
Data source type	The mode in which the data source is added. Set this parameter to **Connection string mode**.
Data Source Name	The name of the data source. The name can contain letters, digits, and underscores (_) and must start with a letter.
Data source description	The description of the data source. The description can be up to 80 characters in length.
Environment	The environment in which the data source is used. Valid values: **Development** and **Production**. **Note** This parameter is displayed only when the workspace is in standard mode.
defaultFS	The address of the NameNode in HDFS. Specify this parameter in the format of `hdfs://ServerIP:Port`.

Parameter	Description
Extended parameters	The advanced parameters in hadoopConfig for HDFS Reader and HDFS Writer. You can configure the advanced parameters of Hadoop, such as those related to high availability (HA).
Special Authentication Method	Specifies whether to enable identity authentication. Default value: **None**. You can alternatively set this parameter to **Kerberos Authentication**. For more information about Kerberos authentication, see Configure Kerberos authentication.
Keytab File	If you set the **Special Authentication Method** parameter to **Kerberos Authentication**, you must select the specified keytab file from the Keytab File drop-down list. If no keytab file is available, you can click **Add Authentication File** to upload a keytab file.
CONF File	If you set the **Special Authentication Method** parameter to **Kerberos Authentication**, you must select the specified CONF file from the CONF File drop-down list. If no CONF file is available, you can click **Add Authentication File** to upload a CONF file.
principal	The Kerberos principal. Specify this parameter in the format of `Principal name/Instance name@Domain name`, such as `***/hadoopclient@**.***`.

- The following table describes the parameters for adding an HDFS data source in **Built-in Mode of CDH Cluster**.



Parameter	Description
Data source type	The type of the data source. Set this parameter to **Built-in Mode of CDH Cluster**.
Data Source Name	The name of the data source. The name can contain letters, digits, and underscores (_) and must start with a letter.
Data source description	The description of the data source. The description can be up to 80 characters in length.
Environment	The environment in which the data source is used. Valid values: **Development** and **Production**. **Note** This parameter is displayed only when the workspace is in standard mode.
Select CDH Cluster	The CDH cluster that you want to use.

Parameter	Description
Special Authentication Method	Specifies whether to enable identity authentication. Default value: **None**. You can alternatively set this parameter to **Kerberos Authentication**. For more information about Kerberos authentication, see Configure Kerberos authentication.
Keytab File	If you set the **Special Authentication Method** parameter to **Kerberos Authentication**, you must select the specified keytab file from the Keytab File drop-down list. If no keytab file is available, you can click **Add Authentication File** to upload a keytab file.
CONF File	If you set the **Special Authentication Method** parameter to **Kerberos Authentication**, you must select the specified CONF file from the CONF File drop-down list. If no CONF file is available, you can click **Add Authentication File** to upload a CONF file.
principal	The Kerberos principal. Specify this parameter in the format of `Principal name/Instance name@Domain name`, such as `****/hadoopclient@**.***`.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

3.5.26. Configure an FTP connection

An FTP connection allows you to read data from and write data to FTP by using FTP Reader and Writer. You can configure sync nodes for FTP by using the codeless UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, choose **Data Source > Data Sources** in the left-side navigation pane. The **Data Source** page under **Workspace Management** appears.
2. Click **Add data source** in the upper-right corner.
3. In the **Add Connection** dialog box, click **FTP** in the Semi-Structured Storage Systems section.
4. In the **Add FTP Connection** dialog box, set the parameters as required.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
Portocol	The protocol used by the FTP server. Only FTP and SFTP are supported.
Host	The address of the FTP server.
Port	The port of the FTP server. The default port is 21 for FTP and 22 for SFTP.
Username	The username that you can use to connect to the FTP server.
Password	The password that you can use to connect to the FTP server.
Enable reverse VPC access	Specifies whether to enable reverse VPC access. Select the Enable check box if you cannot directly access the data store on an ECS instance but can access it by using a VPC.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

3.5.27. Configure a MongoDB connection

MongoDB is a document-oriented database that is second only to Oracle and MySQL. A MongoDB connection allows you to read data from and write data to MongoDB by using MongoDB Reader and Writer. You can configure sync nodes for MongoDB by using the code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, choose **Data Source > Data Sources** in the left-side navigation pane. The **Data Source** page under **Workspace Management** appears.
2. Click **Add data source** in the upper-right corner.
3. In the **Add Connection** dialog box, click **MongoDB** in the NoSQL section.

4. In the **Add MongoDB Connection** dialog box, set the parameters as required.

   You can set the Connect To parameter to **ApsaraDB for RDS** or **Connection Mode** for a MongoDB connection.

   - **ApsaraDB for RDS**: Generally, the classic network is used to access the target ApsaraDB for MongoDB instance in this mode. You can access the ApsaraDB for MongoDB instance in the same region over the classic network. However, the access to the ApsaraDB for MongoDB instance from a different region over the classic network is not guaranteed to be successful.

Parameter	Description
Connect To	The connection type. In this example, set the value to **ApsaraDB for RDS**. **Note** If you have not assigned the default role to Data Integration, log on to the Resource Access Management (RAM) console with your Apsara Stack tenant account and perform authorization. Then, refresh this configuration page.
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
Region	The region where the ApsaraDB for MongoDB instance resides.
Instance ID	The ID of the ApsaraDB for MongoDB instance. You can view the ID in the ApsaraDB for MongoDB console.
Database Name	The name of the database that you created in the ApsaraDB for MongoDB console. You can also specify the database username and password in the console.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.

   - **Connection Mode**: Generally, the Internet is used to access the target database in this mode, which may cost you fees.

Parameter	Description

Parameter	Description
Connect To	The type of the connection. In this example, set the value to **Connection Mode**.
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
Address	The endpoint in the `host:port` format. To add an endpoint, click **Add Address** and specify the endpoint to add. To add more endpoints, repeat the preceding action. **Note** You must add either public endpoints or internal endpoints. Do not mix public endpoints with internal endpoints.
Database Name	The name of the database.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

3.5.28. Configure a Memcache connection

A Memcache connection allows you to write data to ApsaraDB for Memcache by using Memcache Writer. You can configure sync nodes for ApsaraDB for Memcache by using the code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.

iii. On the page that appears, choose **Data Source > Data Sources** in the left-side navigation pane. The **Data Source** page under **Workspace Management** appears.

2. Click **Add data source** in the upper-right corner.
3. In the **Add Connection** dialog box, click **Memcache(OCS)** in the NoSQL section.
4. In the **Add Memcache(OCS) Connection** dialog box, set the parameters as required.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
Proxy Host	The IP address of the host or Memcache proxy. You can view the IP address on the basic information page of the ApsaraDB for Memcache console.
Port	The port for connecting to the ApsaraDB for Memcache instance. Default value: 11211.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

3.5.29. Configure a Redis connection

A Redis connection allows you to read data from and write data to Redis by using Redis Reader and Writer. You can configure sync nodes for Redis by using the code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, choose **Data Source > Data Sources** in the left-side navigation pane. The **Data Source** page under **Workspace Management** appears.
2. Click **Add data source** in the upper-right corner.

3. In the **Add Connection** dialog box, click **Redis** in the NoSQL section.
4. In the **Add Redis Connection** dialog box, set the parameters as required.

   You can set the Connect To parameter to **ApsaraDB for RDS** or **Connection Mode** for a Redis connection.

   - The following table describes the parameters that appear after you set the **Connect To** parameter to **ApsaraDB for RDS**.

Parameter	Description
Connect To	The type of the connection. In this example, set the value to **ApsaraDB for RDS**.
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
Region	The region where the ApsaraDB for Redis instance resides.
Redis Instance ID	The ID of the ApsaraDB for Redis instance. You can view the ID in the ApsaraDB for Redis console.
Redis Password	The password that you can use to connect to the ApsaraDB for Redis instance. Leave it blank if no password is required.

   - The following table describes the parameters that appear after you set the **Connect To** parameter to **Connection Mode**.

Parameter	Description
Connect To	The type of the connection. In this example, set the value to **Connection Mode**.
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.

Parameter	Description
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
Server Address	The server address in the `host:port` format.
Add Server Address	Click **Add Server Address** to add a server address in the format of `host:port`.
Redis Password	The password that you can use to connect to Redis.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

3.5.30. Configure a Tablestore connection

Tablestore is a NoSQL database service built on Apsara distributed operating system. It allows you to store and access large amounts of structured data in real time.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, choose **Data Source > Data Sources** in the left-side navigation pane. The **Data Source** page under **Workspace Management** appears.
2. Click **Add data source** in the upper-right corner.
3. In the **Add Connection** dialog box, click **OTS** in the NoSQL section.
4. In the **Add OTS Connection** dialog box, set the parameters as required.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.

Parameter	Description
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
Endpoint	The endpoint of the Tablestore service.
Table Store Instance ID	The name of the Tablestore instance.
AccessKey ID	The AccessKey ID for connecting to the Tablestore instance. You can view the AccessKey ID on the **User Info** page.
AccessKey Secret	The AccessKey secret for connecting to the Tablestore instance.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

3.5.31. Configure an Elasticsearch connection

An Elasticsearch connection allows you to read data from and write data to Elasticsearch by using Elasticsearch Reader and Writer. You can configure sync nodes for Elasticsearch by using the code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, choose **Data Source > Data Sources** in the left-side navigation pane. The **Data Source** page under **Workspace Management** appears.
2. On the **Data Source** page, click **Add data source** in the upper-right corner.
3. In the **Add Connection** dialog box, click **ElasticSearch** in the NoSQL section.
4. In the **Add ElasticSearch Connection** dialog box, set the parameters as required.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.

Parameter	Description
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
Endpoint	The endpoint of Elasticsearch, in the format of `http://esxxxx.elasticsearch.aliyuncs.com:9200`.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

3.5.32. Configure a LogHub connection

A LogHub connection allows you to read data from and write data to LogHub by using LogHub Reader and Writer. You can configure sync nodes for LogHub by using the codeless UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, choose **Data Source > Data Sources** in the left-side navigation pane. The **Data Source** page under **Workspace Management** appears.
2. Click **Add data source** in the upper-right corner.
3. In the **Add Connection** dialog box, click **LogHub** in the Message Queue section.
4. In the **Add LogHub Connection** dialog box, set the parameters as required.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.

Parameter	Description
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
LogHub Endpoint	The LogHub endpoint, in the format of `http://cn-shanghai.log.aliyun.com`.
Project	The name of the LogHub project.
AccessKey ID	The AccessKey ID for connecting to the LogHub project. You can view the AccessKey ID on the **User Info** page.
AceessKey Secret	The AccessKey secret for connecting to the LogHub project.

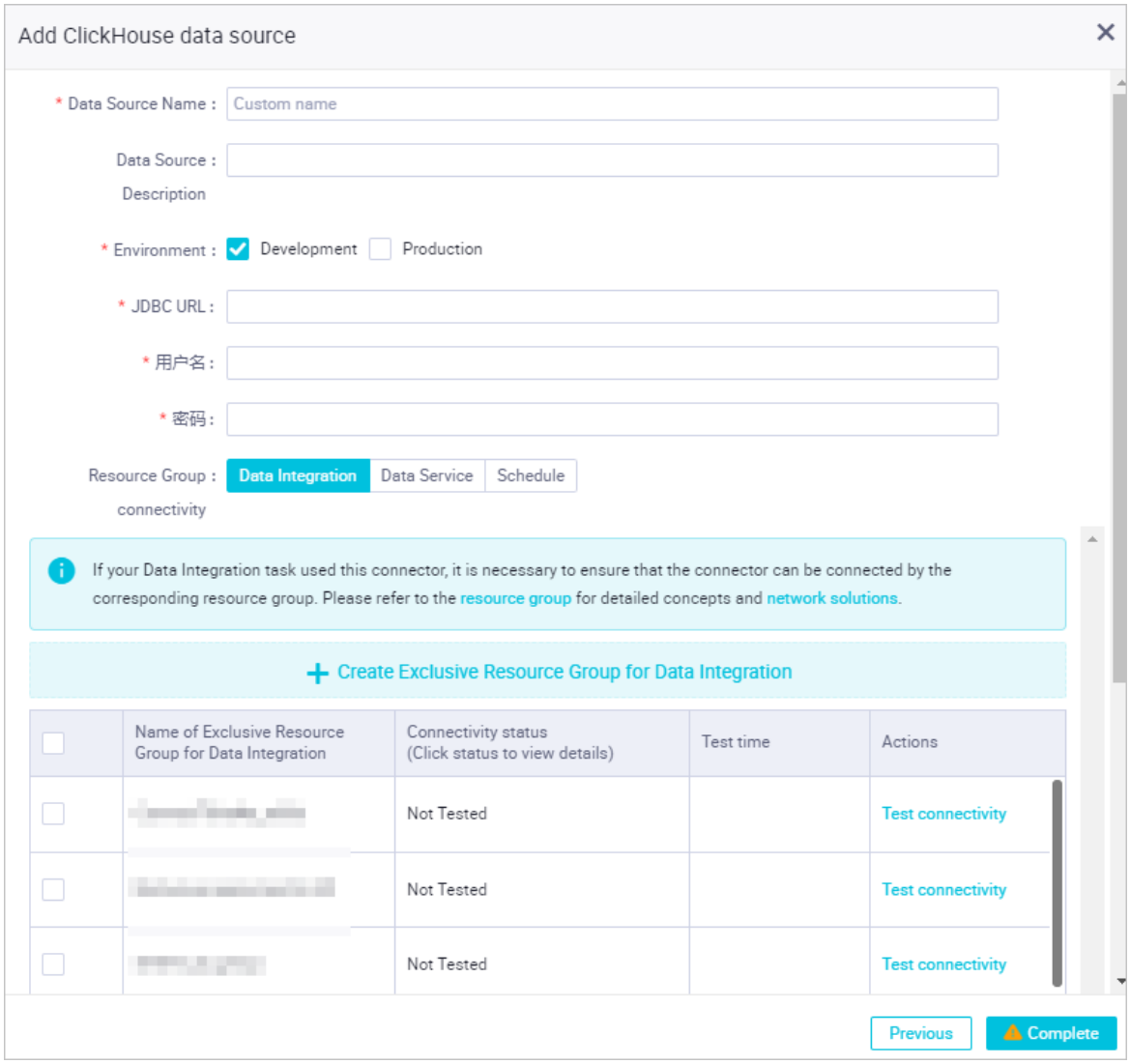
5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

3.5.33. Add a ClickHouse data source

This topic describes how to configure a ClickHouse data source. A ClickHouse data source allows you to read data from and write data to Hive. You can configure synchronization nodes by using the codeless user interface (UI) or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the ☰ icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, choose **Data Source > Data Sources** in the left-side navigation pane. The **Data Source** page under **Workspace Management** appears.
2. On the **Data Source** page, click **Add data source** in the upper-right corner.
3. In the **Add data source** dialog box, click **ClickHouse** in the Big Data Storage section.
4. In the **Add ClickHouse data source** dialog box, set the parameters as required.



Parameter	Description
Data Source Name	The name of the data source. The name can contain letters, digits, and underscores (_) and must start with a letter.
Data source description	The description of the data source. The description can be up to 80 characters in length.
JDBC URL	The Java Database Connectivity (JDBC) URL of the database. Specify a value for this parameter in the format of `jdbc:mysql://ServerIP:Port/Database`.
User name	The username that is used to connect to the database.
Password	The password that is used to connect to the database.

5. Click **Test connectivity**.
6. After the data source passes the connectivity test, click **Complete**.

3.6. Configure data synchronization tasks

3.6.1. Configure a sync node by using the codeless UI

This topic describes how to configure a sync node by using the codeless user interface (UI).

To configure a sync node, follow these steps:

1. Add connections.
2. Create a sync node.
3. Select a source connection.
4. Select a destination connection.
5. Map the fields in the source and destination tables.
6. Configure the channel, such as the maximum transmission rate and dirty data check rules.
7. Configure the node properties.

> **Note** The following sections describe the overall procedure. You can click the links in each step to read relevant instructions and then return to the current page to proceed with subsequent steps.

Add connections

Data synchronization is supported between various homogenous and heterogeneous connections. Before you configure a sync node, add required connections in Data Integration. Added connections are listed as options when you configure a sync node. For more information about connection types supported by Data Integration, see Supported data sources.

You can add connections of supported types to Data Integration. For more information about how to add a connection, see Data sources.

> **Note**
>
> - Data Integration does not support connectivity testing for some connection types. For more information, see Test data store connectivity.
> - Some connections are hosted on the premises. They do not have public IP addresses or network connections cannot be directly established. Such connections will fail the connectivity test. Data Integration allows you to add a custom resource group to resolve these issues. However, if you create sync nodes for such connections, you can only use the code editor. This is because you cannot obtain information such as table schema on the codeless UI if the network connection is unavailable.

Create a sync node

> **Note** This topic describes how to create and configure a sync node by using the codeless UI. Do not switch to the code editor.

1. Log on to the DataWorks console.
2. On the **Data Analytics** tab, move the pointer over the **Create** icon and select **Workflow**.
3. In the **Create Workflow** dialog box that appears, set **Workflow Name** and **Description**. Then, click **Create**.
4. In the left-side navigation pane, click the created workflow. Then, right-click **Data Integration** and choose **Create Data Integration Node > Sync**. In the Create Node dialog box that appears, set **Node Name**.
5. Click **Commit**.

Select a source connection

After the sync node is created, configure the source connection and source table.

> **Note**
> - For more information about how to configure the source connection, see Configure the reader.
> - Incremental data synchronization is required when you configure the source connection for some sync nodes. In this case, you can use the parameter configuration feature of DataWorks to obtain the date and time required by incremental data synchronization.

Select a destination connection

After the source connection is configured, configure the destination connection and destination table.

> **Note**
> - For more information about how to configure the destination connection, see Configure the writer.
> - You can select the writing method for most nodes. For example, the writing method can be overwriting or appending. Supported writing methods vary with the connection type.

Map the fields in the source and destination tables

After the source and destination connections are configured, specify the mapping between the fields in the source and destination tables. You can click **Map Fields with the Same Name**, **Map Fields in the Same Line**, **Delete All Mappings**, and **Auto Layout**.

Button or icon	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.

Button or icon	Description
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	Click Auto Layout. The fields are automatically sorted based on specified rules.
Change Fields	Click the Change Fields icon. In the Change Fields dialog box that appears, you can manually edit fields in the source table. Each field occupies a row. The first and the last blank rows are included, whereas other blank rows are ignored.
Add	• Click Add to add a field. You can enter constants. Each constant must be enclosed in single quotation marks (' '), such as 'abc' and '123'. • You can use scheduling parameters, such as ${bizdate}. • You can enter functions supported by relational databases, such as now() and count(1). • Fields that cannot be parsed are indicated by Unidentified.

Note Make sure that the data type of a source field is the same as or compatible with that of the mapped destination field.

Configure channel control policies

When the preceding steps are completed, configure the channel control policies of the corresponding sync node.

Parameter	Description
Expected Concurrency	The maximum number of concurrent threads to read data from or write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure the node properties

This section describes how to use scheduling parameters for data filtering.

On the sync node configuration tab, click the **Properties** tab in the right-side navigation pane.

You can declare the scheduling parameters by using ${Variable name}. After a variable is declared, enter the initial value of the variable in the Arguments field. In this example, the initial value of the variable is identified by $[]. The content can be a time expression or a constant.

For example, if you write ${today} in the code and enter today=$[yyyymmdd] in the Arguments field, the value of the time variable is the current date. For more information about how to add and subtract the date, see Parameter configuration.

On the Properties tab, you can configure the properties of the sync node, such as the recurrence, scheduled time, and dependencies. Sync nodes have no ancestor nodes because their corresponding jobs are run before extract, transform, and load (ETL) jobs. We recommend that you specify the root node as their parent node.

Use custom scheduling parameters

To use custom scheduling parameters for the sync node, declare the following parameters in the code:

- bizdate: the timestamp of data to be used by the node. The value is one day before the running date of the node.
- cyctime: the time when the node is run, in the format of yyyymmddhhmiss.
- DataWorks provides the bizdate and cyctime parameters as default system parameters.

After the sync node is configured, save and commit the node.

3.6.2. Configure a sync node by using the code editor

This topic describes how to configure a sync node by using the code editor.

To configure a sync node, follow these steps:

1. Add connections.
2. Create a sync node.
3. Apply a template.
4. Configure the reader.
5. Configure the writer.
6. Map the fields in the source and destination tables.
7. Configure the channel, such as the maximum transmission rate and dirty data check rules.
8. Configure the node properties.

Add connections

Data synchronization is supported between various homogenous and heterogeneous connections. Before you configure a sync node, add required connections in Data Integration. Added connections are listed as options when you configure a sync node. For more information about connection types supported by Data Integration, see Supported data sources.

You can add connections of supported types to Data Integration. For more information about how to add a connection, see Data sources.

> **Note** Some connections are hosted on the premises. They do not have public IP addresses or network connections cannot be directly established. Such connections will fail the connectivity test. Data Integration allows you to add a custom resource group to resolve these issues. However, if you create sync nodes for such connections, you can only use the code editor. This is because you cannot obtain information such as table schema on the codeless user interface (UI) if the network connection is unavailable.

Create a sync node

> **Note** This topic describes how to create a sync node by using the codeless UI and configure the sync node by using the code editor.

1. Log on to the DataWorks console.
2. On the **Data Analytics** tab, move the pointer over the **Create** icon and select **Workflow**.
3. In the **Create Workflow** dialog box that appears, set **Workflow Name** and **Description**. Then, click **Create**.
4. In the left-side navigation pane, click the created workflow. Then, right-click **Data Integration** and choose **Create Data Integration Node > Sync**. In the Create Node dialog box that appears, set **Node Name**.
5. Click **Commit**.

Apply a template

1. After the sync node is created, the node configuration tab appears. Click the **Switch to Code Editor** icon in the toolbar.
2. In the **Confirm** dialog box that appears, click **OK** to switch to the code editor.

   > **Note** The code editor supports more features than the codeless UI. For example, you can configure sync nodes in the code editor even when the connectivity test fails.

3. Click the **Apply Template** icon in the toolbar.
4. In the **Apply Template** dialog box that appears, set **Source Connection Type**, **Connection**, **Target Connection Type**, and **Connection**.
5. Click **OK**.

Configure the reader

After the template is applied, the basic settings of the reader are configured. You can configure the source connection and source table as needed.

```
{"type": "job",
    "version": "2.0",
    "steps": [
        {
            "stepType": "mysql", // The reader type.
            "parameter": {
                "datasource": "MySQL", // The connection name.
                "column": [ // The columns to be synchronized.
                    "id",
                    "value",
                    "table"
                ],
                "socketTimeout": 3600000, // The timeout period for reading data from and w
riting data to a socket, in milliseconds.
                "connection": [
                    {
                        "datasource": "MySQL", // The connection name.
                        "table": [
                            "`case`" // The name of the table to be synchronized.
                        ]
                    }
                ],
                "where": "", // The WHERE clause.
                "splitPk": "", // The shard key.
                "encoding": "UTF-8" // The encoding format.
            },
            "name": "Reader",
            "category": "reader" // Indicates that these settings are related to the reader
.
        },
```

The parameters are described as follows:

- type: the type of the sync node. You must set the value to job.
- version: the version number of the sync node. You can set the value to 1.0 or 2.0.

> **Note**
>
> - For more information about how to configure the source connection in the code editor, see Configure the reader.
> - Incremental data synchronization is required when you configure the source connection for some sync nodes. In this case, you can use the parameter configuration feature of DataWorks to obtain the date and time required by incremental data synchronization.

Configure the writer

After the reader is configured, you can configure the destination connection and destination table as needed.

```
{
  "stepType": "odps", // The writer type.
  "parameter": {
      "partition": "", // The partitions that the reader reads.
      "truncate": true, // Specifies whether to clear up previous data and import new data
when a write operation is performed again after failure. Set the value to true to guarantee
the idempotence of write operations.
      "compress": false, // Specifies whether to enable compression.
      "datasource": "odps_first", // The connection name.
      "column": [ // The columns to be synchronized.
          "*"
       ],
       "emptyAsNull": false,
       "table": ""
     },
     "name": "Writer",
     "category": "writer" // Indicates that these settings are related to the writer.
   }
 ],
```

> **Note**
>
> - For more information about how to configure the destination connection in the code editor, see Configure the writer.
> - You can select the writing method for most nodes. For example, the writing method can be overwriting or appending. Supported writing methods vary with the connection type.

Map the fields in the source and destination tables

The code editor only supports mapping of fields in the same row. Note that the data types of the fields must match.

> **Note** Make sure that the data type of a source field is the same as or compatible with that of the mapped destination field.

Configure channel control policies

When the preceding steps are completed, configure the channel control policies of the corresponding sync node. The setting parameter specifies the node efficiency, including the settings on the DUM number, thread concurrency, bandwidth throttling, dirty data policy, and resource group.

```
"setting": {
        "errorLimit": {
            "record": "1024" // The maximum number of dirty data records allowed.
        },
        "speed": {
            "throttle": false, // Specifies whether to enable bandwidth throttling.
            "concurrent": 1, // The maximum number of concurrent threads.
        }
    },
```

Setting	Description
Expected concurrency	The maximum number of concurrent threads to read data from or write data to data storage within the sync node. You can configure the concurrency for a node in the code editor.
Bandwidth throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty data records allowed	The maximum number of dirty data records allowed.
Resource group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure the node properties

This section describes how to use scheduling parameters for data filtering.

On the sync node configuration tab, click the **Properties** tab in the right-side navigation pane.

On the Properties tab, you can configure the properties of the sync node, such as the recurrence, scheduled time, and dependencies. Sync nodes have no ancestor nodes because their corresponding jobs are run before extract, transform, and load (ETL) jobs. We recommend that you specify the root node as their parent node.

After the sync node is configured, save and commit the node.

3.6.3. Configure the reader

3.6.3.1. Configure DRDS Reader

Distributed Relational Database Service (DRDS) Reader allows you to read data from DRDS. DRDS Reader connects to a remote DRDS database and runs a SELECT statement to select and read data from the database.

Currently, DRDS Reader only supports MySQL engines. DRDS is a distributed MySQL database service that complies with MySQL protocols in most cases.

Specifically, DRDS Reader connects to a remote DRDS database through Java Database Connectivity (JDBC), generates a SELECT statement based on your configurations, and then sends the statement to the database. The DRDS database runs the statement and returns the result. Then, DRDS Reader assembles the returned data to abstract datasets in custom data types supported by Data Integration, and passes the datasets to a writer.

DRDS Reader generates the SELECT statement based on the table, column, and where parameters that you have configured, and sends the generated SELECT statement to the DRDS database. DRDS does not support all MySQL specifications, such as JOIN statements.

DRDS Reader supports most DRDS data types. Make sure that your data types are supported.

The following table lists the data types supported by DRDS Reader.

Category	DRDS data type
Integer	INT, TINYINT, SMALLINT, MEDIUMINT, and BIGINT
Floating point	FLOAT, DOUBLE, and DECIMAL
String	VARCHAR, CHAR, TINYTEXT, TEXT, MEDIUMTEXT, and LONGTEXT
Date and time	DATE, DATETIME, TIMESTAMP, TIME, and YEAR
Boolean	BIT and BOOLEAN
Binary	TINYBLOB, MEDIUMBLOB, BLOB, LONGBLOB, and VARBINARY

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None
table	The name of the table to be synchronized.	Yes	None
column	The columns to be synchronized from the source table. The columns are described in a JSON array. The default value is [*], which indicates all columns. • Column pruning is supported. You can select and export specific columns. • Change of the column order is supported. You can export the columns in an order different from that specified in the schema of the table. • Constants are supported. The column names must be arranged in compliance with the SQL syntax supported by MySQL, for example, `["id", "`table`", "1", "'bazhen.csy"', "null", "to_char(a + 1)", "2.3", "true"]`. ◦ id: a column name. ◦ table: the name of a column that contains reserved keywords. ◦ 1: an integer constant. ◦ bazhen.csy: a string constant. ◦ null: a null pointer. ◦ to_char(a + 1): a function expression. ◦ 2.3: a floating-point constant. ◦ true: a Boolean value. • The column parameter must explicitly specify a set of columns to be synchronized. The parameter cannot be left empty.	Yes	None

Parameter	Description	Required	Default value
where	The WHERE clause. DRDS Reader generates a SELECT statement based on the table, column, and where parameters that you have configured, and uses the generated SELECT statement to select and read data. For example, set this parameter to `STRTODATE('${bdp.system.bizdate}', '%Y%m%d') <= today AND today < DATEADD(STRTODATE('${bdp.system.bizdate}', '%Y%m%d'), interval 1 day)`. • You can use the WHERE clause to synchronize incremental data. • If you do not specify the where parameter or leave it empty, all data is synchronized.	No	None

Configure DRDS Reader by using the codeless UI

1. Configure the connections.

   Configure the source and destination connections for the sync node.

Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and enter the name of a connection that has been configured in DataWorks.
Table	The table parameter in the preceding parameter description.
Filter	The filter condition for the data to be synchronized. Currently, filtering based on the limit keyword is not supported. The SQL syntax is determined by the selected connection.
Shard Key	The shard key. You can use a column in the source table as the shard key. We recommend that you use the primary key or an indexed column. Only integer fields are supported. If data sharding is performed based on the configured shard key, data can be read concurrently to improve data synchronization efficiency. **Note** The Shard Key parameter is displayed only when you configure the source connection for a sync node.

2. Configure field mapping, that is, the column parameter in the preceding parameter description.

   Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right. You can click **Add** to add a field, or move the pointer over a field and click the **Delete** icon to delete the field.

Parameter	Description

Parameter	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	Click Auto Layout. The fields are automatically sorted based on specified rules.
Change Fields	Click the Change Fields icon. In the Change Fields dialog box that appears, you can manually edit fields in the source table. Each field occupies a row. The first and the last blank rows are included, whereas other blank rows are ignored.
Add	Click Add to add a field. The rules for adding fields are described as follows: ◦ You can enter constants. Each constant must be enclosed in single quotation marks (' '), such as 'abc' and '123'. ◦ You can use scheduling parameters, such as ${bizdate}. ◦ You can enter functions supported by relational databases, such as now() and count(1). ◦ Fields that cannot be parsed are indicated by Unidentified.

3. Configure channel control policies.

Parameter	Description
Expected Concurrency	The maximum number of concurrent threads to read data from or write data to data storage within the sync node. You can configure the concurrency for a node on the codeless user interface (UI).
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure DRDS Reader by using the code editor

In the following code, a node is configured to read data from a DRDS database.

```
{
    "type":"job",
    "version":"2.0", // The version number.
    "steps":[
        {
            "stepType":"drds", // The reader type.
            "parameter":{
                "datasource":"", // The connection name.
                "column":[ // The columns to be synchronized.
                    "id",
                    "name"
                ],
                "where":"", // The WHERE clause.
                "table":"", // The name of the table to be synchronized.
                "splitPk": "" // The shard key.
            },
            "name":"Reader",
            "category":"reader"
        },
        {// The following template is used to configure Stream Writer. For more information
about how to configure other writers, see the corresponding topic.
            "stepType":"stream", // The writer type.
            "parameter":{},
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0" // The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false, // Specifies whether to enable bandwidth throttling.
            "concurrent":1, // The maximum number of concurrent threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}:"Writer"
            }
        ]
    }
}
```

Additional instructions

- Consistency

As a distributed database service, DRDS cannot provide a consistent view of multiple tables in multiple databases. Different from MySQL where data is synchronized in a single table of a single database, DRDS Reader cannot extract the snapshot of database and table shards at the same time slice. That is, DRDS Reader extracts different snapshots from different shards. As a result, this cannot guarantee strong consistency for data queries.

- Character encoding

  DRDS supports flexible encoding configurations. You can specify the encoding format for an instance, a field, a table, and a database. The configurations for the field, table, database, and instance are prioritized in descending order. We recommend that you use UTF-8 for a database.

  DRDS Reader uses JDBC, which can automatically convert the encoding of characters. Therefore, you do not need to specify the encoding format.

  If you specify the encoding format for a DRDS database but data is written to the DRDS database in a different encoding format, DRDS Reader cannot recognize this inconsistency and may export garbled characters.

- Incremental data synchronization

  DRDS Reader connects to a database through JDBC and uses a SELECT statement with a WHERE clause to read incremental data in the following ways:

  - For data in batches, incremental add, update, and delete operations (including logically delete operations) are distinguished by timestamps. Specify the WHERE clause based on the timestamp. The timestamp must be later than the latest timestamp in the last synchronization.
  - For streaming data, specify the WHERE clause based on the data record ID. The data record ID must be larger than the maximum ID involved in the last synchronization.

  If incremental data cannot be distinguished, DRDS Reader cannot perform incremental synchronization but can perform full synchronization only.

- Syntax validation

  DRDS Reader allows you to specify custom SELECT statements by using the querySql parameter but does not verify the syntax of the custom SELECT statements.

3.6.3.2. Configure HBase Reader

HBase Reader allows you to read data from HBase. HBase Reader connects to a remote HBase database through a Java client of HBase. Then, HBase Reader scans and reads data based on the specified rowkey range, assembles the data to abstract datasets in custom data types supported by Data Integration, and then passes the datasets to a writer.

Data types

The following table lists the data types supported by HBase Reader.

Category	Data Integration data type	HBase data type
Integer	LONG	Short, Int, and Long
Floating point	DOUBLE	Float and Double
String	STRING	Binary_String and String

Category	Data Integration data type	HBase data type
Date and time	DATE	Date
Byte	BYTES	Bytes
Boolean	BOOLEAN	Boolean

Parameters

Parameter	Description	Required	Default value
haveKerberos	Specifies whether Kerberos authentication is required. A value of true indicates that Kerberos authentication is required. **Note** • If the value is true, the following five Kerberos-related parameters must be specified: ◦ kerberosKeytabFilePath ◦ kerberosPrincipal ◦ hbaseMasterKerberosPrincipal ◦ hbaseRegionserverKerberosPrincipal ◦ hbaseRpcProtection • If the value is false, Kerberos authentication is not required and you do not need to specify the preceding parameters.	No	*false*
hbaseConfig	The properties of the HBase cluster, in JSON format. The hbase.zookeeper.quorum parameter is required. It specifies the ZooKeeper ensemble servers. You can also configure other properties, such as those related to the cache and batch for scan operations.	Yes	None
mode	The mode in which data is read from the HBase connection. Valid values: normal and multiVersionFixedColumn.	Yes	None
table	The name of the HBase table from which data is read. The name is case-sensitive.	Yes	None
encoding	The encoding format, by using which binary data stored in byte[] format is converted into strings. Currently, UTF-8 and GBK are supported.	No	UTF-8

Parameter	Description	Required	Default value
column	The HBase columns from which data is read. • In normal mode: The name parameter specifies the name of the column in the HBase table. The format must be columnFamily:columnName except for the rowkey. The type parameter specifies the source data type. The format parameter specifies the date format. The value parameter specifies the column value if the column is a constant column. Example: `"column": [ { "name": "rowkey", "type": "string" }, { "value": "test", "type": "string" } ]` For the column parameter, you must specify the type parameter and specify one of the name and value parameters. • In multiVersionFixedColumn mode: The name parameter specifies the name of the column in the HBase table. The format must be columnFamily:columnName except for the rowkey. The type parameter specifies the source data type. The format parameter specifies the date format. You cannot create constant columns in multiVersionFixedColumn mode. Example: `"column": [ { "name": "rowkey", "type": "string" }, { "name": "info:age", "type": "string" } ]`	Yes	None

Parameter	Description	Required	Default value
maxVersion	The number of versions read by HBase Reader when multiple versions are available. Valid values: -1 and integers greater than 1. A value of -1 indicates that all versions are read.	Required in multiVersionFixedColumn mode	None
range	The rowkey range that HBase Reader reads. • startRowkey: the start rowkey. • endRowkey: the end rowkey. • isBinaryRowkey: the method used to convert the specified start and end rowkeys into the byte[] format. Default value: false. If the value is true, Bytes.toBytesBinary(rowkey) is used. If the value is false, Bytes.toBytes(rowkey) is used. Example: `"range": { "startRowkey": "aaa", "endRowkey": "ccc", "isBinaryRowkey":false }`	No	None
scanCacheSize	The number of rows read by an HBase client with each remote procedure call (RPC) connection.	No	*256*
scanBatchSize	The number of columns read by an HBase client with each RPC connection.	No	*100*

Configure HBase Reader by using the codeless UI

Currently, the codeless user interface (UI) is not supported for HBase Reader.

Configure HBase Reader by using the code editor

In the following code, a node is configured to read data from an HBase connection in normal mode.

```
{
    "type": "job",
    "version": "2.0", // The version number.
    "steps": [
        {
            "stepType": "hbase", // The reader type.
            "parameter": {
                "mode": "normal",
                "scanCacheSize": 256, // The number of rows read by an HBase client with ea
ch RPC connection.
                "scanBatchSize": 256, // The number of columns read by an HBase client with
each RPC connection.
                "hbaseVersion": "094x",
                "datasource": "demo_hbase", // The connection name.
                "column": [
```

```
                    {
                        "name": "info:idx",
                        "type": "long"
                    },
                    {
                        "name": "info:age",
                        "type": "string"
                    },
                    {
                        "name": "info:birthday",
                        "format": "yyyy-MM-dd",
                        "type": "date"
                    }
                ],
                "range": {
                    "startRowKey": "", // The start rowkey.
                    "endRowKey": "", // The end rowkey.
                    "isBinaryRowKey": false // The method used to convert the specified sta
rt and end rowkeys into the byte[] format. Default value: false. If the value is true, Byte
s.toBytesBinary(rowkey) is used. If the value is false, Bytes.toBytes(rowkey) is used.
                },
                "maxVersion": , // The number of versions read by HBase Reader when multipl
e versions are available. Valid values: -1 and integers greater than 1. A value of -1 indic
ates that all versions are read.
                "encoding": "UTF-8",
                "table": "test" // The name of the HBase table from which data is read. The
name is case-sensitive.
            },
            "name": "Reader",
            "category": "reader"
        },
            "tepType": "odps", // The writer type.
            "parameter": {},
            "name": "Writer",
            "category": "writer"
        }
    ],
    "setting": {
    },
    "order": {
        "hops": [
            {
                "from": "Reader",
                "to": "Writer"
            }
        ]
    }
}
```

In the following code, a node is configured to read data from an HBase connection in multiVersionFixedColumn mode.

```
{
```

```
    "type": "job",
    "version": "2.0", // The version number.
    "steps": [
        {
            "stepType": "hbase", // The reader type.
            "parameter": {
                    "table": "users", // The name of the HBase table from which data is rea
d. The name is case-sensitive.
                    "encoding": "utf-8", // The encoding format, by using which binary data
stored in byte[] format is converted into strings. Currently, UTF-8 and GBK are supported.
                    "mode": "multiVersionFixedColumn",
                    "maxVersion": "-1", // The number of versions read by HBase Reader when
multiple versions are available. Valid values: -1 and integers greater than 1. A value of -
1 indicates that all versions are read.
                    "column": [ // The HBase columns from which data is read. The name para
meter specifies the name of the column in the HBase table. The format must be columnFamily:
columnName except for the rowkey. The type parameter specifies the source data type. The fo
rmat parameter specifies the date format. You cannot create constant columns in multiVersio
nFixedColumn mode.
                        {
                            "name": "rowkey",
                            "type": "string"
                        },
                        {
                            "name": "info: age",
                            "type": "string"
                        },
                        {
                            "name": "info: birthday",
                            "type": "date",
                            "format": "yyyy-MM-dd"
                        }
                    ],
                    "range": { // The rowkey range that HBase Reader reads.
                        "startRowkey": "",
                        "endRowkey": ""
                    }
                }
            },
            "name": "Reader",
            "category": "reader"
        },
        {
            "stepType": "odps", // The writer type.
            "parameter": {},
            "name": "Writer",
            "category": "writer"
        }
    ],
    "setting": {
    },
    "order": {
        "hops": [
            {
                "from": "Reader"
```

```
                "from": "Reader",
                "to": "Writer"
            }
        ]
    }
}
```

3.6.3.3. Configure HDFS Reader

HDFS Reader allows you to read data stored in a Hadoop Distributed File System (HDFS). HDFS Reader connects to an HDFS, reads data from files in the HDFS, converts the data into a format that is readable by Data Integration, and then sends the converted data to a writer.

Examples:

TextFile is the default storage format for creating Hive tables, without data compression. Essentially, a TextFile file is stored in HDFS as text. For Data Integration, the implementation of HDFS Reader is similar to that of OSS Reader.

Optimized Row Columnar File (ORCFile) is an optimized RCFile format. It provides an efficient method for storing Hive data. HDFS Reader uses the OrcSerde class provided by Hive to read and parse ORCFile data.

> **Note**
>
> - Considering that a complex network connection is required between the default resource group and HDFS, we recommend that you use a custom resource group to run sync nodes. Make sure that your custom resource group can access the NameNode and DataNode of HDFS through a network.
> - By default, HDFS uses a network whitelist to guarantee data security. In this case, we recommend that you use a custom resource group to run HDFS sync nodes.
> - If you configure an HDFS sync node in the code editor, the HDFS connection does not need to pass the connectivity test. In this case, you can temporarily ignore connectivity test errors.
> - To synchronize data in Data Integration, you must log on as an administrator. Make sure that you have the permissions to read data from and write data to relevant HDFS files.

Features

Currently, HDFS Reader supports the following features:

- Supports the TextFile, ORCFile, RCFile, SequenceFile, CSV, and Parquet file formats. What is stored in each file must be a logical two-dimensional table.
- Reads data of various types as strings. Supports constants and column pruning.
- Supports recursive reading. Supports regular expressions that contain asterisks (*) and question marks (?).
- Compresses ORCFile files in SNAPPY or ZLIB format.
- Compresses SequenceFile files in LZO format.
- Reads multiple files concurrently.
- Compresses CSV files in GZIP, BZIP2, ZIP, LZO, LZO_DEFLATE, or SNAPPY format.
- Supports Hive 1.1.1 and Hadoop 2.7.1 (compatible with Apache JDK 1.6). HDFS Reader can work properly with Hadoop 2.5.0, Hadoop 2.6.0, and Hive 1.2.0 during testing.

> **Note** Currently, HDFS Reader cannot use concurrent threads to read a single file.

Data types

RCFile

RCFile metadata is stored in databases managed by Hive, and in different formats depending on the data type. However, HDFS Reader cannot query metadata from such databases. If you want to synchronize a file of the RCFile format, you must specify the data type for each column. If the data type is BIGINT, DOUBLE, or FLOAT, specify the data type as BIGINT, DOUBLE, or FLOAT. If the data type is VARCHAR or CHAR, specify the data type as STRING.

RCFile data types are automatically converted into the data types supported by Data Integration. The following table lists the supported data types.

Category	HDFS data type
Integer	TINYINT, SMALLINT, INT, and BIGINT
Floating point	FLOAT, DOUBLE, and DECIMAL
String	STRING, CHAR, and VARCHAR
Date and time	DATE and TIMESTAMP
Boolean	BOOLEAN
Binary	BINARY

Parquet files

Parquet file data types are automatically converted into the data types supported by Data Integration. The following table lists the supported data types.

Category	HDFS data type
Integer	INT32, INT64, and INT96
Floating point	FLOAT and DOUBLE
String	FIXED_LEN_BYTE_ARRAY
Date and time	DATE and TIMESTAMP
Boolean	BOOLEAN
Binary	BINARY

TextFile, ORCFile, and SequenceFile

TextFile metadata and ORCFile metadata are stored in databases, such as MySQL databases, managed by Hive. However, HDFS Reader cannot query metadata from such databases. If you want to convert data types during data synchronization, you must specify the data types.

TextFile, ORCFile, and SequenceFile data types are automatically converted into the data types supported by Data Integration. The following table lists the supported data types.

Category	HDFS data type
Integer	TINYINT, SMALLINT, INT, and BIGINT
Floating point	FLOAT and DOUBLE
String	STRING, CHAR, VARCHAR, STRUCT, MAP, ARRAY, UNION, and BINARY
Date and time	DATE and TIMESTAMP
Boolean	BOOLEAN

The data types are described as follows:

- LONG: integer strings in HDFS files, such as 123456789.
- DOUBLE: double value strings in HDFS files, such as 3.1415.
- BOOLEAN: Boolean strings in HDFS files, such as true and false. The strings are case-insensitive.
- DATE: date and time strings in HDFS files, such as 2014-12-31 00:00:00.

> **Note** The TIMESTAMP data type of Hive is accurate to nanoseconds. If you convert TIMESTAMP-type Hive data, such as 2015-08-21 22:40:47.397898389, in TextFile and ORCFile files into the DATE type in Data Integration, the converted data is accurate to seconds. If you need nanosecond-scale accuracy, convert TIMESTAMP-type data into the STRING type in Data Integration.

Parameters

Parameter	Description	Required	Default value

Parameter	Description	Required	Default value
path	The path of the file to read. To read multiple files, use a regular expression such as /hadoop/data_201704*. • If you specify a single HDFS file, HDFS Reader uses only one thread to read the file. • If you specify multiple HDFS files, HDFS Reader uses multiple threads. The number of threads is limited by the transmission rate, in Mbit/s. Note The actual number of threads is determined by both the number of HDFS files to be read and the specified transmission rate. • When a path contains a wildcard, HDFS Reader attempts to read all files that match the path. If the path is ended with a slash (/), HDFS Reader reads all files in the specified directory. For example, if you specify the path as /bazhen/, HDFS Reader reads all files in the bazhen directory. Currently, HDFS Reader only supports asterisks (*) and question marks (?) as file name wildcards. The syntax is similar to that of file name wildcards used on the Linux command line. Note • Data Integration considers all the files on a sync node as a single table. Make sure that all the files on each sync node can adapt to the same schema and Data Integration has the permission to read all these files. • Note: When creating Hive tables, you can specify partitions. For example, if you specify partition(day="20150820",hour="09"), a directory named /20150820 and a subdirectory named /09 are created in the corresponding table directory of the HDFS. Therefore, if you need HDFS Reader to read the data of a partition, specify the file path of the partition. For example, if you need HDFS Reader to read all the data in the partition with the date of 20150820 in the table named mytable01, specify the path as follows: `"path": "/user/hive/warehouse/mytable01/20150820/*"`	Yes	None
defaultFS	The address of the NameNode of the HDFS. If a sync node is run on the default resource group, advanced parameter settings of Hadoop, such as those related to high availability, are not supported.	Yes	None

Parameter	Description	Required	Default value
fileType	The file format. Valid values: text, orc, rc, seq, csv, and parquet. HDFS Reader automatically recognizes the file format and uses corresponding read policies. Before data synchronization, HDFS Reader checks whether all the source files match the specified format. If any source file does not match the format, the sync node fails. The valid values of the fileType parameter are described as follows: • text: the TextFile format. • orc: the ORCFile format. • rc: the RCFile format. • seq: the SequenceFile format. • csv: the common HDFS file format, that is, the logical two-dimensional table. • parquet: the common Parquet file format. **Note** TextFile and ORCFile are different formats. HDFS Reader parses files in the two formats in different ways. After being converted from a composite data type of Hive into the STRING type of Data Integration, the data in a file of the TextFile format can be different from that in the same file of the ORCFile format. Composite data types include MAP, ARRAY, STRUCT, and UNION. The following example uses the conversion from the MAP type to the STRING type as an example: • HDFS Reader converts MAP-type ORCFile data into a string: {job=80, team=60, person=70}. • HDFS Reader converts MAP-type TextFile data into a string: job:80, team:60, person:70. The conversion results show that the data remains unchanged but the formats differ slightly. Therefore, if the data to be synchronized matches a composite data type of Hive, we recommend that you use a uniform file format. Recommendations: • To use a uniform file format, we recommend that you export TextFile tables as ORCFile tables on the Hive client. • If the file format is Parquet, the parquetSchema parameter is required, which specifies the schema of the Parquet table. For the column parameter, you must specify the type parameter and specify one of the index and value parameters.	Yes	None

Parameter	Description	Required	Default value
column	The columns to read. The type parameter specifies the source data type. The index parameter specifies the ID of the column in the source table, starting from 0. The value parameter specifies the column value if the column is a constant column. By default, HDFS Reader reads all data as strings. Specify this parameter as `"column":["*"]`. You can also specify the column parameter in the following way: `{ "type": "long", "index": 0 // The first INT-type column of the source file. }, { "type": "string", "value": "alibaba" // The value of the current column, that is, a constant "alibaba". }`	Yes	None
fieldDelimiter	The column delimiter. To read TextFile data, you must specify the column delimiter. The default delimiter is comma (,). To read ORCFile data, you do not need to specify the column delimiter. The default delimiter is \u0001. • If you need each row to be converted into a column in the destination table, use a string that does not exist in every row, such as \u0001. • Do not use \n as the delimiter.	No	,
encoding	The encoding format of the file to read.	No	UTF-8
nullFormat	The string that represents null. No standard strings can represent null in text files. Therefore, Data Integration provides the nullFormat parameter to define which string represents a null pointer. For example, if you specify nullFormat:"null", Data Integration considers null as a null pointer.	No	None

Parameter	Description	Required	Default value
compress	The compression format. Available compression formats for CSV files are GZIP, BZIP2, ZIP, LZO, LZO_DEFLATE, and SNAPPY. **Note** • Do not mix up LZO with LZO_DEFLATE. • Snappy does not have a uniform stream format. Data Integration currently only supports the most popular two compression formats: hadoop-snappy (Snappy stream format in Hadoop) and framing-snappy (Snappy stream format recommended by Google). • rc indicates the RCFile format. • This parameter is not required for files of the ORCFile format.	No	None

Parameter	Description	Required	Default value
parquetSchema	The schema of the source file. This parameter is required only when the fileType parameter is set to parquet. Format: `message messageTypeName {` `required, dataType, columnName;` `..... ;` `}` The format is described as follows: • messageTypeName: the name of the MessageType object. • required: specifies whether the field is required or optional. We recommend that you set the parameter to optional for all fields. • dataType: the data type of the field. Supported data types: BOOLEAN, INT32, INT64, INT96, FLOAT, DOUBLE, BINARY, and FIXED_LEN_BYTE_ARRAY. Select BINARY if the data type is STRING. **Note** Each line, including the last one, must end with a semicolon (;). An example is provided as follows: `message m {` `optional int64 id;` `optional int64 date_id;` `optional binary datetimestring;` `optional int32 dspId;` `optional int32 advertiserId;` `optional int32 status;` `optional int64 bidding_req_num;` `optional int64 imp;` `optional int64 click_num;` `}`	No	None

Parameter	Description	Required	Default value
csvReaderConfig	The configurations for reading CSV files. The parameter value must match the MAP type. A specific CSV reader is used to read data from CSV files, which supports many configurations. The following example provides common configurations: `"csvReaderConfig":{ "safetySwitch": false, "skipEmptyRecords": false, "useTextQualifier": false }` You can use the following parameters and their default values: `boolean caseSensitive = true; char textQualifier = 34; boolean trimWhitespace = true; boolean useTextQualifier = true; // Specifies whether to use escape characters for CSV files. char delimiter = 44; // The delimiter. char recordDelimiter = 0; char comment = 35; boolean useComments = false; int escapeMode = 1; boolean safetySwitch = true; // Specifies whether to limit the length of each column to 100,000 characters. boolean skipEmptyRecords = true; // Specifies whether to skip empty rows. boolean captureRawRecord = true;`	No	None

Parameter	Description	Required	Default value
hadoopConfig	The advanced parameter settings of Hadoop, such as those related to high availability. `"hadoopConfig":{ "dfs.nameservices": "testDfs", "dfs.ha.namenodes.testDfs": "namenode1,namenode2", "dfs.namenode.rpc-address.youkuDfs.namenode1": "", "dfs.namenode.rpc-address.youkuDfs.namenode2": "", "dfs.client.failover.proxy.provider.testDfs": "org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider" }`	No	None

Configure HDFS Reader by using the codeless UI

Currently, the codeless user interface (UI) is not supported for HDFS Reader.

Configure HDFS Reader by using the code editor

In the following code, a node is configured to read data from an HDFS. For more information about the parameters, see the preceding parameter description.

```
{
    "type": "job",
    "version": "2.0",
    "steps": [
        {
            "stepType": "hdfs", // The reader type.
            "parameter": {
                "path": "", // The path of the file to read.
                "datasource": "", // The connection name.
                "column": [
                    {
                        "index": 0, // The ID of the column in the source table.
                        "type": "string" // The data type.
                    },
                    {
                        "index": 1,
                        "type": "long"
                    },
                    {
                        "index": 2,
                        "type": "double"
                    },
                    {
                        "index": 3,
                        "type": "boolean"
                    },
                    {
                        "format": "yyyy-MM-dd HH:mm:ss", // The format of the time.
```

```
                        "index": 4,
                        "type": "date"
                    }
                ],
                "fieldDelimiter": "," // The column delimiter.
                "encoding": "UTF-8", // The encoding format.
                "fileType": "" // The file format.
            },
            "name": "Reader",
            "category": "reader"
        },
        {// The following template is used to configure the writer. For more information, s
ee the corresponding topic.
            "stepType": "stream",
            "parameter": {},
            "name": "Writer",
            "category": "writer"
        }
    ],
    "setting": {
        "errorLimit": {
            "record": "" // The maximum number of dirty data records allowed.
        },
        "speed": {
            "concurrent": 3, // The maximum number of concurrent threads.
            "throttle": false, // Specifies whether to enable bandwidth throttling. A value
of false indicates that the bandwidth is not throttled. A value of true indicates that the
bandwidth is throttled. The maximum transmission rate takes effect only if you set this par
ameter to true.
        }
    },
    "order": {
        "hops": [
            {
                "from": "Reader",
                "to": "Writer"
            }
        ]
    }
}
```

3.6.3.4. Configure MaxCompute Reader

This topic describes the data types and parameters supported by MaxCompute Reader and how to configure it by using the codeless UI and code editor.

MaxCompute Reader can read data from a MaxCompute project by using the MaxCompute Tunnel service based on the source project, table, partition, and table fields that you have configured.

MaxCompute Reader cannot read views. It can read only partitioned tables and non-partitioned tables. To allow MaxCompute Reader to read partitioned tables, you must specify the partition information. For example, set pt to 1 and ds to hangzhou for the t0 table. The partition information is not required for non-partitioned tables. Additionally, you can select some or all of the table fields, change the order in which the fields are arranged, or add constant fields and partition key columns. Note that partition key columns are not table fields.

Data types

The following table lists the data types supported by MaxCompute Reader.

Category	Data Integration data type	MaxCompute data type
Integer	LONG	BIGINT, INT, TINYINT, and SMALLINT
Boolean	BOOLEAN	BOOLEAN
Date and time	DATE	DATETIME and TIMESTAMP
Floating point	DOUBLE	FLOAT, DOUBLE, and DECIMAL
Binary	BYTES	BINARY
Complex	STRING	ARRAY, MAP, and STRUCT

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None
table	The name of the source table. The name is not case-sensitive.	Yes	None
partition	The partitions that MaxCompute Reader reads. Linux shell wildcards are supported. An asterisk (*) represents zero or more characters, and a question mark (?) represents that the previous character can be included or not. Assume that a partitioned table named test has four partitions: pt=1 and ds=hangzhou, pt=1 and ds=shanghai, pt=2 and ds=hangzhou, and pt=2 and ds=beijing. • To read data from the partition with pt=1 and ds=shanghai, enter `"partition":"pt=1/ds=shanghai"`. • To read data from all the partitions with pt=1, enter `"partition":"pt=1/ds=*"`. • To read data from all the partitions in the test table, enter `"partition":"pt=*/ds=*"`.	Required only for writing data to a partitioned table	None

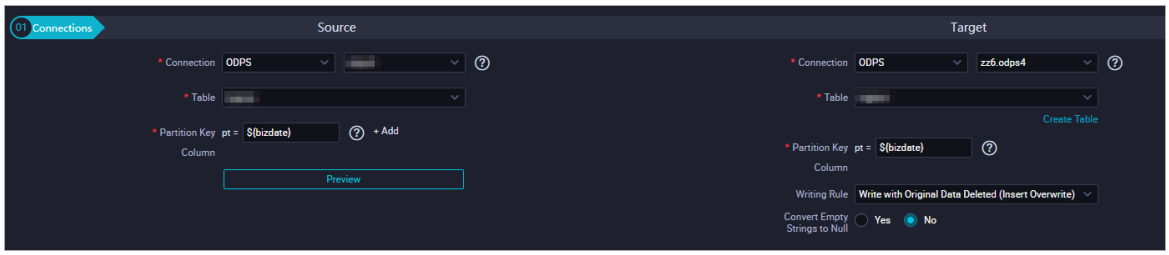
Parameter	Description	Required	Default value
column	The columns in the source table that MaxCompute Reader reads. Assume that the fields of a table named test are id, name, and age. • To read the fields in turn, enter `"column":["id","name","age"]` or `"column":["*"]`. **Note** We recommend that you do not set "column": ["*"]. This is because data synchronization may fail if the source table changes in the column order, data type, or number of columns. • To read the name and id fields in turn, enter `"column":["name","id"]`. • You can add a constant field to extracted data for the purpose of proper mapping between source table columns and destination table columns. Each constant must be enclosed in single quotation marks (`' '`). For example, if you set `"column":["age","name","'1988-08-08 08:08:08'","id"]`, the data extracted contains an age column, a name column, a constant "1988-08-08 08:08:08", and an id column in turn. The single quotation marks (`' '`) are used to identify constant columns. The constant column values exclude the single quotation marks (`' '`). **Note** ◦ MaxCompute Reader does not use SELECT statements to read data. Therefore, you cannot specify function fields. ◦ The column parameter must explicitly specify a set of columns to be synchronized. The parameter cannot be left empty.	Yes	None

Configure MaxCompute Reader by using the codeless UI

On the **DataStudio** page, create a sync node under a workflow and configure the node.

1. Configure the connections.

   Configure the connections to the source and destination data stores for the sync node.



Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and then select a connection name that you have configured in DataWorks.
Table	The table parameter in the preceding parameter description.
Partition Key Column	The partition information. You can click **Add** on the right to add partition key columns.

> **Note** To synchronize all columns in the source table, enter "column":[""]. The partition parameter supports wildcards and includes one or more partitions.
>
> - `"partition":"pt=20140501/ds=*"` specifies that all ds partitions with pt=20140501 are to be synchronized.
> - `"partition":"pt=top?"` specifies that the partitions with pt=top and pt=to are to be synchronized.
>
> You can specify the partition key columns to be synchronized, such as a partition key column named pt. Assume that the partition key column of a MaxCompute table is pt=${bdp.system.bizdate}. You can configure the column to be synchronized to pt. Ignore it if the column is marked as unidentified. To synchronize all partitions, enter pt=*. To synchronize specified partitions, specify the corresponding dates.

2. Configure field mapping. It is equivalent to setting the column parameter provided in the preceding parameter description.

   Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right. You can click **Add** to add a field, or move the pointer over a field and click the **Delete** icon to delete the field.

GUI element	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	Click Auto Layout to sort the fields based on specified rules.
Change Fields	Click the Change Fields icon. In the Change Fields dialog box, you can manually edit fields in the source table. Each field occupies a row. The first and the last blank rows are included, whereas other blank rows are ignored.

GUI element	Description
Add	◦ Click Add to add a field. You can enter constants. Each constant must be enclosed in single quotation marks (' '), such as 'abc' and '123'. ◦ You can use scheduling parameters, such as ${bizdate}. ◦ You can enter functions supported by relational databases, such as now() and count(1). ◦ Fields that cannot be parsed are indicated by Unidentified.

3. Configure channel control policies.

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read data from and write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure MaxCompute Reader by using the code editor

In the following code, a node is configured to read data from MaxCompute. For more information about the parameters, see the preceding parameter description.

```
{
    "type":"job", // The type of the sync node.
    "version":"2.0", // The version number.
    "steps":[
        {
            "stepType":"odps",// The reader type.
            "parameter":{
                "partition":[], // The partitions that MaxCompute Reader reads.
                "isCompress":false, // Specifies whether to enable compression.
                "datasource":"",// The connection name.
                "column":[// The columns to be synchronized.
                    "id"
                ],
                "emptyAsNull":true,
                "table":""// The table name.
            },
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"stream", // The writer type.
            "parameter":{ // The parameters that you specify for the writer.
            },
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0" // The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false,// Specifies whether to enable bandwidth throttling. A value o
f false indicates that the bandwidth is not throttled. A value of true indicates that the b
andwidth is throttled. The maximum transmission rate takes effect only if you set this para
meter to true.
            "concurrent":1,// The maximum number of concurrent threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader", // The source connection of the node.
                "to":"Writer" // The destination connection of the node.
            }
        ]
    }
}
```

3.6.3.5. Configure MongoDB Reader

This topic describes the data types and parameters supported by MongoDB Reader and how to configure it by using the code editor.

MongoDB Reader connects to a remote MongoDB database by using the Java client named MongoClient and reads data from the database. The latest version of MongoDB has improved the locking feature from database locks to document locks. By using the powerful functionalities of indexes in MongoDB, MongoDB Reader can efficiently read data from MongoDB databases.

> **Note**
>
> - If you use ApsaraDB for MongoDB, the MongoDB database has a root account by default. For security concerns, Data Integration supports access to a MongoDB database only by using a MongoDB database account. When you add a MongoDB connection, do not use the root account for access.
> - JavaScript syntax is not supported for queries.

MongoDB Reader shards data in the MongoDB database based on specified rules, reads data from the database with multiple threads, and then converts the data to a format readable by Data Integration.

Data types

MongoDB Reader supports most MongoDB data types. Make sure that your data types are supported.

The following table lists the data types supported by MongoDB Reader.

Category	MongoDB data type
Long	INT, LONG, DOCUMENT.INT, and DOCUMENT.LONG
Double	DOUBLE and DOCUMENT.DOUBLE
String	STRING, ARRAY, DOCUMENT.STRING, DOCUMENT.ARRAY, and COMBINE
Date	DATE and DOCUMENT.DATE
Boolean	BOOLEAN and DOCUMENT.BOOLEAN
Bytes	BYTES and DOCUMENT.BYTES

Note

- The DOCUMENT data type is used to store embedded documents. It is also called the OBJECT data type.
- The following content describes how to use the COMBINE data type:

  When MongoDB Reader reads data from a MongoDB database, it combines and converts multiple fields in MongoDB documents to a JSON string.

  For example, doc1, doc2, and doc3 are three MongoDB documents with different fields, which are represented by keys instead of key-value pairs. The keys a and b represent common fields in all the three documents. The key x_n represents an unfixed field.

  ```
  doc1: a b x_1 x_2
  ```

  ```
  doc2: a b x_2 x_3 x_4
  ```

  ```
  doc3: a b x_5
  ```

  To import the preceding three MongoDB documents to MaxCompute, you must specify the fields to retain, set a name for each combined string, and set the data type of each combined string to COMBINE in the configuration file. Make sure that the name of each combined string is unique among all existing fields in the documents.

  ```
  "column": [
  {
  "name": "a",
  "type": "string",
  },
  {
  "name": "b",
  "type": "string",
  },
  {
  "name": "doc",
  "type": "combine",
  }
  ]
  ```

  The following table lists the output in MaxCompute.

odps_column1	odps_column2
a	b
a	b
a	b

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None
collectionName	The name of the replica set in MongoDB.	Yes	None
column	The columns in MongoDB. • name: the name of the column. • type: the data type of the column. • splitter: the delimiter. Specify this parameter only when you need to convert the string to an array. MongoDB supports arrays, but Data Integration does not. The array elements read by MongoDB are joined to a string by using this delimiter.	Yes	None
query	The filter condition for obtaining data from MongoDB. Only the time type is supported. For example, you can use the statement `"query":"{'operationTime': {'$gte':ISODate('${last_day}T00:00:00.424+0800')}}"` to obtain data where the time specified by operationTime is not earlier than 00:00 on the day specified by ${last_day}. In the preceding JSON string, ${last_day} is a scheduling parameter of DataWorks. The format is $[yyyy-mm-dd]. You can use comparison operators (such as $gt, $lt, $gte, and $lte), logical operators (such as $and and $or), and functions (such as max, min, sum, avg, and ISODate) supported by MongoDB as needed.	No	None

Configure MongoDB Reader by using the codeless UI

The codeless UI is not supported for MongoDB Reader.

Configure MongoDB Reader by using the code editor

In the following code, a node is configured to read data from a MongoDB database. For more information about the parameters, see the preceding parameter description.

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
            "reader": {
            "plugin": "mongodb", // The reader type.
            "parameter": {
                "datasource": "datasourceName", // The connection name.
                "collectionName": "tag_data", // The name of the MongoDB collection.
                "query":"",
                "column": [
                        {
                            "name": "unique_id", // The field name.
                            "type": "string" // The data type.
                        },
                        {
```

```
                "name": "sid",
                "type": "string"
            },
            {
                "name": "user_id",
                "type": "string"
            },
            {
                "name": "auction_id",
                "type": "string"
            },
            {
                "name": "content_type",
                "type": "string"
            },
            {
                "name": "pool_type",
                "type": "string"
            },
            {
                "name": "frontcat_id",
                "type": "array",
                "splitter": ""
            },
            {
                "name": "categoryid",
                "type": "array",
                "splitter": ""
            },
            {
                "name": "gmt_create",
                "type": "string"
            },
            {
                "name": "taglist",
                "type": "array",
                "splitter": " "
            },
            {
                "name": "property",
                "type": "string"
            },
            {
                "name": "scorea",
                "type": "int"
            },
            {
                "name": "scoreb",
                "type": "int"
            },
            {
                "name": "scorec",
                "type": "int"
            },
```

```
                {
                                "name": "a.b",
                                "type": "document.int"
                            },
                            {
                                "name": "a.b.c",
                                "type": "document.array",
                                "splitter": " "
                            }
                    ]
                }
            },
            {
                "stepType":"stream",
                "parameter":{},
                "name":"Writer",
                "category":"writer"
            }
        ],
        "setting":{
            "errorLimit":{
                "record":"0"// The maximum number of dirty data records allowed.
            },
            "speed":{
                "throttle":false,// Specifies whether to enable bandwidth throttling. A value o
f false indicates that the bandwidth is not throttled. A value of true indicates that the b
andwidth is throttled. The maximum transmission rate takes effect only if you set this para
meter to true.
                "concurrent":1,// The maximum number of concurrent threads.
            }
        },
        "order":{
            "hops":[
                {
                    "from":"Reader",
                    "to":"Writer"
                }
            ]
        }
}
```

> **Note** You cannot retrieve data elements from arrays.

3.6.3.6. Configure Db2 Reader

This topic describes the data types and parameters supported by Db2 Reader and how to configure it by using the code editor.

Db2 Reader allows you to read data from Db2. Db2 Reader connects to a remote Db2 database and runs a SELECT statement to select and read data from the database.

Specifically, Db2 Reader connects to a remote Db2 database through Java Database Connectivity (JDBC), generates a SELECT statement based on your configurations, and then sends the statement to the database. The Db2 database runs the statement and returns the result. Then, Db2 Reader assembles the returned data to abstract datasets in custom data types supported by Data Integration, and passes the datasets to a writer.

- Db2 Reader generates the SELECT statement based on the table, column, and where parameters that you have configured, and sends the generated SELECT statement to the Db2 database.
- If you specify the querySql parameter, Db2 Reader directly sends the value of this parameter to the Db2 database.

Db2 Reader supports most Db2 data types. Make sure that your data types are supported.

The following table lists the data types supported by Db2 Reader.

Category	Db2 data type
Integer	SMALLINT
Floating point	DECIMAL, REAL, and DOUBLE
String	CHAR, CHARACTER, VARCHAR, GRAPHIC, VARGRAPHIC, LONG VARCHAR, CLOB, LONG VARGRAPHIC, and DBCLOB
Date and time	DATE, TIME, and TIMESTAMP
Boolean	N/A
Binary	BLOB

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None
jdbcUrl	The JDBC URL for connecting to the Db2 database. In accordance with official Db2 specifications, the URL must be in the `jdbc:db2://ip:port/database` format. You can also specify the information of the attachment facility.	Yes	None
username	The username for connecting to the database.	Yes	None
password	The password for connecting to the database.	Yes	None
table	The name of the table to be synchronized. You can select only one source table for each sync node.	Yes	None

Parameter	Description	Required	Default value
column	The columns to be synchronized from the source table. The columns are described in a JSON array. The default value is [*], which indicates all columns. • Column pruning is supported. You can select and export specific columns. • Change of the column order is supported. You can export the columns in an order different from that specified in the schema of the table. • Constants are supported. The column names must be arranged in compliance with the SQL syntax supported by Db2, for example, `["id", "1", "'const name'", "null", "upper('abc_lower')", "2.3", "true"]`. ◦ id: a column name. ◦ 1: an integer constant. ◦ 'const name': a string constant, which is enclosed in single quotation marks (' '). ◦ null: a null pointer. ◦ upper('abc_lower'): a function expression. ◦ 2.3: a floating-point constant. ◦ true: a Boolean value. • The column parameter must explicitly specify a set of columns to be synchronized. The parameter cannot be left empty.	Yes	None
splitPk	The field used for data sharding when Db2 Reader extracts data. If you specify the splitPk parameter, the table is sharded based on the shard key specified by this parameter. Data Integration then runs concurrent threads to synchronize data. This improves efficiency. • We recommend that you set the splitPk parameter to the primary key of the table. Based on the primary key, data can be well distributed to different shards, but not intensively distributed to certain shards. • Currently, the splitPk parameter supports data sharding only for integers but not for other data types such as string, floating point, and date. If you specify this parameter to a column of an unsupported type, Db2 Reader returns an error.	No	""
where	The WHERE clause. Db2 Reader generates a SELECT statement based on the table, column, and where parameters that you have configured, and uses the generated SELECT statement to select and read data. For example, set this parameter to `gmt_create>$bizdate`. You can use the WHERE clause to synchronize incremental data. If you do not specify the where parameter or leave it empty, all data is synchronized.	No	None

Parameter	Description	Required	Default value
querySql	The SELECT statement used for refined data filtering. If you specify this parameter, Data Integration directly filters data based on this parameter. For example, if you want to join multiple tables for data synchronization, set this parameter to `select a,b from table_a join table_b on table_a.id = table_b.id`. If you specify the querySql parameter, Db2 Reader ignores the table, column, and where parameters that you have configured.	No	None
fetchSize	The number of data records to read at a time. This parameter determines the number of interactions between Data Integration and the database and affects reading efficiency. **Note** A value greater than 2048 may lead to out of memory (OOM) during the data synchronization process.	No	*1024*

Configure Db2 Reader by using the codeless UI

Currently, the codeless user interface (UI) is not supported for Db2 Reader.

Configure Db2 Reader by using the code editor

In the following code, a node is configured to read data from a Db2 database.

```
{
    "type":"job",
    "version":"2.0", // The version number.
    "steps":[
        {
            "stepType":"db2", // The reader type.
            "parameter":{
                "password":"", // The password for connecting to the database.
                "jdbcUrl":"", // The JDBC URL for connecting to the Db2 database.
                "column":[
                    "id"
                ],
                "where":"", // The WHERE clause.
                "splitPk":"", // The field used for data sharding. If you specify the split
Pk parameter, the table is sharded based on the shard key specified by this parameter.
                "table":"", // The name of the table to be synchronized.
                "username":"" // The username for connecting to the database.
            },
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"stream",
            "parameter":{},
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0" // The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false, // Specifies whether to enable bandwidth throttling. A value
of false indicates that the bandwidth is not throttled. A value of true indicates that the
bandwidth is throttled. The maximum transmission rate takes effect only if you set this par
ameter to true.
            "concurrent":1, // The maximum number of concurrent threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

Additional instructions

- Data synchronization between primary and secondary databases

A secondary Db2 database can be deployed for disaster recovery. The secondary database continuously synchronizes data from the primary database based on binlogs. Especially when network conditions are unfavorable, data latency between the primary and secondary databases is unavoidable, which can lead to data inconsistency.

- Concurrency control

  Db2 is a relational database management system (RDBMS), which supports strong consistency for data queries. A database snapshot is created before a sync node starts. Db2 Reader reads data from the database snapshot. Therefore, if new data is written to the database during data synchronization, Db2 Reader cannot obtain the new data.

  Data consistency cannot be guaranteed when you enable Db2 Reader to run concurrent threads on a single sync node.

  Db2 Reader shards the table based on the splitPk parameter and runs multiple concurrent threads to synchronize data. These concurrent threads belong to different transactions. They read data at different time points. This means that the concurrent threads observe different snapshots.

  Theoretically, the data inconsistency issue is unavoidable if a single sync node includes multiple threads. However, two workarounds are available:

  - Do not enable concurrent threads on a single sync node. Essentially, do not specify the splitPk parameter. In this way, data consistency is guaranteed although data is synchronized at a low efficiency.
  - Disable writers to make sure that the data is unchanged during data synchronization. For example, lock the table and disable data synchronization between primary and secondary databases. In this way, data is synchronized efficiently but your ongoing services may be interrupted.

- Character encoding

  Db2 Reader uses JDBC, which can automatically convert the encoding of characters. Therefore, you do not need to specify the encoding format.

- Incremental data synchronization

  Db2 Reader connects to a database through JDBC and uses a SELECT statement with a WHERE clause to read incremental data in the following ways:

  - For data in batches, incremental add, update, and delete operations (including logically delete operations) are distinguished by timestamps. Specify the WHERE clause based on the timestamp. The timestamp must be later than the latest timestamp in the last synchronization.
  - For streaming data, specify the WHERE clause based on the data record ID. The data record ID must be larger than the maximum ID involved in the last synchronization.

  If incremental data cannot be distinguished, Db2 Reader cannot perform incremental synchronization but can perform full synchronization only.

- Syntax validation

  Db2 Reader allows you to specify custom SELECT statements by using the querySql parameter but does not verify the syntax of the custom SELECT statements.

3.6.3.7. Configure MySQL Reader

This topic describes the data types and parameters supported by MySQL Reader and how to configure it by using the codeless user interface (UI) and code editor.

MySQL Reader connects to a remote MySQL database through Java Database Connectivity (JDBC), generates a SELECT statement based on your configurations, and then sends the statement to the database. The MySQL database runs the statement and returns the result. Then, MySQL Reader assembles the returned data to abstract datasets in custom data types supported by Data Integration, and passes the datasets to a writer.

In short, MySQL Reader connects to a remote MySQL database and runs a SELECT statement to select and read data from the database.

MySQL Reader can read tables and views. For table fields, you can specify all or some of the columns in sequence, adjust the column order, specify constant fields, and configure MySQL functions, such as now().

Data types

The following table lists the data types supported by MySQL Reader.

Category	MySQL data type
Integer	INT, TINYINT, SMALLINT, MEDIUMINT, and BIGINT
Floating point	FLOAT, DOUBLE, and DECIMAL
String	VARCHAR, CHAR, TINYTEXT, TEXT, MEDIUMTEXT, and LONGTEXT
Date and time	DATE, DATETIME, TIMESTAMP, TIME, and YEAR
Boolean	BIT and BOOLEAN
Binary	TINYBLOB, MEDIUMBLOB, BLOB, LONGBLOB, and VARBINARY

> **Note**
>
> - Data types that are not listed in the table are not supported.
> - MySQL Reader considers tinyint(1) as the INTEGER type.
> - Currently, MySQL Reader does not support MySQL 8.0 or later.

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None
table	The name of the table to be synchronized. You can select only one source table for each sync node.	Yes	None

Parameter	Description	Required	Default value
column	The columns to be synchronized from the source table. The columns are described in a JSON array. The default value is [*], which indicates all columns. • Column pruning is supported. You can select and export specific columns. • Change of the column order is supported. You can export the columns in an order different from that specified in the schema of the table. • Constants are supported. The column names must be arranged in compliance with the SQL syntax supported by MySQL, for example, `["id","table","1","'mingya.wmy'","'null'","to_char(a+1)","2.3","true"]`. ○ id: a column name. ○ table: the name of a column that contains reserved keywords. ○ 1: an integer constant. ○ 'mingya.wmy': a string constant, which is enclosed in single quotation marks (' '). ○ null: ■ '' '' indicates an empty value. ■ null indicates a null value. ■ 'null' indicates the string null. ○ to_char(a + 1): a function expression. ○ 2.3: a floating-point constant. ○ true: a Boolean value. • The column parameter must explicitly specify a set of columns to be synchronized. The parameter cannot be left empty.	Yes	None

Parameter	Description	Required	Default value
splitPk	The field used for data sharding when MySQL Reader extracts data. If you specify the splitPk parameter, the table is sharded based on the shard key specified by this parameter. Data Integration then runs concurrent threads to synchronize data. This improves efficiency. • We recommend that you set the splitPk parameter to the primary key of the table. Based on the primary key, data can be well distributed to different shards, but not intensively distributed to certain shards. • Currently, the splitPk parameter supports data sharding only for integers but not for other data types such as string, floating point, and date. If you specify this parameter to a column of an unsupported type, MySQL Reader ignores the splitPk parameter and synchronizes data through a single thread. • If you do not specify the splitPk parameter or leave it empty, Data Integration synchronizes data through a single thread.	No	None
where	The WHERE clause. For example, set this parameter to `gmt_create>$bizdate`. • You can use the WHERE clause to synchronize incremental data. If you do not specify the where parameter or leave it empty, all data is synchronized. • Do not set the where parameter to limit 10, which does not conform to the constraints of MySQL on the SQL WHERE clause.	No	None
querySql (only available in the code editor)	The SELECT statement used for refined data filtering. If you specify this parameter, Data Integration directly filters data based on this parameter. For example, if you want to join multiple tables for data synchronization, set this parameter to `select a,b from table_a join table_b on table_a.id = table_b.id`. The priority of the querySql parameter is higher than those of the table, column, where, and splitPk parameters. If you specify the querySql parameter, MySQL Reader ignores the table, column, where, and splitPk parameters that you have configured. The datasource parameter parses information, including the username and password, from this parameter.	No	None

Parameter	Description	Required	Default value
singleOrMulti (applicable only to database and table sharding)	Specifies whether to shard the database or table. After you switch from the codeless UI to the code editor, the following configuration is automatically generated: `"singleOrMulti":"multi"`. However, if you use the code editor since the beginning, the configuration is not automatically generated and you must manually specify this parameter. If you do not specify this parameter or leave it empty, MySQL Reader can only read data from the first shard.	Yes	*multi*

Configure MySQL Reader by using the codeless UI

1. Configure the connections.

   Configure the source and destination connections for the sync node.

Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and enter the name of a connection that has been configured in DataWorks.
Table	The table parameter in the preceding parameter description.
Filter	The filter condition for the data to be synchronized. Currently, filtering based on the limit keyword is not supported. The SQL syntax is determined by the selected connection.
Shard Key	The shard key. You can use a column in the source table as the shard key. We recommend that you use the primary key or an indexed column. Only integer fields are supported. If data sharding is performed based on the configured shard key, data can be read concurrently to improve data synchronization efficiency. **Note** The Shard Key parameter is displayed only when you configure the source connection for a sync node.

2. Configure field mapping, that is, the column parameter in the preceding parameter description.

   Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right. You can click **Add** to add a field, or move the pointer over a field and click the **Delete** icon to delete the field.

Parameter	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.

Parameter	Description
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	Click Auto Layout. The fields are automatically sorted based on specified rules.
Change Fields	Click the Change Fields icon. In the Change Fields dialog box that appears, you can manually edit fields in the source table. Each field occupies a row. The first and the last blank rows are included, whereas other blank rows are ignored.
Add	◦ Click Add to add a field. You can enter constants. Each constant must be enclosed in single quotation marks (' '), such as 'abc' and '123'. ◦ You can use scheduling parameters, such as ${bizdate}. ◦ You can enter functions supported by relational databases, such as now() and count(1). ◦ Fields that cannot be parsed are indicated by Unidentified.

3. Configure channel control policies.

Parameter	Description
Expected Concurrency	The maximum number of concurrent threads to read data from or write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure MySQL Reader by using the code editor

In the following code, a node is configured to read data from a database or table that is not sharded. For more information about the parameters, see the preceding parameter description.

```
{
    "type":"job",
    "version":"2.0", // The version number.
    "steps":[
```

```
        {
            "stepType":"mysql", // The reader type.
            "parameter":{
                "column":[ // The columns to be synchronized.
                    "id"
                ],
                "connection":[
                    {   "querysql":["select a,b from join1 c join join2 d on c.id = d.id;"]
, // Specify the querySql parameter in the connection parameter as a string.
                        "datasource":"", // The connection name.
                        "table":[
                            "xxx" // The name of the table to be synchronized.
                        ]
                    }
                ],
                "where":"", // The WHERE clause.
                "splitPk":"", // The shard key.
                "encoding":"UTF-8" // The encoding format.
            },
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"stream",
            "parameter":{},
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0" // The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false, // Specifies whether to enable bandwidth throttling. A value
of false indicates that the bandwidth is not throttled. A value of true indicates that the
bandwidth is throttled. The maximum transmission rate takes effect only if you set this par
ameter to true.
            "concurrent":1, // The maximum number of concurrent threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

In the following code, a node is configured to read data from a database or table that is sharded. For more information about the parameters, see the preceding parameter description.

Note In the case of database and table sharding, MySQL Reader can read multiple MySQL tables with the same schema.

```
{
    "type": "job",
    "version": "1.0",
    "configuration": {
        "reader": {
            "plugin": "mysql",
            "parameter": {
                "connection": [
                    {
                        "table": [
                            "tbl1",
                            "tbl2",
                            "tbl3"
                        ],
                        "datasource": "datasourceName1"
                    },
                    {
                        "table": [
                            "tbl4",
                            "tbl5",
                            "tbl6"
                        ],
                        "datasource": "datasourceName2"
                    }
                ],
                "singleOrMulti": "multi",
                "splitPk": "db_id",
                "column": [
                    "id", "name", "age"
                ],
                "where": "1 < id and id < 100"
            }
        },
        "writer": {
        }
    }
}
```

3.6.3.8. Configure Oracle Reader

This topic describes the data types and parameters supported by Oracle Reader and how to configure it by using the codeless user interface (UI) and code editor.

Oracle Reader allows you to read data from Oracle. Oracle Reader connects to a remote Oracle database and runs a SELECT statement to select and read data from the database.

Specifically, Oracle Reader connects to a remote Oracle database through Java Database Connectivity (JDBC), generates a SELECT statement based on your configurations, and then sends the statement to the database. The Oracle database runs the statement and returns the result. Then, Oracle Reader assembles the returned data to abstract datasets in custom data types supported by Data Integration, and passes the datasets to a writer.

- Oracle Reader generates the SELECT statement based on the table, column, and where parameters that you have configured, and sends the generated SELECT statement to the Oracle database.
- If you specify the querySql parameter, Oracle Reader directly sends the value of this parameter to the Oracle database.

Data types

Oracle Reader supports most Oracle data types. Make sure that your data types are supported.

The following table lists the data types supported by Oracle Reader.

Category	Oracle data type
Integer	NUMBER, ROWID, INTEGER, INT, and SMALLINT
Floating point	NUMERIC, DECIMAL, FLOAT, DOUBLE PRECISION, and REAL
String	LONG, CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, CLOB, NCLOB, CHARACTER, CHARACTER VARYING, CHAR VARYING, NATIONAL CHARACTER, NATIONAL CHAR, NATIONAL CHARACTER VARYING, NATIONAL CHAR VARYING, and NCHAR VARYING
Date and time	TIMESTAMP and DATE
Boolean	BIT and BOOLEAN
Binary	BLOB, BFILE, RAW, and LONG RAW

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None
table	The name of the table to be synchronized.	Yes	None

Parameter	Description	Required	Default value
column	The columns to be synchronized from the source table. The columns are described in a JSON array. The default value is [*], which indicates all columns. • Column pruning is supported. You can select and export specific columns. • Change of the column order is supported. You can export the columns in an order different from that specified in the schema of the table. • Constants are supported. The column names must be arranged in JSON format. `["id", "1", "'mingya.wmy'", "null", "to_char(a + 1)", "2.3", "true"]` ◦ id: a column name. ◦ 1: an integer constant. ◦ 'mingya.wmy': a string constant, which is enclosed in single quotation marks (' '). ◦ null: a null pointer. ◦ to_char(a + 1): a function expression. ◦ 2.3: a floating-point constant. ◦ true: a Boolean value. • The column parameter must be specified.	Yes	None
splitPk	The field used for data sharding when Oracle Reader extracts data. If you specify the splitPk parameter, the table is sharded based on the shard key specified by this parameter. Data Integration then runs concurrent threads to synchronize data. This improves efficiency. • We recommend that you set the splitPk parameter to the primary key of the table. Based on the primary key, data can be well distributed to different shards, but not intensively distributed to certain shards. • The data types supported by the splitPk parameter include INTEGER, STRING, FLOAT, and DATE. • If you do not specify the splitPk parameter or leave it empty, Oracle Reader synchronizes data through a single thread.	No	None
where	The WHERE clause. Oracle Reader generates a SELECT statement based on the table, column, and where parameters that you have configured, and uses the generated SELECT statement to select and read data. For example, set this parameter to row_number() or `id>2 and sex=1`. • You can use the WHERE clause to synchronize incremental data. • If you do not specify the where parameter or leave it empty, all data is synchronized.	No	None

Parameter	Description	Required	Default value
querySql (only available in the code editor)	The SELECT statement used for refined data filtering. If you specify this parameter, Data Integration directly filters data based on this parameter. For example, if you want to join multiple tables for data synchronization, set this parameter to `select a,b from table_a join table_b on table_a.id = table_b.id`. If you specify the querySql parameter, Oracle Reader ignores the table, column, and where parameters that you have configured.	No	None
fetchSize	The number of data records to read at a time. This parameter determines the number of interactions between Data Integration and the database and affects reading efficiency. **Note** A value greater than 2048 may lead to out of memory (OOM) during the data synchronization process.	No	1024

Configure Oracle Reader by using the codeless UI

1. Configure the connections.

   Configure the source and destination connections for the sync node.

Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and enter the name of a connection that has been configured in DataWorks.
Table	The table parameter in the preceding parameter description.
Filter	The filter condition for the data to be synchronized. Currently, filtering based on the limit keyword is not supported. The SQL syntax is determined by the selected connection.
Shard Key	The shard key. You can use a column in the source table as the shard key. We recommend that you use the primary key or an indexed column. Only integer fields are supported. If data sharding is performed based on the configured shard key, data can be read concurrently to improve data synchronization efficiency. **Note** The Shard Key parameter is displayed only when you configure the source connection for a sync node.

2. Configure field mapping, that is, the column parameter in the preceding parameter description.

   Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right. You can click **Add** to add a field, or move the pointer over a field and click the **Delete** icon to delete the field.

Parameter	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	Click Auto Layout. The fields are automatically sorted based on specified rules.
Change Fields	Click the Change Fields icon. In the Change Fields dialog box that appears, you can manually edit fields in the source table. Each field occupies a row. The first and the last blank rows are included, whereas other blank rows are ignored.
Add	◦ Click Add to add a field. You can enter constants. Each constant must be enclosed in single quotation marks (' '), such as 'abc' and '123'. ◦ You can use scheduling parameters, such as ${bizdate}. ◦ You can enter functions supported by relational databases, such as now() and count(1). ◦ Fields that cannot be parsed are indicated by Unidentified.

3. Configure channel control policies.

Parameter	Description
Expected Concurrency	The maximum number of concurrent threads to read data from or write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure Oracle Reader by using the code editor

In the following code, a node is configured to read data from an Oracle database.

```
{
    "type":"job",
    "version":"2.0", // The version number.
    "steps":[
        {
            "stepType":"oracle",
            "parameter":{
                "fetchSize":1024, // The number of data records to read at a time.
                "datasource":"", // The connection name.
                "column":[ // The columns to be synchronized.
                    "id",
                    "name"
                ],
                "where":"", // The WHERE clause.
                "splitPk":"", // The shard key.
                "table":"" // The name of the table to be synchronized.
            },
            "name":"Reader",
            "category":"reader"
        },
        {// The following template is used to configure Stream Writer. For more information about how to configure other writers, see the corresponding topic.
            "stepType":"stream",
            "parameter":{},
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0" // The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false, // Specifies whether to enable bandwidth throttling. A value of false indicates that the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maximum transmission rate takes effect only if you set this parameter to true.
            "concurrent":1, // The maximum number of concurrent threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
} "to":"Writer"
            }
        ]
    }
}
```

Additional instructions

- Data synchronization between primary and secondary databases

  A secondary Oracle database can be deployed for disaster recovery. The secondary database continuously synchronizes data from the primary database based on binlogs. Especially when network conditions are unfavorable, data latency between the primary and secondary databases is unavoidable, which can lead to data inconsistency.

- Concurrency control

  Oracle is a relational database management system (RDBMS), which supports strong consistency for data queries. A database snapshot is created before a sync node starts. Oracle Reader reads data from the database snapshot. Therefore, if new data is written to the database during data synchronization, Oracle Reader cannot obtain the new data.

  Data consistency cannot be guaranteed when you enable Oracle Reader to run concurrent threads on a single sync node.

  Oracle Reader shards the table based on the splitPk parameter and runs multiple concurrent threads to synchronize data. These concurrent threads belong to different transactions. They read data at different time points. This means that the concurrent threads observe different snapshots.

  Theoretically, the data inconsistency issue is unavoidable if a single sync node includes multiple threads. However, two workarounds are available:

  - Do not enable concurrent threads on a single sync node. Essentially, do not specify the splitPk parameter. In this way, data consistency is guaranteed although data is synchronized at a low efficiency.
  - Disable writers to make sure that the data is unchanged during data synchronization. For example, lock the table and disable data synchronization between primary and secondary databases. In this way, data is synchronized efficiently but your ongoing services may be interrupted.

- Character encoding

  Oracle Reader uses JDBC, which can automatically convert the encoding of characters. Therefore, you do not need to specify the encoding format.

- Incremental data synchronization

  Oracle Reader connects to a database through JDBC and uses a SELECT statement with a WHERE clause to read incremental data in the following ways:

  - For data in batches, incremental add, update, and delete operations (including logically delete operations) are distinguished by timestamps. Specify the WHERE clause based on the timestamp. The timestamp must be later than the latest timestamp in the last synchronization.
  - For streaming data, specify the WHERE clause based on the data record ID. The data record ID must be larger than the maximum ID involved in the last synchronization.

  If incremental data cannot be distinguished, Oracle Reader cannot perform incremental synchronization but can perform full synchronization only.

- Syntax validation

  Oracle Reader allows you to specify custom SELECT statements by using the querySql parameter but does not verify the syntax of the custom SELECT statements.

3.6.3.9. Configure OSS Reader

This topic describes the data types and parameters supported by OSS Reader and how to configure it by using the codeless UI and code editor.

OSS Reader can read data stored in OSS. OSS Reader connects to OSS by using the official OSS Java SDK, reads data from OSS, converts the data into a format that is readable by Data Integration, and then sends the converted data to a writer.

OSS stores unstructured data only. OSS Reader supports the following features:

- Reads TXT objects that store logical two-dimensional tables. OSS Reader can read only TXT objects.
- Reads data stored in formats similar to CSV with custom delimiters.
- Reads data of various types as strings and supports constants and column pruning.
- Supports recursive reading and object name-based filtering.
- Supports the following object compression formats: GZIP, BZIP2, and ZIP.

  > **Note** You cannot compress multiple objects into one package.

- Reads multiple objects concurrently.

OSS Reader does not support the following features:

- Uses concurrent threads to read an uncompressed object.
- Uses concurrent threads to read a compressed object.

OSS Reader supports the following OSS data types: Bigint, Double, String, Datatime, and Boolean.

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None

Parameter	Description	Required	Default value
Object	The name of the OSS object to read. You can specify multiple object names. For example, if a bucket has a directory named yunshi and this directory contains an object named ll.txt, you can set this parameter to yunshi/ll.txt. • If you specify a single OSS object, OSS Reader uses only one thread to read the object. Concurrent multi-thread reading of a single uncompressed object is coming soon. • If you specify multiple OSS objects, OSS Reader uses multiple threads to read these objects. The actual number of threads is determined by the number of channels. • When a name contains a wildcard, OSS Reader attempts to read all objects that match the name. For example, if you set the value to abc[0-9], OSS Reader reads objects abc0 to abc9. We recommend that you do not use wildcards because wildcards may cause out of memory (OOM). For more information, see OSS documentation. **Note** • Data Integration considers all the objects on a sync node as a single table. Make sure that all the objects on each sync node can adapt to the same schema. • Control the number of objects stored in a single directory. If a directory contains excessive objects, an OOM error may be returned. In this case, store the objects in different directories and then synchronize data.	Yes	None

Parameter	Description	Required	Default value
column	The columns to read. The type parameter specifies the source data type. The index parameter specifies the ID of the column in the source table, starting from 0. The value parameter specifies the column value if the column is a constant column. By default, OSS Reader reads all data as strings. You can specify the column parameter in the following way: `json` `"column": ["*"]` You can also specify the column parameter in the following way: `json` `"column":` `{` `"type": "long",` `"index": 0 // The first INT-type column of the source object.` `},` `{` `"type": "string",` `"value": "alibaba" // The value of the current column. In this case, the value is a constant "alibaba."` `}` **Note** For the column parameter, you must specify the type parameter and specify one of the index and value parameters.	Yes	By default, OSS Reader reads all data as strings.
fieldDelimiter	The column delimiter. **Note** You must specify the column delimiter for OSS Reader. The default delimiter is comma (,). The default setting for the column delimiter on the codeless UI is comma (,), too.	Yes	,
compress	The compression format of the object. By default, this parameter is left empty, indicating that objects are not compressed. OSS Reader supports the following object compression formats: GZIP, BZIP2, and ZIP.	No	*By default, objects are not compressed.*
encoding	The encoding format of the object to read.	No	*utf-8*

Parameter	Description	Required	Default value
nullFormat	The string that represents null. No standard strings can represent null in TXT objects. Therefore, Data Integration provides the nullFormat parameter to define which string represents a null pointer. For example, if you specify `nullFormat="null"`, Data Integration considers null as a null pointer. You can use the following formula to escape empty strings: \N=\\N.	No	None
skipHeader	Specifies whether to skip the header (if exists) of a CSV-like object. The skipHeader parameter is not supported for compressed objects.	No	*false*
csvReaderConfig	The configurations for reading CSV objects. The parameter value must match the MAP type. A specific CSV reader is used to read data from CSV objects, which supports many configurations.	No	None

Configure OSS Reader by using the codeless UI

1. Configure the connections.

   Configure the connections to the source and destination data stores for the sync node.

Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and then select a connection name that you have configured in DataWorks.
Object Name Prefix	The object parameter in the preceding parameter description. **Note** If an OSS object is named based on the date, for example, named as aaa/20171024abc.txt, you can set the object parameter to aaa/${bdp.system.bizdate}abc.txt.
Field Delimiter	The fieldDelimiter parameter in the preceding parameter description. The default delimiter is comma (,).
Encoding	The encoding parameter in the preceding parameter description. Default value: UTF-8.
Null String	The nullFormat parameter in the preceding parameter description. Enter a string that represents null. If the source connection contains the string, the string is replaced with null.
Compression Format	The compress parameter in the preceding parameter description. Default value: None.
Include Header	The skipHeader parameter in the preceding parameter description. Default value: No.

2. Configure field mapping. It is equivalent to setting the column parameter provided in the preceding

table.

Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right. You can click **Add** to add a field, or move the pointer over a field and click the **Delete** icon to delete the field.

Button	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.

3. Configure channel control policies.

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read data from and write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure OSS Reader by using the code editor

In the following code, a node is configured to read data from OSS. For more information about the parameters, see the preceding parameter description.

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"oss",// The reader type.
            "parameter":{
                "nullFormat":"",// The string that represents null.
                "compress":"",// The compression format.
                "datasource":"",// The connection name.
                "column":[// The columns to be synchronized.
```

```
                    {
                        "index":0,// The ID of the column in the source table.
                        "type":"string"// The data type.
                    },
                    {
                        "index":1,
                        "type":"long"
                    },
                    {
                        "index":2,
                        "type":"double"
                    },
                    {
                        "index":3,
                        "type":"boolean"
                    },
                    {
                        "format":"yyyy-MM-dd HH:mm:ss", // The format of the time.
                        "index":4,
                        "type":"date"
                    }
                ],
                "skipHeader":"", // Specifies whether to skip the header (if exists) of a C
SV-like object.
                "encoding":"",// The encoding format.
                "fieldDelimiter":",",// The column delimiter.
                "fileFormat": "",// The format of the object saved by OSS Reader.
                "Object":[]// The name of the OSS object to read.
            },
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"stream",
            "parameter":{},
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":""// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false,// Specifies whether to enable bandwidth throttling. A value o
f false indicates that the bandwidth is not throttled. A value of true indicates that the b
andwidth is throttled. The maximum transmission rate takes effect only if you set this para
meter to true.
            "concurrent":1,// The maximum number of concurrent threads.
        }
    },
    "order":{
        "hops":[
            {
```

```
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

3.6.3.10. Configure FTP Reader

This topic describes the data types and parameters supported by File Transfer Protocol (FTP) Reader and how to configure it by using the codeless user interface (UI) and code editor.

FTP Reader allows you to read data from a remote FTP server. FTP Reader connects to an FTP server, reads data from the server, converts the data into a format that is readable by Data Integration, and then sends the converted data to a writer.

FTP Reader can read only FTP files that store logical two-dimensional tables, for example, text information in CSV format.

FTP servers store unstructured data only. Currently, FTP Reader supports the following features:

- Reads TXT files that store logical two-dimensional tables. FTP Reader can read only TXT files.
- Reads data stored in formats similar to CSV with custom delimiters.
- Reads data of various types as strings. Supports constants and column pruning.
- Supports recursive reading and file name-based filtering.
- Supports the following file compression formats: GZIP, BZIP2, ZIP, LZO, and LZO_DEFLATE.
- Reads multiple files concurrently.

Currently, FTP Reader does not support the following features:

- Uses concurrent threads to read an uncompressed file.
- Uses concurrent threads to read a compressed file.

The data types of remote FTP files are defined by FTP Reader.

Data Integration data type	FTP file data type
LONG	LONG
DOUBLE	DOUBLE
STRING	STRING
BOOLEAN	BOOLEAN
DATE	DATE

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None
path	The path of the FTP file to read. You can specify multiple FTP file paths. • If you specify a single FTP file, FTP Reader uses only one thread to read the file. Concurrent multi-thread reading of a single uncompressed file is coming soon. • If you specify multiple FTP files, FTP Reader uses multiple threads to read these files. The actual number of threads is determined by the number of channels. • When a path contains a wildcard, FTP Reader attempts to read all files that match the path. If the path is ended with a slash (/), FTP Reader reads all files in the specified directory. For example, if you specify the path as /bazhen/, FTP Reader reads all files in the bazhen directory. Currently, FTP Reader only supports asterisks (*) as file name wildcards. **Note** • We recommend that you do not use asterisks (*) because this may cause out of memory (OOM) on a Java virtual machine (JVM). • Data Integration considers all the files on a sync node as a single table. Make sure that all the files on each sync node can adapt to the same schema and Data Integration has the permission to read all these files. • Make sure that the data format is similar to CSV. • An error occurs if no readable files exist in the specified path.	Yes	None

Parameter	Description	Required	Default value
column	The columns to read. The type parameter specifies the source data type. The index parameter specifies the ID of the column in the source table, starting from 0. The value parameter specifies the column value if the column is a constant column. By default, FTP Reader reads all data as strings. Specify this parameter as `"column":["*"]`. You can also specify the column parameter in the following way: `{ "type": "long", "index": 0 // The first INT-type column of the source file. }, { "type": "string", "value": "alibaba" // The value of the current column, that is, a constant "alibaba". }` For the column parameter, you must specify the type parameter and specify one of the index and value parameters.	Yes	By default, FTP Reader reads all data as strings.
fieldDelimiter	The column delimiter. **Note** You must specify the column delimiter for FTP Reader. The default delimiter is comma (,). The default setting for the column delimiter on the codeless UI is comma (,), too.	Yes	,
skipHeader	Specifies whether to skip the header (if exists) of a CSV-like file. The skipHeader parameter is not supported for compressed files.	No	false
encoding	The encoding format of the file to read.	No	*UTF-8*

Parameter	Description	Required	Default value
nullFormat	The string that represents null. No standard strings can represent null in text files. Therefore, Data Integration provides the nullFormat parameter to define which string represents a null pointer. For example, if you specify `nullFormat:"null"`, Data Integration considers null as a null pointer.	No	None
markDoneFileName	The name of the file used to indicate that the sync node can start. Data Integration checks whether the file exists before data synchronization. If the file does not exist, Data Integration checks again later. Data Integration starts the sync node only after the file is detected.	No	None
maxRetryTime	The maximum number of checks for the file used to indicate that the sync node can start. By default, 60 checks are allowed. Data Integration checks for the file every 1 minute. The whole process lasts at most 60 minutes.	No	*60*
csvReaderConfig	The configurations for reading CSV files. The parameter value must match the MAP type. A specific CSV reader is used to read data from CSV files, which supports many configurations.	No	None
fileFormat	The format of the file saved by FTP Reader. By default, FTP Reader converts the data into a two-dimensional table and stores the table in a CSV file. If you specify binary as the file format, Data Integration converts data into the binary format for replication and transmission. Generally, you need to specify this parameter only when you want to replicate the complete directory structure between storage systems such as FTP and Object Storage Service (OSS).	No	None

Configure FTP Reader by using the codeless UI

1. Configure the connections.

   Configure the source and destination connections for the sync node.

Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and enter the name of a connection that has been configured in DataWorks.
File Path	The path parameter in the preceding parameter description.
File Type	The format of the file saved by FTP Reader. The default format is CSV.
Field Delimiter	The fieldDelimiter parameter in the preceding parameter description. The default delimiter is comma (,).

Parameter	Description
Encoding	The encoding parameter in the preceding parameter description. The default encoding format is *UTF-8*.
Null String	The nullFormat parameter in the preceding parameter description, which defines a string that represents the null value.
Compression Format	The compression format. By default, files are not compressed.
Include Header	The skipHeader parameter in the preceding parameter description. The default value is *No*.

2. Configure field mapping, that is, the column parameter in the preceding parameter description.

   Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right. You can click **Add** to add a field, or move the pointer over a field and click the **Delete** icon to delete the field.

Parameter	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.

3. Configure channel control policies.

Parameter	Description
Expected Concurrency	The maximum number of concurrent threads to read data from or write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure FTP Reader by using the code editor

In the following code, a node is configured to read data from an FTP server.

```
{
    "type":"job",
    "version":"2.0", // The version number.
    "steps":[
        {
            "stepType":"ftp", // The reader type.
            "parameter":{
                "path":[], // The file path.
                "nullFormat":"", // The string that represents null.
                "compress":"", // The compression format.
                "datasource":"", // The connection name.
                "column":[ // The columns to be synchronized.
                    {
                        "index":0, // The ID of the column in the source table.
                        "type":"" // The data type.
                    }
                ],
                "skipHeader":"", // Specifies whether to skip the file header.
                "fieldDelimiter":",", // The column delimiter.
                "encoding":"UTF-8", // The encoding format.
                "fileFormat":"csv" // The format of the file saved by FTP Reader.
            },
            "name":"Reader",
            "category":"reader"
        },
        {// The following template is used to configure the writer. For more information, s
ee the corresponding topic.
            "stepType":"stream",
            "parameter":{},
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0" // The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false, // Specifies whether to enable bandwidth throttling. A value
of false indicates that the bandwidth is not throttled. A value of true indicates that the
bandwidth is throttled. The maximum transmission rate takes effect only if you set this par
ameter to true.
            "concurrent":1, // The maximum number of concurrent threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
```

```
        }
}
```

3.6.3.11. Configure Tablestore Reader

This topic describes the data types and parameters supported by Tablestore Reader and how to configure it by using the code editor.

Tablestore Reader can read incremental data from Tablestore based on the specified range. Tablestore Reader can read incremental data in the following ways:

- Reads data from the entire table.
- Reads data based on the specified range.
- Reads data from the specified shard.

Tablestore is a NoSQL database service that is built on the Apsara distributed operating system. The service allows you to store and access large volumes of structured data in real time. Tablestore organizes data into instances and tables. It uses data sharding and load balancing technologies to seamlessly expand the data scale.

Tablestore Reader connects to the Tablestore server by using Tablestore SDK for Java and reads data from the server. Then, Tablestore Reader converts the data into a format that is readable to Data Integration based on the official data synchronization protocols, and sends the converted data to a writer.

Tablestore Reader splits a synchronization node into multiple concurrent tasks based on the table range to synchronize data in a Tablestore table. Each Tablestore Reader thread runs a task.

Tablestore Reader supports all Tablestore data types. The following table lists the data types.

Category	Tablestore data type
Integer	INTEGER
Floating point	DOUBLE
String	STRING
Boolean	BOOLEAN
Binary	BINARY

Note Tablestore does not support DATE-type data. Applications use the LONG-type UNIX timestamp to indicate time.

Parameters

Parameter	Description	Required	Default value
endpoint	The endpoint of the Tablestore server.	Yes	None

Parameter	Description	Required	Default value
accessId	The AccessKey ID of the account that is used to access Tablestore.	Yes	None
accessKey	The AccessKey secret of the account that is used to access Tablestore.	Yes	None
instanceName	The name of the Tablestore instance. The instance is an entity for you to use and manage Tablestore. After you activate the Tablestore service, you must create an instance in the Tablestore console before you can create and manage tables. Instances are the basic units that you can use to manage Tablestore resources. Access control and resource measurement for applications are implemented at the instance level.	Yes	None
table	The name of the source table. You can specify only one table as the source table. Multi-table synchronization is not required for Tablestore.	Yes	None
column	The columns that you want to synchronize from the source table. The columns are described in a JSON array. Tablestore is a NoSQL database service. You must specify column names for Tablestore Reader to read data. • You can specify common columns. For example, you can specify {"name":"col1"} for Tablestore Reader to read data from column 1. • You can specify partial columns. Tablestore Reader reads data only from the specified columns. • You can specify constant columns. For example, you can specify {"type":"STRING", "value":"DataX"} for Tablestore Reader to read data from the column in which data is of the STRING type and the data value is DataX. The type parameter specifies the constant type. The supported types are STRING, INT, DOUBLE, Boolean, BINARY, INF_MIN, and INF_MAX. If the constant type is BINARY, the constant value must be Base64-encoded. INF_MIN indicates the minimum value specified by Tablestore, and INF_MAX indicates the maximum value specified by Tablestore. If you set the type to INF_MIN or INF_MAX, do not set the value. If you set the value, errors may occur. • You cannot specify a function or custom expression. This is because Tablestore does not provide functions or expressions that are similar to those of SQL. Tablestore Reader cannot read data from columns that contain functions or expressions.	Yes	None

Parameter	Description	Required	Default value
begin and end	The Tablestore table range from which you want to read data. You can specify both or neither of the two parameters. The begin and end parameters define the value ranges of primary key columns in the Tablestore table. Make sure that you specify the value ranges for all primary key columns in the table. If you do not need to limit a range, specify the parameters as {"type":"INF_MIN"} and {"type":"INF_MAX"}. For example, to read data from a Tablestore table with the primary key of [DeviceID, SellerID], specify the begin and end parameters in the following way: `"range": { "begin": [{"type":"INF_MIN"}, // The minimum value of the DeviceID field. {"type":"INT", "value":"0"} // The minimum value of the SellerID field.], "end": [{"type":"INF_MAX"}, // The maximum value of the DeviceID field. {"type":"INT", "value":"9999"} // The maximum value of the SellerID field.] }` To read all data from the table, specify the begin and end parameters in the following way: `"range": { "begin": [{"type":"INF_MIN"}, // The minimum value of the DeviceID field. {"type":"INF_MIN"} // The minimum value of the SellerID field.], "end": [{"type":"INF_MAX"}, // The maximum value of the DeviceID field. {"type":"INF_MAX"} // The maximum value of the SellerID field.] }`	Yes	None
	The custom rule for data sharding. This parameter is an advanced configuration item. We recommend that you do not set this parameter.		

Parameter	Description	Required	Default value
split	If data is unevenly distributed in a Tablestore table and the automatic sharding feature of Tablestore Reader fails to work, you can customize a sharding rule. The sharding rule that is specified by the split parameter must fall in the range that is specified by the begin and end parameters and must be the values of the partition key. This means that you specify only the values of the partition key instead of the values of primary key columns in the split parameter. To read data from a Tablestore table with the primary key of [DeviceID, SellerID], specify the following parameters: `"range": { "begin": { {"type":"INF_MIN"}, // The minimum value of the DeviceID field. {"type":"INF_MIN"} // The minimum value of the SellerID field. }, "end": { {"type":"INF_MAX"}, // The maximum value of the DeviceID field. {"type":"INF_MAX"} // The maximum value of the SellerID field. }, // The specified sharding rule. If you specify a sharding rule, the synchronization node is split into concurrent tasks based on the values of the begin, end, and split parameters. Data is sharded based only on the partition key, which the first column of the primary key. // The data type of the partition key can be INF_MIN, INF_MAX, STRING, or INT. "split":[{"type":"STRING", "value":"1"}, {"type":"STRING", "value":"2"}, {"type":"STRING", "value":"3"}, {"type":"STRING", "value":"4"}, {"type":"STRING", "value":"5"}] }`	No	None

Parameter	Description	Required	Default value

Configure Tablestore Reader by using the codeless UI

This method is not supported.

Configure Tablestore Reader by using the code editor

In the following code, a node is configured to read data from a Tablestore table:

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"ots",// The reader type.
            "parameter":{
                "datasource":"",// The data source.
                "column":[// The columns from which data is read.
                    {
                        "name":"column1"// The name of the column.
                    },
                    {
                        "name":"column2"
                    },
                    {
                        "name":"column3"
                    },
                    {
                        "name":"column4"
                    },
                    {
                        "name":"column5"
                    }
                ],
                "range":{
                    "split":[
                        {
                            "type":"INF_MIN"
                        },
                        {
                            "type":"STRING",
                            "value":"splitPoint1"
                        },
                        {
                            "type":"STRING",
                            "value":"splitPoint2"
                        },
                        {
                            "type":"STRING",
                            "value":"splitPoint3"
                        },
```

```
                        {
                            "type":"INF_MAX"
                        }
                    ],
                    "end":[
                        {
                            "type":"INF_MAX"
                        },
                        {
                            "type":"INF_MAX"
                        },
                        {
                            "type":"STRING",
                            "value":"end1"
                        },
                        {
                            "type":"INT",
                            "value":"100"
                        }
                    ],
                    "begin":[
                        {
                            "type":"INF_MIN"
                        },
                        {
                            "type":"INF_MIN"
                        },
                        {
                            "type":"STRING",
                            "value":"begin1"
                        },
                        {
                            "type":"INT",
                            "value":"0"
                        }
                    ]
                },
                "table":""// The name of the source table.
            },
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"stream",
            "parameter":{},
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0"// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false,// Specifies whether to enable bandwidth throttling. The value
```

```
            "throttle":false,// Specifies whether to enable bandwidth throttling. The value
false indicates that bandwidth throttling is disabled, and the value true indicates that ba
ndwidth throttling is enabled. The concurrent parameter takes effect only when the throttle
parameter is set to true.
            "concurrent":1,// The maximum number of concurrent threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

3.6.3.12. Configure PostgreSQL Reader

This topic describes the data types and parameters supported by PostgreSQL Reader and how to configure it by using the codeless UI and code editor.

PostgreSQL Reader connects to a remote PostgreSQL database and runs a SELECT statement to select and read data from the database. ApsaraDB for Relational Database Service (RDS) provides the PostgreSQL storage engine.

Specifically, PostgreSQL Reader connects to a remote PostgreSQL database through Java Database Connectivity (JDBC), generates a SELECT statement based on your configurations, and sends the statement to the database. The PostgreSQL database runs the statement and returns the result. Then, PostgreSQL Reader assembles the returned data to abstract datasets in custom data types supported by Data Integration, and passes the datasets to a writer.

- PostgreSQL Reader generates the SELECT statement based on the table, column, and where parameters that you have configured, and sends the generated SQL statement to the PostgreSQL database.
- If you specify the querySql parameter, PostgreSQL Reader directly sends the value of this parameter to the PostgreSQL database.

Data types

PostgreSQL Reader supports most PostgreSQL data types. Ensure that your data types are supported.

The following table lists the data types supported by PostgreSQL Reader.

Category	PostgreSQL data type
Integer	bigint, bigserial, integer, smallint, and serial
Float	double, precision, money, numeric, and real
String	varchar, char, text, bit, and inet
Date and time	date, time, and timestamp

Category	PostgreSQL data type
Boolean	boolean
Binary	bytea

> **Note**
> - Except for the preceding data types, other types are not supported.
> - You need to convert the money, inet, and bit types by using syntax such as `a_inet::varchar`.

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None
table	The name of the table to be synchronized.	Yes	None
column	An array of columns to be synchronized from the configured table, in JSON format. The default value is [*], which indicates all columns. • Column pruning is supported, which means that you can select and export specific columns. • Change of the column order is supported, which means that you can export the columns in an order different from that specified in the schema of the table. • Constants are supported. The column names must be arranged in compliance with SQL syntax supported by MySQL. For example, `["id", "table","1", "'mingya.wmy'", "'null'", "to_char(a+1)", "2.3", "true"]`. ○ id: a column name. ○ table: the name of a column that contains reserved keywords. ○ 1: an integer constant. ○ 'mingya.wmy': a string constant, which is enclosed in a pair of single quotation marks ('). ○ 'null': a string. ○ to_char(a + 1): a function expression. ○ 2.3: a float value. ○ true: a Boolean value. • The column parameter must explicitly specify a set of columns to be synchronized. It cannot be left empty.	Yes	None

Parameter	Description	Required	Default value
splitPk	The field used for data sharding when PostgreSQL Reader extracts data. If you specify the splitPk parameter, the table is sharded based on the shard key specified by this parameter. Data Integration then initiates concurrent data synchronization threads, which improves efficiency. • We recommend that you set the splitPk parameter to the primary key of the table. Based on the primary key, data can be well distributed to different shards, but not intensively distributed to certain shards. • Currently, the splitPk parameter supports data sharding only for integers but not for other data types such as string, float, and date. If you specify this parameter to a column of an unsupported type, PostgreSQL Reader ignores the splitPk parameter and synchronizes data through a single thread. • If you do not specify the splitPk parameter or leave it empty, Data Integration synchronizes data through a single thread.	No	None
where	The WHERE clause. PostgreSQL Reader generates a SELECT statement based on the table, column, and where parameters that you have configured, and uses the generated SELECT statement to select and read data. For example, set this parameter to `id>2 and sex=1`. • The WHERE clause can be used for synchronizing incremental data. • If you do not specify the where parameter or leave it empty, all data is synchronized.	No	None
querySql (only available in the code editor)	The SELECT statement used for refined data filtering. If you specify this parameter, Data Integration directly filters data based on this parameter. For example, if you need to join multiple tables for data synchronization, set this parameter to `select a,b from table_a join table_b on table_a.id = table_b.id`. If you specify the querySql parameter, PostgreSQL Reader ignores the table, column, and where parameters that you have configured.	No	None
fetchSize	The number of data records to read at a time. This parameter determines the number of interactions between Data Integration and the database and affects reading efficiency. **Note** A value larger than 2048 may lead to the out of memory (OOM) error during the data synchronization process.	No	*512*

Configure PostgreSQL Reader by using the codeless UI

1. Configure the connections.

   Configure the source and destination connections for the data synchronization node.

Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and enter the name of a connection that has been configured in DataWorks.
Table	The table parameter in the preceding parameter description.
Filter	The filter condition for the data to be synchronized. Currently, filtering based on the limit keyword is not supported. The SQL syntax is determined by the selected data store.
Shard Key	The shard key. You can use a column in the source table as the shard key. We recommend that you use the primary key or an indexed column. Only integer fields are supported. If data sharding is performed based on the configured shard key, data can be read concurrently to improve data synchronization efficiency. **Note** The Shard Key parameter is displayed only when you configure the source connection for a data synchronization node.

2. Configure field mapping (the column parameter in the preceding parameter description).

   Fields in the source table (left) have a one-to-one mapping with fields in the destination table (right). You can click **Add** to add a field or move the pointer over a field and click the **Delete** icon to delete a field.

Configuration item	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	The fields are automatically sorted based on specified rules.
Change Fields	You can manually edit fields in the source table. Each field occupies a row. The first and the last blank rows are included, while other blank rows are ignored.

Configuration item	Description
Add	◦ You can enter constants. Each constant must be enclosed in single quotation marks ('), such as 'abc' and '123'. ◦ You can use scheduling parameters, such as ${bizdate}. ◦ You can enter functions supported by relational databases, such as now() and count(1). ◦ If the value you entered cannot be parsed, the type is displayed as Unidentified.

3. Configure the channel.

Parameter	Description
DMU	The billing unit of Data Integration. **Note** Use caution while setting the DMU parameter. Its value limits the maximum number of concurrent threads.
Concurrent Threads	The maximum number of concurrent data synchronization threads. If you specify this parameter, the data records are split based on the shard key specified for the reader so that they are synchronized in multiple threads concurrently. This improves the transmission rate.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.
Resource Group	The resource group used for running the data synchronization node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the data synchronization node on the custom resource group. Set the resource group properly based on network conditions of the data stores, resource group usage, and business importance.

Configure PostgreSQL Reader by using the code editor

In the following code, a node is configured to read data from a PostgreSQL database.

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"postgresql",// The reader type.
            "parameter":{
                "datasource":"",// The connection name.
                "column":[// The columns to be synchronized.
                    "col1",
                    "col2"
                ],
                "where":"",// The WHERE clause.
                "splitPk":"",// The shard key based on which the table is sharded. Data Int
egration initiates concurrent threads to synchronize data.
                "table":""// The name of the table to be synchronized.
            },
            "name":"Reader",
            "category":"reader"
        },
        { // The following template is used to configure the writer. For more information,
see the document of the corresponding writer.
            "stepType":"stream",
            "parameter":{},
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0"// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false,// A value of false indicates that the bandwidth is not thrott
led. A value of true indicates that the bandwidth is throttled. The maximum transmission ra
te takes effect only if you set this parameter to true.
            "concurrent":1,// The maximum number of concurrent threads.
            "dmu":1// The DMU value.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

Additional instructions

- Data synchronization between primary and secondary databases

A secondary PostgreSQL database can be deployed for disaster recovery. The secondary database continuously synchronizes data from the primary database based on binlogs. Especially when network conditions are unfavorable, data latency between the primary and secondary databases is unavoidable, which can lead to data inconsistency.

- Concurrency control

  PostgreSQL is a relational database management system (RDBMS), which supports strong consistency for data queries. A database snapshot is created before a data synchronization node starts. PostgreSQL Reader reads data from the database snapshot. Therefore, if new data is written to the database during data synchronization, the reader cannot obtain the new data.

  Data consistency cannot be ensured when you enable PostgreSQL Reader to run concurrent threads on a single data synchronization node.

  PostgreSQL Reader shards the table based on the splitPk parameter and runs multiple concurrent threads to synchronize data. These concurrent threads belong to different transactions, and they read data at different times. This means that the concurrent threads observe different snapshots.

  Theoretically, the data inconsistency issue is unavoidable if a single data synchronization node includes multiple threads. However, two workarounds are available:

  - Do not enable concurrent threads on a single data synchronization node. Essentially, do not specify the splitPk parameter. In this way, data consistency is ensured while data is synchronized at a low efficiency.
  - Disable writers to ensure that the data is unchanged during data synchronization. For example, lock the table and disable data synchronization between primary and secondary databases. In this way, data is synchronized efficiently but your ongoing services can be interrupted.

- Character encoding

  A PostgreSQL database supports only EUC_CN and UTF-8 encoding formats for simplified Chinese characters. PostgreSQL Reader uses JDBC, which can automatically convert encoding of characters. Therefore, you do not need to specify the encoding.

  If data is written to the PostgreSQL database in an encoding format different from that specified by the PostgreSQL database, PostgreSQL Reader cannot recognize this inconsistency and may export garbled characters.

- Incremental data synchronization

  PostgreSQL Reader connects to a database through JDBC and uses `a SELECT statement with a WHERE clause` to read incremental data in either of the following ways:

  - For batch data, incremental add, update, and delete operations (including logical delete operations) are distinguished by timestamps. Specify the WHERE clause based on the timestamp. The timestamp must be later than the latest timestamp in the last synchronization.
  - For streaming data, specify the WHERE clause based on the data record ID. The data record ID must be larger than the maximum ID involved in the last synchronization.

  If incremental data cannot be distinguished, PostgreSQL Reader cannot perform incremental synchronization but can perform full synchronization only.

- Syntax validation

  PostgreSQL Reader allows you to specify custom SELECT statements by using the querySql parameter but does not verify the syntax of the custom SELECT statements.

3.6.3.13. Configure SQL Server Reader

This topic describes the data types and parameters supported by SQL Server Reader and how to configure it by using the codeless user interface (UI) and code editor.

SQL Server Reader connects to a remote SQL Server database and runs a SELECT statement to select and read data from the database.

Specifically, SQL Server Reader connects to a remote SQL Server database through Java Database Connectivity (JDBC), generates a SELECT statement based on your configurations, and then sends the statement to the database. The SQL Server database runs the statement and returns the result. Then, SQL Server Reader assembles the returned data to abstract datasets in custom data types supported by Data Integration, and passes the datasets to a writer.

- SQL Server Reader generates the SELECT statement based on the table, column, and where parameters that you have configured, and sends the generated SELECT statement to the SQL Server database.
- If you specify the querySql parameter, SQL Server Reader directly sends the value of this parameter to the SQL Server database.

SQL Server Reader supports most SQL Server data types. Make sure that your data types are supported.

The following table lists the data types supported by SQL Server Reader.

Category	SQL Server data type
Integer	bigint, int, smallint, and tinyint
Floating point	float, decimal, real, and numeric
String	char, nchar, ntext, nvarchar, text, varchar, nvarchar (max), and varchar (max)
Date and time	date, datetime, and time
Boolean	bit
Binary	binary, varbinary, varbinary (max), and timestamp

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None
table	The name of the table to be synchronized. You can select only one source table for each sync node.	Yes	None

Parameter	Description	Required	Default value
column	The columns to be synchronized from the source table. The columns are described in a JSON array. The default value is [*], which indicates all columns. • Column pruning is supported. You can select and export specific columns. • Change of the column order is supported. You can export the columns in an order different from that specified in the schema of the table. • Constants are supported. The column names must be arranged in compliance with the SQL syntax supported by MySQL, for example, `["id", "table","1", "'mingya.wmy'", "'null'", "to_char(a+1)", "2.3", "true"]`. ○ id: a column name. ○ table: the name of a column that contains reserved keywords. ○ 1: an integer constant. ○ 'mingya.wmy': a string constant, which is enclosed in single quotation marks (' '). ○ 'null': a string. ○ to_char(a + 1): a function expression. ○ 2.3: a floating-point constant. ○ true: a Boolean value. • The column parameter must explicitly specify a set of columns to be synchronized. The parameter cannot be left empty.	Yes	None
splitPk	The field used for data sharding when SQL Server Reader extracts data. If you specify the splitPk parameter, the table is sharded based on the shard key specified by this parameter. Data Integration then runs concurrent threads to synchronize data. This improves efficiency. • We recommend that you set the splitPk parameter to the primary key of the table. Based on the primary key, data can be well distributed to different shards, but not intensively distributed to certain shards. • Currently, the splitPk parameter supports data sharding only for integers but not for other data types such as string, floating point, and date. If you specify this parameter to a column of an unsupported type, SQL Server Reader returns an error.	No	None

Parameter	Description	Required	Default value
where	The WHERE clause. SQL Server Reader generates a SELECT statement based on the table, column, and where parameters that you have configured, and uses the generated SELECT statement to select and read data. For example, set this parameter to limit 10 during a test. For example, if you need to synchronize data generated on the current day, set this parameter to `gmt_create > $bizdate`. • You can use the WHERE clause to synchronize incremental data. • If you do not specify the where parameter or leave it empty, all data is synchronized.	No	None
querySql	The SELECT statement used for refined data filtering. Specify this parameter in the following format: `"querysql" : "SELECT statement",`. If you specify this parameter, Data Integration directly filters data based on this parameter. For example, if you want to join multiple tables for data synchronization, set this parameter to `select a,b from table_a join table_b on table_a.id = table_b.id`. If you specify the querySql parameter, SQL Server Reader ignores the table, column, and where parameters that you have configured.	No	None
fetchSize	The number of data records to read at a time. This parameter determines the number of interactions between Data Integration and the database and affects reading efficiency. **Note** A value larger than 2048 may lead to the out of memory (OOM) error during the data synchronization process.	No	*1024*

Configure SQL Server Reader by using the codeless UI

1. Configure the connections.

   Configure the source and destination connections for the sync node.

Configuration item	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and enter the name of a connection that has been configured in DataWorks.
Table	The table parameter in the preceding parameter description.
Filter	The filter condition for the data to be synchronized. Currently, filtering based on the limit keyword is not supported. The SQL syntax is determined by the selected connection.
Shard Key	The shard key. You can use a column in the source table as the shard key. We recommend that you use the primary key or an indexed column.

2. Configure field mapping, that is, the column parameter in the preceding parameter description.

   Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right. You can click **Add** to add a field, or move the pointer over a field and click the **Delete** icon to delete the field.

Configuration item	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	Click Auto Layout. The fields are automatically sorted based on specified rules.
Change Fields	Click the Change Fields icon. In the Change Fields dialog box that appears, you can manually edit fields in the source table. Each field occupies a row. The first and the last blank rows are included, whereas other blank rows are ignored.
Add	◦ Click Add to add a field. You can enter constants. Each constant must be enclosed in single quotation marks (' '), such as 'abc' and '123'. ◦ You can use scheduling parameters, such as ${bizdate}. ◦ You can enter functions supported by relational databases, such as now() and count(1). ◦ Fields that cannot be parsed are indicated by Unidentified.

3. Configure channel control policies.

Configuration item	Description
Expected Concurrency	The maximum number of concurrent threads to read data from or write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure SQL Server Reader by using the code editor

In the following code, a node is configured to read data from an SQL Server database.

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"sqlserver",// The reader type.
            "parameter":{
                "datasource":"",// The connection name.
                "column":[// The columns to be synchronized.
                    "id",
                    "name"
                ],
                "where":"",// The WHERE clause.
                "splitPk":"",// The shard key based on which the table is sharded.
                "table":""// The name of the table to be synchronized.
            },
            "name":"Reader",
            "category":"reader"
        },
        {// The following template is used to configure the writer. For more information, s
ee the corresponding topic.
            "stepType":"stream",
            "parameter":{},
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0"// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false,// Specifies whether to enable bandwidth throttling. A value o
f false indicates that the bandwidth is not throttled. A value of true indicates that the b
andwidth is throttled. The maximum transmission rate takes effect only if you set this para
meter to true.
            "concurrent":1,// The maximum number of concurrent threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

If you want to use the querySql parameter to specify a SELECT statement to query data, see the following sample code in the script of SQL Server Reader. Assume that the SQL Server connection is sql_server_source, the table to be queried is dbo.test_table, and the column to be queried is name.

```
{
    "stepType": "sqlserver",
    "parameter": {
        "querySql": "select name from dbo.test_table",
        "datasource": "sql_server_source",
        "column": [
            "name"
        ],
        "where": "",
        "splitPk": "id"
    },
    "name": "Reader",
    "category": "reader"
},
```

Additional instructions

- Data synchronization between primary and secondary databases

  A secondary SQL Server database can be deployed for disaster recovery. The secondary database continuously synchronizes data from the primary database based on binlogs. Especially when network conditions are unfavorable, data latency between the primary and secondary databases is unavoidable, which can lead to data inconsistency.

- Concurrency control

  SQL Server is a relational database management system (RDBMS), which supports strong consistency for data queries. A database snapshot is created before a sync node starts. SQL Server Reader reads data from the database snapshot. Therefore, if new data is written to the database during data synchronization, the reader cannot obtain the new data.

  Data consistency cannot be guaranteed when you enable SQL Server Reader to run concurrent threads on a single sync node.

  SQL Server Reader shards the table based on the splitPk parameter and runs multiple concurrent threads to synchronize data. These concurrent threads belong to different transactions. They read data at different time points. This means that the concurrent threads observe different snapshots.

  Theoretically, the data inconsistency issue is unavoidable if a single sync node includes multiple threads. However, two workarounds are available:

  - Do not enable concurrent threads on a single sync node. Essentially, do not specify the splitPk parameter. In this way, data consistency is guaranteed although data is synchronized at a low efficiency.
  - Disable writers to make sure that the data is unchanged during data synchronization. For example, lock the table and disable data synchronization between primary and secondary databases. In this way, data is synchronized efficiently but your ongoing services may be interrupted.

- Character encoding

  SQL Server Reader uses JDBC, which can automatically convert the encoding of characters. Therefore, you do not need to specify the encoding format.

- Incremental data synchronization

  SQL Server Reader connects to a database through JDBC and uses a SELECT statement with a WHERE clause to read incremental data in the following ways:

  - For data in batches, incremental add, update, and delete operations (including logically delete operations) are distinguished by timestamps. Specify the WHERE clause based on the timestamp. The timestamp must be later than the latest timestamp in the last synchronization.
  - For streaming data, specify the WHERE clause based on the data record ID. The data record ID must be larger than the maximum ID involved in the last synchronization.

  If incremental data cannot be distinguished, SQL Server Reader cannot perform incremental synchronization but can perform full synchronization only.

- Syntax validation

  SQL Server Reader allows you to specify custom SELECT statements by using the querySql parameter but does not verify the syntax of the custom SELECT statements.

3.6.3.14. Configure LogHub Reader

This topic describes the data types and parameters supported by LogHub Reader and how to configure it by using the codeless UI and code editor.

As an all-in-one real-time data logging service, Log Service provides features to collect, consume, deliver, query, and analyze log data. It can comprehensively improve the capabilities to process and analyze numerous logs. LogHub Reader consumes real-time log data in LogHub by using the Java SDK for Log Service, converts the data to a format that can be read by the Data Integration service, and sends the converted data to a writer.

How it works

LogHub Reader consumes real-time log data in LogHub by using the following version of the Java SDK for Log Service:

```
<dependency>
    <groupId>com.aliyun.openservices</groupId>
    <artifactId>aliyun-log</artifactId>
    <version>0.6.7</version>
</dependency>
```

In Log Service, Logstore is a basic unit for collecting, storing, and querying log data. The read and write logs of a Logstore are stored in a shard. Each Logstore consists of several shards, each of which is defined by a left-closed and right-open interval of MD5 values so that intervals do not overlap each other. The range of all intervals covers all the allowed MD5 values. Each shard can independently provide some services.

- Write: 5 Mbit/s, 2,000 times/s.
- Read: 10 Mbit/s, 100 times/s.

LogHub Reader consumes log data in shards by following this process in which the GetCursor and BatchGetLog API operations are called:

- Obtain a cursor based on the time range.
- Read logs based on the cursor and step parameters and return the next cursor.

- Keep moving the cursor to consume logs.
- Split the node to concurrent threads based on shards.

Data types

The following table lists the data types supported by LogHub Reader.

Data Integration data type	LogHub data type
STRING	STRING

Parameters

Parameter	Description	Required	Default value
endpoint	The Log Service endpoint, which is a URL for accessing a project and log data. It varies based on the Alibaba Cloud region where the project resides and the project name.	Yes	None
accessId	The AccessKey ID for connecting to Log Service.	Yes	None
accessKey	The AccessKey secret for connecting to Log Service.	Yes	None
project	The name of the project. A project is the basic unit for managing resources in Log Service. You can exercise access control at the project level, and isolate resources among different projects.	Yes	None
logstore	The name of the Logstore. A Logstore is the basic unit for collecting, storing, and querying log data in Log Service.	Yes	None
batchSize	The number of entries queried from Log Service at a time.	No	*128*
column	The column name in each log entry. You can configure a column that stores metadata in a source table of LogHub in such a way that the metadata in this column is inserted into the destination table. Supported metadata includes the log topic, unique identifier of the collection machine, host name, path, and log time. **Note** The column name is case-sensitive.	Yes	None

Parameter	Description	Required	Default value
beginDateTime	The start time of data consumption. The value is the time when log data arrives at LogHub. This parameter defines the left boundary of a left-closed and right-open interval in the format of yyyyMMddHHmmss, for example, 20180111013000. The parameter can work with the scheduling time parameter in DataWorks. **Note** The beginDateTime and endDateTime parameters must be used in pairs.	You must specify either beginDateTime or beginTimestampMillis, but not both.	None
endDateTime	The end time of data consumption in the format of yyyyMMddHHmmss, such as 20180111013010. This parameter defines the right boundary of a left-closed and right-open interval and can work with the scheduling time parameter in DataWorks. **Note** Make sure that the time specified by the endDateTime parameter of the previous interval is the same as or later than the time specified by the beginDateTime parameter of the current interval. If the intervals do not overlap, data may fail to be read in some regions.	You must specify either endDateTime or endTimestampMillis, but not both.	None
beginTimestampMillis	The start time of data consumption. This parameter specifies the left boundary of the left-closed and right-open interval, measured in milliseconds. **Note** The beginTimestampMillis and endTimestampMillis parameters must be used in pairs. A value of -1 indicates the position where the cursor starts in Log Service, which is specified by CursorMode.BEGIN. We recommend that you specify the beginDateTime parameter.	You must specify either beginTimestampMillis or beginDateTime, but not both.	None

Parameter	Description	Required	Default value
endTimestampMillis	The end time of data consumption, measured in milliseconds. This parameter defines the right boundary of the left-closed and right-open interval. **Note** The endTimestampMillis and beginTimestampMillis parameters must be used in pairs. A value of -1 indicates the position where the cursor ends in Log Service, which is specified by CursorMode.END. We recommend that you specify the endDateTime parameter.	You must specify either endTimestampMillis or endDateTime, but not both.	None

Configure LogHub Reader in the codeless UI

1. Configure the connections.

   Configure the connections to the source and destination data stores for the sync node.

Parameter	Description
Connection	The type and name of the source connection. Select a connection that you have configured in DataWorks.
Logstore	The name of the Logstore from which data is read.
Start Timestamp	The start time of data consumption. The value is the time when log data arrives at LogHub. This parameter defines the left boundary of a left-closed and right-open interval in the format of yyyyMMddHHmmss, for example, 20180111013000. The parameter can work with the scheduling time parameter in DataWorks.
End Timestamp	The end time of data consumption in the format of yyyyMMddHHmmss, such as 20180111013010. This parameter defines the right boundary of a left-closed and right-open interval and can work with the scheduling time parameter in DataWorks.
Records per Batch	The number of entries queried from Log Service at a time.

2. Configure field mapping. It is equivalent to setting the column parameter provided in the preceding parameter description.

   Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right. You can click **Add** to add a field, or move the pointer over a field and click the **Delete** icon to delete the field.

GUI element	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	Click Auto Layout to sort the fields based on specified rules.
Change Fields	Click the Change Fields icon. In the Change Fields dialog box, you can manually edit fields in the source table. Each field occupies a row. The first and the last blank rows are included, whereas other blank rows are ignored.
Add	○ Click Add to add a field. You can enter constants. Each constant must be enclosed in single quotation marks (' '), such as 'abc' and '123'. ○ You can use scheduling parameters, such as ${bizdate}. ○ You can enter functions supported by relational databases, such as now() and count(1). ○ Fields that cannot be parsed are indicated by Unidentified.

3. Configure channel control policies.

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read data from and write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure LogHub Reader by using the code editor

In the following code, a node is configured to read data from LogHub. For more information about the parameters, see the preceding parameter description.

```
{
 "type":"job",
```

```
 "version":"2.0",// The version number.
 "steps":[
     {
         "stepType":"loghub",// The reader type.
         "parameter":{
             "datasource":"",// The connection name.
             "column":[// The columns to be synchronized.
                 "col0",
                 "col1",
                 "col2",
                 "col3",
                 "col4",
                 "=Topic",// The log topic.
                 "HostName",// The hostname.
                 "Path",// The path.
                 "LogTime"// The log time.
             ],
             "beginDateTime":"",// The start time of data consumption.
             "batchSize":"",// The number of entries that are queried from Log Service at a
time.
             "endDateTime":",",// The end time of data consumption.
             "fieldDelimiter":",",// The column delimiter.
             "encoding":"UTF-8",// The encoding format.
             "logstore":"// The name of the target Logstore.
         },
         "name":"Reader",
         "category":"reader"
     },
     {
         "stepType":"stream",
         "parameter":{},
         "name":"Writer",
         "category":"writer"
     }
 ],
 "setting":{
     "errorLimit":{
         "record":"0"// The maximum number of dirty data records allowed.
     },
     "speed":{
         "throttle":false, // Specifies whether to enable bandwidth throttling. A value of
false indicates that the bandwidth is not throttled. A value of true indicates that the ban
dwidth is throttled. The maximum transmission rate takes effect only if you set this parame
ter to true.
         "concurrent":1,// The maximum number of concurrent threads.
     }
 },
 "order":{
     "hops":[
         {
             "from":"Reader",
             "to":"Writer"
         }
     ]
```

```
  }
}
```

Note If the metadata in JSON format is prefixed by tag, delete the tag prefix. For example, change `__tag__:__client_ip__` to `__client_ip__`.

3.6.3.15. Configure Tablestore Reader-Internal

This topic describes the data types and parameters supported by Tablestore Reader-Internal and how to configure it by using the code editor.

Tablestore is a NoSQL database service built on the Apsara distributed operating system that allows you to store and access large amounts of structured data in real time. Tablestore organizes data into instances and tables. It can seamlessly expand the data scale by using data sharding and load balancing technologies.

Tablestore Reader-Internal is used to export data for the Tablestore Internal model, whereas Tablestore Reader is used to export data for the Tablestore Public model.

Tablestore Reader-Internal can export data in multi-version mode or normal mode:

- Multi-version mode: Tablestore stores multiple versions of column values, and this mode allows you to export data of multiple versions.

  Tablestore Reader-Internal converts a cell to a 4-tuple of a one-dimensional table: PrimaryKey (columns 1 to 4), ColumnName, Timestamp, and Value. This process is similar to that for the multi-version mode of HBase Reader. Each {PrimaryKey, ColumnName, Timestamp, Value} tuple is sent to a writer as four columns in Data Integration records.
- Normal mode: This mode allows you to export the latest version of each column in each row, which is the same as the normal mode of HBase Reader. For more information, see the normal mode of HBase Reader in Configure an HBase connection.

Tablestore Reader-Internal connects to a Tablestore server by using the official Java SDK for Tablestore and reads data from the server. Tablestore Reader-Internal optimizes the read process by providing features such as performing retry attempts when a timeout or exception occurs.

Tablestore Reader-Internal supports all Tablestore data types. The following table lists the data types supported by Tablestore Reader-Internal.

Data Integration data type	Tablestore data type
LONG	INTEGER
DOUBLE	DOUBLE
STRING	STRING
BOOLEAN	BOOLEAN
BYTES	BINARY

Parameters

Parameter	Description	Required	Default value
mode	The mode in which Tablestore Reader-Internal exports data. Valid values: *normal* and *multiVersion*.	Yes	None
endpoint	The endpoint of the Tablestore server.	Yes	None
accessId	The AccessKey ID for connecting to Tablestore.	Yes	None
accessKey	The AccessKey secret for connecting to Tablestore.	Yes	None
instanceName	The name of the Tablestore instance. The instance is an entity for you to use and manage Tablestore. After you activate the Tablestore service, you must create an instance in the console before you create and manage tables. Instances are the basic units for managing Tablestore resources. All access control and resource measurement for applications are implemented at the instance level.	Yes	None
table	The name of the source table. You can specify only one table as the source table. Multi-table synchronization is not required for Tablestore.	Yes	None
range	The range of the data to export, in the format of [begin,end). • If the value of the begin parameter is smaller than that of the end parameter, data is read in forward order. • If the value of the begin parameter is larger than that of the end parameter, data is read in reverse order. • The value of the begin parameter cannot be the same as that of the end parameter. • The following value types are supported: STRING, INT, and BINARY. Binary data is passed in as Base64 strings in binary format. INF_MIN represents an infinitely small value and INF_MAX represents an infinitely large value.	No	By default, data is read from the beginning of the table to the end of the table.

Parameter	Description	Required	Default value
range: {"begin"}	The start of the data to export. Enter an empty array, a primary key prefix, or a complete primary key. In forward order, the default primary key suffix is INF_MIN. In reverse order, the default primary key suffix is INF_MAX. This parameter specifies the value range of the Tablestore primary key and is used for data filtering. If you do not specify this parameter, the minimum value is used by default. The JSON format does not support binary data. If the data type of the PrimaryKey column is BINARY, you must use the Java method Base64.encodeBase64String to convert binary data to a string, and then enter the string as the value of the parameter. Example: • `byte[] bytes = "hello".getBytes();` : constructs binary data, which is the byte value of the string hello. • `String inputValue = Base64.encodeBase64String(bytes)` : calls the Base64.encodeBase64String method to convert the binary data to a string. After you run the preceding code, the string `"aGVsbG8="` is returned for the inputValue parameter. Finally, set this parameter to `{"type":"binary","value":"aGVsbG8="}` .	No	Data is read from the beginning of the table.
range: {"end"}	The end of the data to export. Enter an empty array, a primary key prefix, or a complete primary key. In forward order, the default primary key suffix is INF_MAX. In reverse order, the default primary key suffix is INF_MIN. The JSON format does not support binary data. If the data type of the PrimaryKey column is BINARY, you must use the Java method Base64.encodeBase64String to convert binary data to a string, and then enter the string as the value of the parameter. Example: • `byte[] bytes = "hello".getBytes();` : constructs binary data, which is the byte value of the string hello. • `String inputValue = Base64.encodeBase64String(bytes)` : calls the Base64.encodeBase64String method to convert the binary data to a string. After you run the preceding code, the string `"aGVsbG8="` is returned for the inputValue parameter. Finally, set this parameter to `{"type":"binary","value":"aGVsbG8="}` .	No	Data is read until the end of the table.

Parameter	Description	Required	Default value
range: {"split"}	If an excessively large amount of data needs to be exported, you can specify this parameter to split one node to multiple concurrent threads. **Note** • The field based on which the node is split must be the shard key, which is the first column of the primary key, and the data type of the field must be the same as that of the partition key. • The specified field must be within the value range that is specified by the begin and end parameters. • The values of this field must be sorted in the descending or ascending order based on the data reading order that is determined by values of the begin and end parameters.	No	No sharding rule is specified.
column	The columns to be exported. Both regular and constant columns can be exported. A regular column is in the format of `{"name":"{your column name}"}`. **Note** • Constant columns cannot be exported in multi-version mode. • You cannot specify the PrimaryKey column. The exported tuple data contains the complete primary key by default. • Each column can be exported only once.	None	All versions of all columns are exported.
timeRange (applicable only to the multi-version mode)	The time range of the requested data, in the format of [begin,end). **Note** The value of the begin parameter must be smaller than that of the end parameter.	No	All the data is read.
timeRange: {"begin"} (applicable only to the multi-version mode)	The start time for reading data. Valid values: 0 to LONG_MAX.	No	0

Parameter	Description	Required	Default value
timeRange: {"end"} (applicable only to the multi-version mode)	The end time for reading data. Valid values: 0 to LONG_MAX.	No	*LONG_MAX (922337203685477 5806L)*
maxVersion (applicable only to the multi-version mode)	The specified version of the requested data. Valid values: 1 to INT32_MAX.	No	The data of all versions is read.

Configure Tablestore Reader-Internal by using the codeless UI

The codeless UI is not supported for Tablestore Reader-Internal.

Configure Tablestore Reader-Internal by using the code editor

- Multi-version mode

```
{
    "type": "job",
    "version": "1.0",
    "configuration": {
        "reader": {
            "plugin": "otsreader-internalreader",
            "parameter": {
                "mode": "multiVersion",
                "endpoint": "",
                "accessId": "",
                "accessKey": "",
                "instanceName": "",
                "table": "",
                "range": {
                    "begin": [
                        {
                            "type": "string",
                            "value": "a"
                        },
                        {
                            "type": "INF_MIN"
                        }
                    ],
                    "end": [
                        {
                            "type": "string",
                            "value": "g"
                        },
                        {
                            "type": "INF_MAX"
                        }
```

```
                    ],
                    "split": [
                        {
                            "type": "string",
                            "value": "b"
                        },
                        {
                            "type": "string",
                            "value": "c"
                        }
                    ]
                },
                "column": [
                    {
                        "name": "attr1"
                    }
                ],
                "timeRange": {
                    "begin": 1400000000,
                    "end": 1600000000
                },
                "maxVersion": 10
            }
        }
    },
    "writer": {}
}
```

- Normal mode

```
{
    "type": "job",
    "version": "1.0",
    "configuration": {
        "reader": {
            "plugin": "otsreader-internalreader",
            "parameter": {
                "mode": "normal",
                "endpoint": "",
                "accessId": "",
                "accessKey": "",
                "instanceName": "",
                "table": "",
                "range": {
                    "begin": [
                        {
                            "type": "string",
                            "value": "a"
                        },
                        {
                            "type": "INF_MIN"
                        }
                    ],
                    "end": [
```

```
                        {
                            "type": "string",
                            "value": "g"
                        },
                        {
                            "type": "INF_MAX"
                        }
                    ],
                    "split": [
                        {
                            "type": "string",
                            "value": "b"
                        },
                        {
                            "type": "string",
                            "value": "c"
                        }
                    ]
                },
                "column": [
                    {
                        "name": "pk1"
                    },
                    {
                        "name": "pk2"
                    },
                    {
                        "name": "attr1"
                    },
                    {
                        "type": "string",
                        "value": ""
                    },
                    {
                        "type": "int",
                        "value": ""
                    },
                    {
                        "type": "double",
                        "value": ""
                    },
                    {
                        "type": "binary",
                        "value": "aGVsbG8="
                    }
                ]
            }
        }
    },
    "writer": {}
}
```

3.6.3.16. Configure OTSStream Reader

This topic describes the data types and parameters supported by OTSStream Reader and how to configure it by using the code editor.

OTSStream Reader is mainly used to synchronize the incremental data of Tablestore. Incremental data can be considered as operation logs that include data and operation information.

Unlike plug-ins used to synchronize full data, OTSStream Reader supports only the multi-version mode. When you use OTSStream Reader to synchronize incremental data, you cannot synchronize the data of specified columns, which is related to the principle of synchronizing incremental data. The following section describes the implementation process.

Before you use OTSStream Reader, make sure that the Stream feature is enabled for your table. You can enable this feature when you create the table, or you can use the UpdateTable operation in the SDK to enable this feature.

The following example describes how to enable the Stream feature:

```
SyncClient client = new SyncClient("", "", "", "");
Enable this feature when you create a table.
CreateTableRequest createTableRequest = new CreateTableRequest(tableMeta);
createTableRequest.setStreamSpecification(new StreamSpecification(true, 24));// The value 2
4 indicates that the incremental data is retained for 24 hours.
client.createTable(createTableRequest);
If this feature is not enabled when you create a table, enable it by using the UpdateTable
operation.
UpdateTableRequest updateTableRequest = new UpdateTableRequest("tableName");
updateTableRequest.setStreamSpecification(new StreamSpecification(true, 24));
client.updateTable(updateTableRequest);
```

How it works

You can enable the Stream feature and set the expiration time by using the UpdateTable operation in the SDK. After the Stream feature is enabled, the Tablestore server additionally saves your operation logs. Each partition has a sequential operation log queue. Each operation log is recycled after your specified expiration time.

The Tablestore SDK provides several Stream APIs that are used to read these operation logs. OTSStream Reader obtains incremental data by using these APIs, transforms the incremental data into multiple six-tuples (pk, colName, version, colValue, opType, and sequenceInfo), and then synchronizes them into MaxCompute.

Format of the synchronized data

In the multi-version model of Tablestore, table data is organized in a three-level mode: row, column, and version. One row can have multiple columns, and the column name is not fixed. Each column can have multiple versions, and each version has a specific timestamp (the version number).

You can perform read or write operations by using Tablestore APIs. Tablestore records the incremental data by recording the recent write and modification operations on table data. Therefore, incremental data can be considered as a set of operation records.

Tablestore supports the following three types of modification operations:

- PutRow: writes a row. If the row already exists, it is overwritten.
- UpdateRow: updates a row without the need to change other data of the original row. You can add column values, overwrite column values if the related version of the column already exists, delete all

the versions of a column, or delete a version of a column.

- DeleteRow: deletes a row.

Tablestore generates incremental data records based on each type of operation. OTSStream Reader reads these records and synchronizes the data in the format supported by DataX.

Tablestore supports dynamic columns and the multi-version mode. Therefore, a row exported by OTSStream Reader corresponds to a version of a column rather than a row in Tablestore. A row in Tablestore may correspond to multiple synchronized rows. Each synchronized row includes the primary key value, column name, timestamp of the version for the column (version number), value of the version, and operation type. If the isExportSequenceInfo parameter is set to true, time series information is also included.

When the data is transformed into the format supported by DataX, the following four types of operations are defined:

- U (UPDATE): writes a version of a column.
- DO (DELETE_ONE_VERSION): deletes a version of a column.
- DA (DELETE_ALL_VERSION): deletes all the versions of a column. Delete all the versions of the column based on the primary key and the column name.
- DR (DELETE_ROW): deletes a row. Delete all the data of the row based on the primary key.

In the following example, the table has two primary key columns: pkName1 and pkName2.

pkName1	pkName2	columnName	timestamp	columnValue	opType
pk1_V1	pk2_V1	col_a	1441803688001	col_val1	U
pk1_V1	pk2_V1	col_a	1441803688002	col_val2	U
pk1_V1	pk2_V1	col_b	1441803688003	col_val3	U
pk1_V2	pk2_V2	col_a	1441803688000	-	DO
pk1_V2	pk2_V2	col_b	-	-	DA
pk1_V3	pk2_V3	-	-	-	DR
pk1_V3	pk2_V3	col_a	1441803688005	col_val1	U

In this example, seven rows are synchronized, which corresponds to three rows in the Tablestore table. The primary keys for the three rows are (pk1_V1, pk2_V1), (pk1_V2, pk2_V2), and (pk1_V3, pk2_V3).

- For the row whose primary key is (pk1_V1, pk2_V1), three operations are included: writing two versions of column col_a and one version of column col_b.
- For the row whose primary key is (pk1_V2, pk2_V2), two operations are included: deleting one version of column col_a and all the versions of column col_b.
- For the row whose primary key is (pk1_V3, pk2_V3), two operations are included: deleting the row and writing one version of column col_a.

Data types

OTSStream Reader supports all Tablestore data types. The following table lists the data types supported by OTSStream Reader.

Category	OTSStream data type
Integer	INTEGER
Floating point	DOUBLE
String	STRING
Boolean	BOOLEAN
Binary	BINARY

Parameters

Parameter	Description	Required	Default value
dataSource	The name of the data source. It must be the same as the added data source. You can add data sources by using the code editor.	Yes	No default value
dataTable	The name of the table from which incremental data is synchronized. You must enable the Stream feature for a table when you create the table, or you can call the UpdateTable operation to enable this feature.	Yes	No default value

Parameter	Description	Required	Default value
statusTable	The name of the table that OTSStream Reader uses to store status records. These records help find the data that is not required and improve synchronization efficiency. A status table is used to store status records. If no such table exists, OTSStream Reader automatically creates one. After the running of an offline export task is completed, you do not need to delete the table. The status records in the table can be used for the next export task. • You need only to provide a table name rather than manually creating a status table. OTSStreamReader attempts to create a status table under your instance. If no such table exists, OTSStream Reader automatically creates one. If the table already exists, OTSStream Reader determines whether the metadata of the table meets the expectation. If not, an error is reported. • After the running of an export task is completed, you do not need to delete the table. The status records in the table can be used for the next export task. • The table enables Time To Live (TTL), and data automatically expires, which indicates that the data volume is small. • You can use the same status table to store the status records of the multiple tables that are specified by the dataTable parameter and managed by the same instance. The status records are independent of each other. In conclusion, you can configure a name similar to TableStoreStreamReaderStatusTable. You must make sure that the name is inconsistent with that of a business-related table.	Yes	No default value
startTimestampMillis	The start time (included) of the incremental data, in milliseconds. • OTSStream Reader finds a point that corresponds to the time specified by the startTimestampMillis parameter from the status table, and starts to read and synchronize data from this point. • If OTSStream Reader cannot find the required point, it starts to read incremental data retained by the system from the first entry, and skips the data which is written later than the time specified by startTimestampMillis.	No	No default value
endTimestampMillis	The end time (excluded) of the incremental data, in milliseconds. • OTSStream Reader exports data from the time specified by the startTimestampMillis parameter and stops exporting data when the timestamp of a data record is later than or equal to the time specified by the endTimestampMillis parameter. • After all the incremental data is read, OTSStream Reader stops reading data even before the time specified by the endTimestampMillis parameter.	No	No default value

Parameter	Description	Required	Default value
date	The date on which data is synchronized. Specify this parameter in the yyyyMMdd format, such as 20151111. You must specify either the date parameter or the startTimestampMillis and endTimestampMillis parameters. For example, Alibaba Cloud Data Process Center performs scheduling only at the day level. Therefore, the date parameter is provided.	No	No default value
isExportSequenceInfo	Specifies whether to synchronize time series information which includes the time when data is written. The default value is *false*, which indicates that time series information is not synchronized.	No	No default value
maxRetries	The maximum number of retries for each request of reading incremental data from Tablestore. The default value is 30. Retries are performed at specific intervals. The total time of 30 retries is about 5 minutes. You can keep the default settings.	No	No default value
startTimeString	The start time (included) of the incremental data, in milliseconds. Specify this parameter in the `yyyymmddhh24miss` format.	No	No default value
endTimeString	The end time (excluded) of the incremental data, in milliseconds. Specify this parameter in the `yyyymmddhh24miss` format.	No	No default value
mode	The synchronization mode. If this parameter is set to single_version_and_update_only, data is exported by row. By default, this parameter is not specified, and data is not synchronized by column.	No	No default value

Configure OTSStream Reader by using the codeless UI

This method is not supported.

Configure OTSStream Reader by using the code editor

The following example shows how to configure a synchronization node to read the incremental data of Tablestore. For more information about the parameters, see the preceding parameter description.

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"otsstream",// The reader type.
            "parameter":{
                "statusTable":"TableStoreStreamReaderStatusTable",// The name of the table
that OTSStream Reader uses to store status records.
                "maxRetries":30,// The maximum number of retries on each request of reading
incremental data from Tablestore. It is set to 30 by default.
                "isExportSequenceInfo":false,// Specifies whether to synchronize the time s
eries information.
                "datasource":"$srcDatasource",// The name of the data source.
                "startTimeString":"${startTime}",// The start time (included) of the increm
ental data.
                "table":"",// The name of the table from which you want to read data.
                "endTimeString":"${endTime}"// The end time (excluded) of the incremental d
ata.
            },
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"stream",
            "parameter":{},
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0"// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false,// Specifies whether to enable bandwidth throttling. The value
false indicates that bandwidth throttling is disabled, and the value true indicates that ba
ndwidth throttling is enabled. The mbps parameter takes effect only when the throttle param
eter is set to true.
            "concurrent":1// The maximum number of parallel threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

3.6.3.17. Configure RDBMS Reader

This topic describes the data types and parameters supported by RDBMS Reader and how to configure it by using the code editor.

Background information

RDBMS Reader allows you to read data from an RDBMS database. RDBMS Reader connects to a remote RDBMS database and runs a SELECT statement to select and read data from the database. RDBMS Reader can read data from databases such as Dameng, Db2, PPAS, and Sybase databases. If you need RDBMS Reader to read data from a common relational database, register the driver for the corresponding database type.

RDBMS Reader connects to a remote RDBMS database by using JDBC, generates a SELECT statement based on your configurations, and then sends the statement to the database. The RDBMS database runs the statement and returns the result. Then, RDBMS Reader assembles the returned data to abstract datasets in custom data types supported by Data Integration, and passes the datasets to a writer.

- RDBMS Reader generates the SQL statement based on the table, column, and where parameters that you have configured, and sends the generated SQL statement to the RDBMS database.
- If you specify the querySql parameter, RDBMS Reader directly sends the value of this parameter to the RDBMS database.

RDBMS Reader supports most data types of a common relational database, such as numbers and characters. Make sure that your data types are supported.

Parameters

Parameter	Description	Required	Default value
	The JDBC URL for connecting to the RDBMS database. The format must be in accordance with the official RDBMS specifications. You can also specify the information of the attachment facility. The format varies based on the database type. Data Integration selects an appropriate driver for data reading based on the format. • Format for DM databases: `jdbc:dm://ip:port/database` • Format for Db2 databases: `jdbc:db2://ip:port/database` • Format for PPAS databases: `jdbc:edb://ip:port/database` You can enable RDBMS Reader to support a new database by using the following method: • Go to the RDBMS Reader directory. In the directory, ${DATAX_HOME} indicates the main directory of Data Integration. • Open the *plugin.json* file in the RDBMS Reader directory, and add the driver of your database to the drivers array in the file. RDBMS Reader dynamically selects the appropriate database driver to connect to the database when nodes are run.		

Parameter	Description	Required	Default value
jdbcUrl	{ "name": "rdbmsreader", "class": "com.alibaba.datax.plugin.reader.rdbmsreader.RdbmsReader", "description": "useScene: prod. mechanism: Jdbc connection using the database, execute select sql, retrieve data from the ResultSet. warn: The more you know about the database, the less problems you encounter.", "developer": "alibaba", "drivers": ["dm.jdbc.driver.DmDriver", "com.ibm.db2.jcc.DB2Driver", "com.sybase.jdbc3.jdbc.SybDriver", "com.edb.Driver"] } ``` - Add the driver package to the libs directory in the RDBMS Reader directory. ``` $tree . |-- libs | |-- Dm7JdbcDriver16.jar | |-- commons-collections-3.0.jar | |-- commons-io-2.4.jar | |-- commons-lang3-3.3.2.jar | |-- commons-math3-3.1.1.jar | |-- datax-common-0.0.1-SNAPSHOT.jar | |-- datax-service-face-1.0.23-20160120.024328-1.jar | |-- db2jcc4.jar | |-- druid-1.0.15.jar | |-- edb-jdbc16.jar | |-- fastjson-1.1.46.sec01.jar | |-- guava-r05.jar | |-- hamcrest-core-1.3.jar | |-- jconn3-1.0.0-SNAPSHOT.jar | |-- logback-classic-1.0.13.jar | |-- logback-core-1.0.13.jar | |-- plugin-rdbms-util-0.0.1-SNAPSHOT.jar | `-- slf4j-api-1.7.10.jar |-- plugin.json |-- plugin_job_template.json `-- rdbmsreader-0.0.1-SNAPSHOT.jar	Yes	None
username	The username for connecting to the database.	Yes	None
password	The password for connecting to the database.	Yes	None
table	The name of the source table.	Yes	None

Parameter	Description	Required	Default value
column	The columns to be synchronized from the source table. The columns are described in a JSON array. The default value is [*], which indicates all columns. • Column pruning is supported. You can select specific columns to export. • The column order can be changed. You can export the specified columns in an order different from that specified in the schema of the table. • Constants are supported. The column names must be arranged in JSON format, for example, `["id","1", "'bazhen.csy'", "null", "to_char(a + 1)", "2.3" , "true"]`. ○ id: a column name. ○ 1: an integer constant. ○ 'bazhen.csy': a string constant. ○ null: a null pointer. ○ to_char(a + 1): a function expression. ○ 2.3: a floating-point constant. ○ true: a Boolean value. • The column parameter must explicitly specify a set of columns to be synchronized, and cannot be left empty.	Yes	None
splitPk	The field used for data sharding when RDBMS Reader reads data. If you specify the splitPk parameter, the table is sharded based on the shard key specified by this parameter. Data Integration then runs concurrent threads to synchronize data. This improves efficiency. • We recommend that you set the splitPk parameter to the primary key of the table. Based on the primary key, data can be well distributed to different shards, but not intensively distributed to specific shards. • The splitPk parameter supports data sharding only for integers but not for other data types such as string, floating point, and date. If you specify this parameter to a column of an unsupported type, RDBMS Reader returns an error. • If you do not specify the splitPk parameter or leave it empty, RDBMS Reader synchronizes data by using a single thread.	No	An empty string
where	The WHERE clause. RDBMS Reader generates a SELECT statement based on the table, column, and where parameters that you have configured, and uses the generated SELECT statement to select and read data. For example, set this parameter to limit 10. To synchronize data generated on the current day, set the where parameter to `gmt_create > $bizdate`. • You can use the WHERE clause to read incremental data. • If you do not specify the where parameter or leave it empty, all data is read.	No	None

Parameter	Description	Required	Default value
querySql	The SELECT statement used to for refined data filtering. If you specify this parameter, Data Integration directly filters data based on this parameter. For example, if you need to join multiple tables for data synchronization, set this parameter to `select a,b from table_a join table_b on table_a.id = table_b.id`. If you specify the querySql parameter, RDBMS Reader ignores the table, column, and where parameters that you have configured.	No	None
fetchSize	The number of data records to read at a time. This parameter determines the number of interactions between Data Integration and the database and affects reading efficiency. **Note** A value greater than 2048 may lead to OOM during the data synchronization process.	No	*1,024*

Configure RDBMS Reader by using the codeless UI

The codeless UI is not supported for RDBMS Reader.

Configure RDBMS Reader by using the code editor

In the following code, a node is configured to read data from an RDBMS database.

```
{
    "order": {
        "hops": [
            {
                "from": "Reader",
                "to": "Writer"
            }
        ]
    },
    "setting": {
        "errorLimit": {
            "record": "0"
        },
        "speed": {
            "concurrent": 1,
            "throttle": false
        }
    },
    "steps": [
        {
            "category": "reader",
            "name": "Reader",
            "parameter": {
                "connection": [
                    {
```

```
                        "jdbcUrl": [
                            "jdbc:dm://ip:port/database"
                        ],
                        "table": [
                            "table"
                        ]
                    }
                ],
                "username": "username",
                "password": "password",
                "table": "table",
                "column": [
                    "*"
                ],
                "preSql": [
                    "delete from XXX;"
                ]
            },
            "stepType": "rdbms"
        },
        {
            "category": "writer",
            "name": "Writer",
            "parameter": {},
            "stepType": "stream"
        }
    ],
    "type": "job",
    "version": "2.0"
}
```

3.6.3.18. Configure Stream Reader

This topic describes the data types and parameters supported by Stream Reader and how to configure it by using the code editor.

Stream Reader automatically generates data from the memory. It is mainly used for performance testing for data synchronization and basic functional testing.

The following table lists the data types supported by Stream Reader.

Data type	Description
String	A sequence of characters.
Long	A long integer.
Date	A value that represents dates.
Boolean	A Boolean data type that has one of two possible values.
Bytes	An 8-bit signed two's complement integer.

Parameters

Parameter	Description	Required	Default value
column	The column data and type of the source data. Multiple columns can be configured. You can set to generate random strings and specify the range. The example is as follows: `"column" : [ { "random": "8,15" }, { "random": "10,10" } ]` The parameters are described as follows: • "random": "8, 15": generates a random string that is 8 to 15 bytes in length. • "random": "10, 10": generates a 10-byte random string.	Yes	None
sliceRecordCount	The number of columns generated repeatedly.	Yes	None

Configure Stream Reader by using the codeless UI

Currently, the codeless user interface (UI) is not supported for Stream Reader.

Configure Stream Reader by using the code editor

In the following code, a node is configured to read data from the memory and then write the data to Stream Reader.

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"stream",// The reader type.
            "parameter":{
                "column":[// The columns to be synchronized.
                    {
                        "type":"string",// The data type.
                        "value":"field" // The value.
                    },
                    {
                        "type":"long",
                        "value":100
                    },
                    {
                        "dateFormat":"yyyy-MM-dd HH:mm:ss",// The format of the time.
                        "type":"date"
```

```
                        "type":"date",
                        "value":"2014-12-12 12:12:12"
                    },
                    {
                        "type":"bool",
                        "value":true
                    },
                    {
                        "type":"bytes",
                        "value":"byte string"
                    }
                ],
                "sliceRecordCount":"100000"// The number of columns repeatedly generated.
            },
            "name":"Reader",
            "category":"reader"
        },
        {// The following template is used to configure the writer. For more information, s
ee the corresponding topic.
            "stepType":"stream",
            "parameter":{},
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0"// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false,// Specifies whether to enable bandwidth throttling. A value o
f false indicates that the bandwidth is not throttled. A value of true indicates that the b
andwidth is throttled. The maximum transmission rate takes effect only if you set this para
meter to true.
            "concurrent":1,// The maximum number of concurrent threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

3.6.3.19. Configure Hive Reader

Parameters

Parameter	Description	Required	Default value
column	The fields to read. Example: `"column": ["id", "name"]`.	Yes	None
table	The name of the Hive table to read. The name is case sensitive.	Yes	None
partition	The partition information of the table to read. The last-level partition must be specified. For example, if you want to read data from a three-level partition table, set this parameter to a value that contains the last-level partition information, such as pt=20150101/type=1/biz=2.	Yes	None

Configure Hive Reader by using the code editor

In the following code, a node is configured to read data from a Hive data store.

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"stream",
            "parameter":{},
            "name":"Writer",
            "category":"writer"
        },
        {
            "stepType":"hive", // The reader type. The name is the same as that in MaxCompu
te.
            "parameter":{
                "parameter":{
                    "column": [// The columns to be synchronized.
                        "id",
                        "name"
                    ],
                    "table": "student_tmp_2", // The name of the table to be synchronized.
                    "partition": "academy=yx/class=001", // The partition settings.
                    "datasource": "hive_demo"
                }
            },
            "name":"Reader",
            "category":"reader"
        }
    ],
    "setting":{
    },
    "order":{
        "hops":[// Synchronize data from the reader to the writer.
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

3.6.3.20. Configure Elasticsearch Reader

This topic describes the working principles, features, and parameters of Elasticsearch Reader.

Working principles

- Elasticsearch Reader reads data from Elasticsearch by slicing scroll queries. The slices are processed by multiple threads of a data synchronization node.
- Data types are converted based on the mapping configuration of Elasticsearch.

Basic settings

```
{
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    },
    "setting":{
        "errorLimit":{
            "record":"0" // The maximum number of dirty data records allowed.
        },
        "jvmOption":"",
        "speed":{
            "concurrent":3,
            "throttle":false
        }
    },
    "steps":[
        {
            "category":"reader",
            "name":"Reader",
            "parameter":{
                "column":[ // The fields to read.
                    "id",
                    "name"
                ],
                "endpoint":"", // The endpoint.
                "index":"",  // The index name.
                "password":"",  // The password.
                "scroll":"",  // The scroll ID.
                "search":"",  // The search criteria. The value is the same as the Elastics
earch query that uses the _search API.
                "type":"default",
                "username":""  // The username.
            },
            "stepType":"elasticsearch"
        },
        {
            "category":"writer",
            "name":"Writer",
            "parameter":{ },
            "stepType":"stream"
        }
    ],
    "type":"job",
    "version":"2.0" // The version number.
}
```

Advanced features

- Supports storing all data of an Elasticsearch document in one column.

You can create a column to store all data of an Elasticsearch document.

- Supports converting semi-structured data to structured data.

Item	Description
Background	Data in Elasticsearch is deeply nested. Elasticsearch may contain fields of various types and lengths and may use Chinese names. To facilitate data computing and storage in downstream businesses, Elasticsearch Reader supports converting semi-structured data to structured data.
Principle	Elasticsearch Reader flattens nested JSON data obtained from Elasticsearch to single-dimensional data based on the paths of properties in the JSON data. Then, Elasticsearch Reader maps the single-dimensional data to structured tables. In this way, Elasticsearch data in a complex structure is converted to multiple structured tables.
Solution	◦ Elasticsearch Reader converts nested JSON data to single-dimensional data by using the following path formats: ▪ Property ▪ Property.Child property ▪ Property[0].Child property ◦ If a property has multiple child properties, Elasticsearch Reader traverses all data of the property and splits the data to multiple tables or multiple rows in the following format: Property[*].Child property ◦ Elasticsearch Reader merges data in a string array to one property in the following format and removes duplicates: Property[] where duplicates are removed ◦ Elasticsearch Reader merges multiple properties to one property in the following format: Property 1,Property 2 ◦ Elasticsearch Reader presents optional properties in the following format: Property 1|Property 2

Parameters

Parameter	Description	Required	Default value
endpoint	The endpoint of Elasticsearch.	Yes	None
username	The username for HTTP authentication.	No	Empty string
password	The password for HTTP authentication.	No	Empty string

Parameter	Description	Required	Default value
index	The index name in Elasticsearch.	Yes	None
type	The type name in the index of Elasticsearch.	No	Index name
pageSize	The number of data records to read at a time.	No	*100*
search	The query parameter of Elasticsearch.	Yes	None
scroll	The scroll parameter of Elasticsearch, which sets the timestamp of the snapshot taken for a scroll.	Yes	None
sort	The field based on which the returned results are sorted.	No	None
retryCount	The number of retries after a failure.	No	*300*
connTimeOut	The connection timeout of the client.	No	*600,000*
readTimeOut	The data reading timeout of the client.	No	*600,000*
multiThread	Specifies whether to use multiple threads for an HTTP request.	No	*true*
column	The fields to read.	Yes	None
full	Specifies whether to create a column to record all data of an Elasticsearch document.	No	*false*
multi	Specifies whether to split an array to multiple rows. If you enable this feature, you need to specify additional settings.	No	*false*

Additional settings:

```
"full":false,
        "multi": {
          "multi": true,
          "key":"crn_list[*]"
        }
```

3.6.3.21. Configure Vertica Reader

Vertica is a column-oriented database using the Massively Parallel Processing (MPP) architecture. Vertica Reader allows you to read data from Vertica. This topic describes how Vertica Reader works, the supported parameter, and how to configure it by using the code editor.

How it works

Vertica Reader connects to a remote Vertica database by using JDBC and executes a SELECT statement to select and read data from the database.

Vertica Reader connects to a remote Vertica database by using JDBC, generates a SELECT statement based on your configurations, and then sends the statement to the database. The Vertica database executes the statement and returns the result. Then, Vertica Reader assembles the returned data to abstract datasets in custom data types supported by Data Integration, and sends the datasets to a writer.

- Vertica Reader generates the SELECT statement based on the table, column, and where parameters that you have configured, and sends the generated SELECT statement to the Vertica database.
- If you specify the querySql parameter, Vertica Reader directly sends the value of this parameter to the Vertica database.

Vertica Reader accesses a Vertica database by using the Vertica database driver. Confirm the compatibility between the driver version and your Vertica database. Vertica Reader uses the following version of the Vertica database driver:

```
<dependency>
    <groupId>com.vertica</groupId>
    <artifactId>vertica-jdbc</artifactId>
    <version>7.1.2</version>
</dependency>
```

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None

Parameter	Description	Required	Default value
jdbcUrl	The JDBC URL for connecting to the Vertica database. You can specify multiple JDBC URLs for a database. The JDBC URLs are described in a JSON array. If you specify multiple JDBC URLs, Vertica Reader verifies the connectivity of the URLs in sequence to find a valid URL. If no URL is valid, Vertica Reader returns an error. **Note** The jdbcUrl parameter must be included in the connection parameter. The value of the jdbcUrl parameter must be in compliance with the standard format supported by Vertica. You can also specify the information of the attachment facility. Example: `jdbc:vertica://1**.0.0.1:3306/database` .	No	None
username	The username for connecting to the Vertica database.	No	None
password	The password for connecting to the Vertica database.	No	None
table	The name of the source table from which Vertica Reader reads data. Vertica Reader can read data from multiple tables. The tables are described in a JSON array. If you specify multiple tables, make sure that the tables have the same schema. Vertica Reader does not check whether the tables have the same schema. **Note** The table parameter must be included in the connection parameter.	Yes	None
column	The columns to be synchronized from the source table. The columns are described in a JSON array. The default value is [*], which indicates all columns in the source table. • Column pruning is supported. You can select specific columns to export. • The column order can be changed. You can export the specified columns in an order different from that specified in the schema of the table. • Constants are supported. • The column parameter must explicitly specify a set of columns to be synchronized, and cannot be left empty.	Yes	None

Parameter	Description	Required	Default value
splitPk	The field used for data sharding when Vertica Reader reads data. If you specify the splitPk parameter, the table is sharded based on the shard key specified by this parameter. Data Integration then runs concurrent threads to synchronize data. This improves efficiency. • We recommend that you set the splitPk parameter to the primary key of the table. Based on the primary key, data can be well distributed to different shards, but not intensively distributed to specific shards. • The splitPk parameter supports data sharding only for integers but not for other data types such as string, floating point, and date. If you set this parameter to a column of an unsupported type, Vertica Reader returns an error. • If you leave the splitPk parameter empty, Vertica Reader reads data from the source table by using a single thread.	No	None
where	The WHERE clause. Vertica Reader generates a SELECT statement based on the table, column, and where parameters that you have configured, and uses the generated SELECT statement to select and read data. For example, you can specify the where parameter during testing. To synchronize data generated on the current day, set the where parameter to `gmt_create > $bizdate` . • You can use the WHERE clause to synchronize incremental data. • If you do not specify the where parameter or leave it empty, all data is read.	No	None
querySql	The SELECT statement used for refined data filtering. If you specify this parameter, Data Integration directly filters data based on this parameter. If you specify the querySql parameter, Vertica Reader ignores the table, column, and where parameters that you have configured.	No	None
fetchSize	The number of data records to read at a time. This parameter determines the number of interactions between Data Integration and the database and affects data reading efficiency. **Note** A value greater than 2048 may lead to OOM during the data synchronization process.	No	*1024*

Configure Vertica Reader by using the codeless UI

The codeless UI is not supported for Vertica Reader.

Configure Vertica Reader by using the code editor

In the following code, a node is configured to read data from a Vertica database.

```
{
    "type": "job",
    "steps": [
        {
            "stepType":"vertica", // The reader type.
             "parameter": {
                "datasource": "", // The connection name.
                 "username": "",
                "password": "",
                "where": "",
                "column": [ // The columns to be synchronized.
                     "id",
                    "name"
                ],
                "splitPk": "id",
                "connection": [
                    {
                        "table": [ // The name of the table to be synchronized.
                             "table"
                        ],
                        "jdbcUrl": [
                            "jdbc:vertica://host:port/database"
                        ]
                    }
                ]
            },
            "name": "Reader",
            "category": "reader"
        },
        {
            "stepType": "stream",
            "parameter": {
                "print": false,
                "fieldDelimiter": ","
            },
            "name": "Writer",
            "category": "writer"
        }
    ],
    "version": "2.0",
    "order": {
        "hops": [
            {
                "from": "Reader",
                "to": "Writer"
            }
        ]
    },
    "setting": {
```

```
        "errorLimit": {
            "record":"0" // The maximum number of dirty data records allowed.
         },
        "speed": {
            "throttle": false, // Specifies whether to enable bandwidth throttling. A value
of false indicates that the bandwidth is not throttled. A value of true indicates that the
bandwidth is throttled. The maximum transmission rate takes effect only if you set this par
ameter to true.
             "concurrent": 1 // The maximum number of concurrent threads.
         }
    }
}
```

3.6.3.22. Configure GBase Reader

This topic describes how GBase Reader reads data and how to configure a sync node to read data from a GBase database.

GBase Reader connects to a remote GBase database through the MySQL Java Database Connectivity (JDBC) Driver, generates SQL statements based on your configurations, and then reads data from the remote GBase database. Then, GBase Reader assembles the returned data to abstract datasets in custom data types supported by Data Integration, and passes the datasets to a writer.

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None
table	The name of the table to be synchronized. You can select only one source table for each sync node.	Yes	None

Parameter	Description	Required	Default value
column	The columns to be synchronized from the source table. The columns are described in a JSON array. The default value is [*], which indicates all columns. • Column pruning is supported. You can select and export specific columns. • Change of the column order is supported. You can export the columns in an order different from that specified in the schema of the table. • Constants are supported. The column names must be arranged in compliance with the SQL syntax supported by MySQL, for example, `["id","table","1","'mingya.wmy'","'null'","to_char(a+1)","2.3","true"]`. ○ id: a column name. ○ table: the name of a column that contains reserved keywords. ○ 1: an integer constant. ○ 'mingya.wmy': a string constant, which is enclosed in single quotation marks (' '). ○ null: ■ '' '' indicates an empty value. ■ null indicates a null value. ■ 'null' indicates the string null. ○ to_char(a+1): a function expression. ○ 2.3: a floating-point constant. ○ true: a Boolean value. • The column parameter must explicitly specify a set of columns to be synchronized. The parameter cannot be left empty.	Yes	None

Parameter	Description	Required	Default value
splitPk	The field used for data sharding when GBase Reader extracts data. If you specify the splitPk parameter, the table is sharded based on the shard key specified by this parameter. Data Integration then runs concurrent threads to synchronize data. This improves efficiency. • We recommend that you set the splitPk parameter to the primary key of the table. Based on the primary key, data can be well distributed to different shards, but not intensively distributed to certain shards. • Currently, the splitPk parameter supports data sharding only for integers but not for other data types such as string, floating point, and date. If you specify this parameter to a column of an unsupported type, GBase Reader ignores the splitPk parameter and synchronizes data through a single thread. • If you do not specify the splitPk parameter or leave it empty, Data Integration synchronizes data through a single thread.	No	None
where	The WHERE clause. For example, set this parameter to `gmt_create>$bizdate`. • You can use the WHERE clause to synchronize incremental data. If you do not specify the where parameter or leave it empty, all data is synchronized. • Do not set the where parameter to limit 10, which does not conform to the constraints of MySQL on the SQL WHERE clause.	No	None
querySql (only available in the code editor)	The SELECT statement used for refined data filtering. If you specify this parameter, Data Integration directly filters data based on this parameter. For example, if you want to join multiple tables for data synchronization, set this parameter to `select a,b from table_a join table_b on table_a.id = table_b.id`. The priority of the querySql parameter is higher than those of the table, column, where, and splitPk parameters. If you specify the querySql parameter, GBase Reader ignores the table, column, where, and splitPk parameters that you have configured. The datasource parameter parses information, including the username and password, from this parameter.	No	None

Configure GBase Reader by using the codeless UI

Currently, the codeless user interface (UI) is not supported for GBase Reader.

Configure GBase Reader by using the code editor

In the following code, a node is configured to read data from a GBase database.

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"gbase // The reader type.
            "parameter":{
                "column":[// The columns to be synchronized.
                    "id"
                ],
                "connection":[
                    {   "querysql":["select a,b from join1 c join join2 d on c.id = d.id;"]
, // Specify the querySql parameter in the connection parameter as a string.
                        "datasource":"",// The connection name.
                        "table":[// The name of the table to be synchronized.
                            "xxx"
                        ]
                    }
                ],
                "where":"",// The WHERE clause.
                "splitPk":"",// The shard key.
                "encoding":"UTF-8"// The encoding format.
            },
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"stream",
            "parameter":{},
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0"// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false,// Specifies whether to enable bandwidth throttling. A value o
f false indicates that the bandwidth is not throttled. A value of true indicates that the b
andwidth is throttled. The maximum transmission rate takes effect only if you set this para
meter to true.
            "concurrent":1,// The maximum number of concurrent threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
```

```
                }
            ]
        }
}
```

3.6.3.23. KingbaseES Reader

This topic describes the data types and parameters that are supported by KingbaseES Reader and how to configure KingbaseES Reader by using the codeless user interface (UI) and code editor. Before you create a Data Integration node, you can refer to this topic to familiarize yourself with the data types and parameters that you must configure for KingbaseES Reader to read data from data sources.

Context

KingbaseES Reader connects to a remote KingbaseES database by using Java Database Connectivity (JDBC), generates an SQL statement based on your configurations, and then sends the statement to the database. The system executes the statement on the database and returns data. Then, KingbaseES Reader assembles the returned data into abstract datasets of the data types supported by Data Integration and sends the datasets to a writer.

Data types

The following table lists the data types that are supported by KingbaseES Reader.

Data type	The data type of SAP HANA
Integer	INT, TINYINT, SMALLINT, MEDIUMINT, and BIGINT
Floating point	FLOAT, DOUBLE, and DECIMAL
String	VARCHAR, CHAR, TINYTEXT, TEXT, MEDIUMTEXT, and LONGTEXT
Date and time	DATE, DATETIME, TIMESTAMP, TIME, and YEAR
Boolean	BIT and BOOL
Binary	TINYBLOB, MEDIUMBLOB, BLOB, LONGBLOB, and VARBINARY

Notice

- Data types that are not listed in the preceding table are not supported.
- KingbaseES Reader processes TINYINT(1) as an integer data type.

Parameters

Parameter	Description
datasource	The name of the data source.

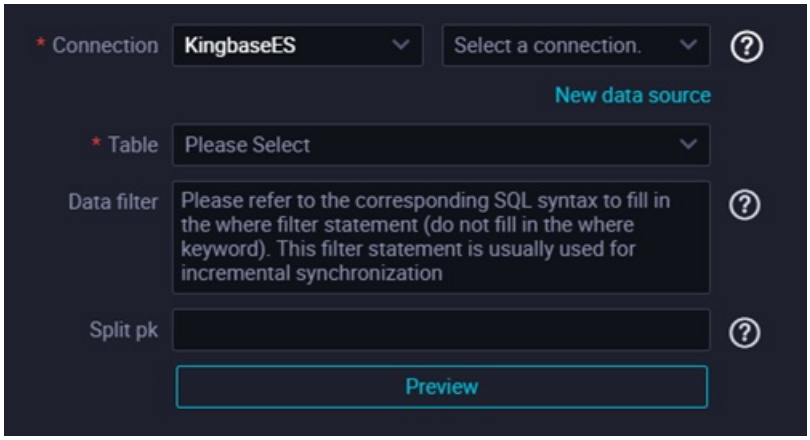
Parameter	Description
column	The names of the columns from which you want to read data. If you want to read data from all the columns in the source table, set this parameter to an asterisk (*).
table	The name of the source table.
splitPk	The field that is used for data sharding when KingbaseES Reader reads data. If you specify this parameter, the source table is sharded based on the value of this parameter. Data Integration then runs parallel threads to read data. You can specify a field of an integer data type for the splitPk parameter. If the source table does not contain fields of integer data types, you can leave this parameter empty.

Configure KingbaseES Reader by using the codeless UI

1. Configure data sources.

   Log on to the DataWorks console. The **DataStudio** page appears. On the DataStudio page, move the pointer over the icon and choose **Data Integration > Batch Synchronization**. In the **Create Node** dialog box, configure the parameters to create a batch synchronization node.

   Configure **Source** and Target for the synchronization node.

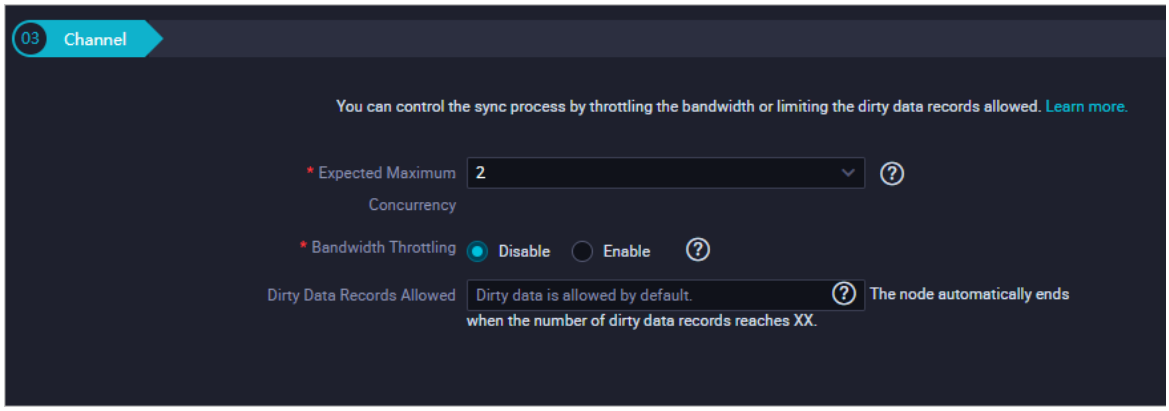
   

2. Configure field mappings. This operation is equivalent to setting the column parameter that is described in the preceding section.

Fields in the source on the left have a one-to-one mapping with fields in the destination on the right. You can click **Add** to add a field. To remove an added field, move the pointer over the field and click the **Remove** icon.

Operation	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish mappings between fields with the same name. The data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish mappings between fields in the same row. The data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove the mappings that are established.
Auto Layout	Click Auto Layout. Then, the system automatically sorts the fields based on specific rules.
Change Fields	Click the Change Fields icon. In the Change Fields dialog box, you can manually edit the fields in the source table. Each field occupies a row. The first and the last blank rows are included, whereas other blank rows are ignored.
Add	Click **Add** to add a field. Take note of the following rules when you add a field: ◦ You can enter constants. Each constant must be enclosed in single quotation marks ('), such as 'abc' and '123'. ◦ You can use scheduling parameters, such as ${bizdate}. ◦ You can enter functions that are supported by relational databases, such as now() and count(1). ◦ If the field that you entered cannot be parsed, the value of Type for the field is Unidentified.

3. Configure channel control policies.



Parameter	Description
Expected Maximum Concurrency	The maximum number of parallel threads that the synchronization node uses to read data from the source or write data to the destination. You can configure the parallelism for the synchronization node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workloads on the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure KingbaseES Reader by using the code editor

The following examples show how to configure KingbaseES Reader to read data from a database or table that is not sharded and how to configure KingbaseES Reader to read data from a database or table that is sharded.

- Configure KingbaseES Reader to read data from a database or table that is not sharded

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"kingbasees",// The reader type.
            "parameter":{
                "column":[// The names of the columns from which you want to read data.
                    "id"
                ],
                "connection":[
                    {   "querySql":["select a,b from join1 c join join2 d on c.id = d.id;
"], // The SQL statement that is used to read data from the source table.
                        "datasource":"",// The name of the data source.
                        "table":[// The name of the source table. The table name must be
enclosed in brackets [].
                            "xxx"
                        ]
                    }
```

```
                ],
                "where":"",// The WHERE clause.
                "splitPk":"",// The shard key.
                "encoding":"UTF-8"// The encoding format.
            },
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"stream",
            "parameter":{},
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0"// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false,// Specifies whether to enable bandwidth throttling. The val
ue false indicates that bandwidth throttling is disabled, and the value true indicates th
at bandwidth throttling is enabled. The mbps parameter takes effect only when the throttl
e parameter is set to true.
            "concurrent":1,// The maximum number of parallel threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

- Configure KingbaseES Reader to read data from a database or table that is sharded

  > **Note** When you configure a synchronization node to read data from a sharded database or table, you can select multiple KingbaseES tables with the same schema.

```
{
    "type": "job",
    "version": "1.0",
    "configuration": {
        "reader": {
            "plugin": "kingbasees",
            "parameter": {
                "connection": [
                    {
                        "table": [
                            "tbl1",
                            "tbl2",
                            "tbl3"
                        ],
                        "datasource": "datasourceName1"
                    },
                    {
                        "table": [
                            "tbl4",
                            "tbl5",
                            "tbl6"
                        ],
                        "datasource": "datasourceName2"
                    }
                ],
                "singleOrMulti": "multi",
                "splitPk": "db_id",
                "column": [
                    "id", "name", "age"
                ],
                "where": "1 < id and id < 100"
            }
        },
        "writer": {
        }
    }
}
```

3.6.3.24. SAP HANA Reader

This topic describes the data types and parameters that are supported by SAP HANA Reader and how to configure SAP HANA Reader by using the codeless user interface (UI) and code editor. Before you create a Data Integration node, you can refer to this topic to familiarize yourself with the data types and parameters that you must configure for SAP HANA Reader to read data from data sources.

Context

SAP HANA Reader connects to a remote SAP HANA database by using Java Database Connectivity (JDBC), generates an SQL statement based on your configurations, and then sends the statement to the database. The system executes the statement on the database and returns data. Then, SAP HANA Reader assembles the returned data into abstract datasets of the data types supported by Data Integration and sends the datasets to a writer.

Data types

The following table lists the data types that are supported by SAP HANA Reader.

Category	SAP HANA data type
Integer	INT, TINYINT, SMALLINT, MEDIUMINT, and BIGINT
Floating point	FLOAT, DOUBLE, and DECIMAL
String	VARCHAR, CHAR, TINYTEXT, TEXT, MEDIUMTEXT, and LONGTEXT
Date and time	DATE, DATETIME, TIMESTAMP, TIME, and YEAR
Boolean	BIT and BOOLEAN
Binary	TINYBLOB, MEDIUMBLOB, BLOB, LONGBLOB, and VARBINARY

> **Notice**
>
> - Data types that are not listed in the preceding table are not supported.
> - SAP HANA Reader processes TINYINT(1) as an integer data type.

Parameters

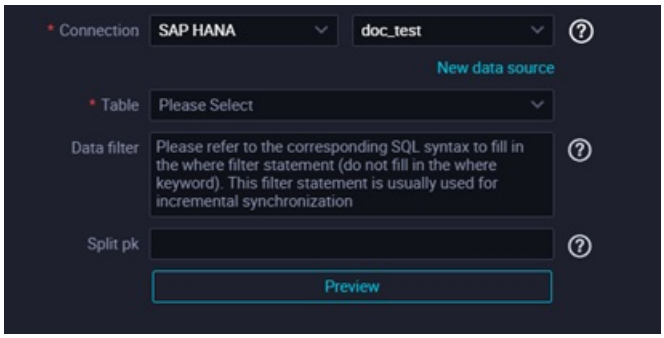
Parameter	Description
datasource	The name of the data source from which you want to read data. If no data source is available, click **Add data source** to add a data source.
column	The names of the columns from which you want to read data. If you want to read data from all the columns in the source table, set this parameter to an asterisk (*).
table	The name of the source table.
splitPk	The field that is used for data sharding when SAP HANA Reader reads data. If you specify this parameter, the source table is sharded based on the value of this parameter. Data Integration then runs parallel threads to read data. You can specify a field of an integer data type for the splitPk parameter. If the source table does not contain fields of integer data types, you can leave this parameter empty.

Configure SAP HANA Reader by using the codeless UI

1. Configure data sources.

   Log on to the DataWorks console. The **DataStudio** page appears. On the DataStudio page, move the pointer over the icon and choose **Data Integration > Batch Synchronization**. In the **Create Node** dialog box, configure the parameters to create a batch synchronization node.

Configure **Source** and Target for the synchronization node.



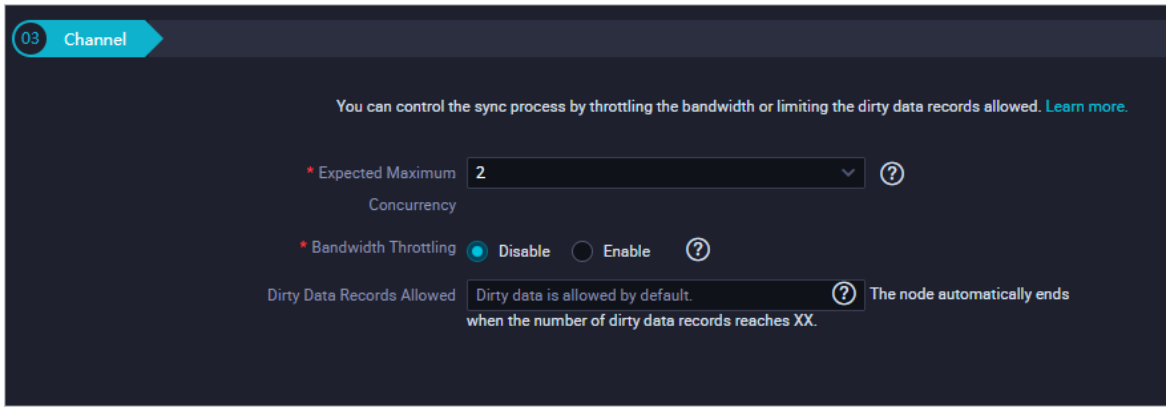
Parameter	Description
Connection	The name of the data source from which you want to read data. This parameter corresponds to the datasource parameter that is described in the preceding section.
Table	The name of the table from which you want to read data. This parameter corresponds to the table parameter that is described in the preceding section.
Data filter	The condition that is used to filter the data you want to read. Filtering based on the LIMIT keyword is not supported. The SQL syntax is determined by the selected data source.
Split pk	The shard key. You can use a column in the source table as the shard key. We recommend that you use the primary key column or an indexed column. Only integer columns are supported. If you specify this parameter, data sharding is performed based on the value of this parameter, and parallel threads can be used to read data. This improves data synchronization efficiency. **Note** The Split pk parameter is displayed only after you select the data source for the synchronization node.

2. Configure field mappings. This operation is equivalent to setting the column parameter that is described in the preceding section.

   Fields in the source on the left have a one-to-one mapping with fields in the destination on the right. You can click **Add** to add a field. To remove an added field, move the pointer over the field and click the **Remove** icon.

Operation	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish mappings between fields with the same name. The data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish mappings between fields in the same row. The data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove the mappings that are established.
Auto Layout	Click Auto Layout. Then, the system automatically sorts the fields based on specific rules.
Change Fields	Click the Change Fields icon. In the Change Fields dialog box, you can manually edit the fields in the source table. Each field occupies a row. The first and the last blank rows are included, whereas other blank rows are ignored.
Add	Click **Add** to add a field. Take note of the following rules when you add a field: ○ You can enter constants. Each constant must be enclosed in single quotation marks ('), such as 'abc' and '123'. ○ You can use scheduling parameters, such as ${bizdate}. ○ You can enter functions that are supported by relational databases, such as now() and count(1). ○ If the field that you entered cannot be parsed, the value of Type for the field is Unidentified.

3. Configure channel control policies.



Parameter	Description
Expected Maximum Concurrency	The maximum number of parallel threads that the synchronization node uses to read data from the source or write data to the destination. You can configure the parallelism for the synchronization node on the codeless UI.

Parameter	Description
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workloads on the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure SAP HANA Reader by using the code editor

The following examples show how to configure SAP HANA Reader to read data from a database or table that is not sharded and how to configure SAP HANA Reader to read data from a database or table that is sharded.

- Configure SAP HANA Reader to read data from a database or table that is not sharded

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"saphana",// The reader type.
            "parameter":{
                "column":[// The names of the columns from which you want to read data.
                    "id"
                ],
                "connection":[
                    {   "querySql":["select a,b from join1 c join join2 d on c.id = d.id;
"], // The SQL statement that is used to read data from the source table.
                        "datasource":"",// The name of the data source.
                        "table":[// The name of the source table. The table name must be
enclosed in brackets [].
                            "xxx"
                        ]
                    }
                ],
                "where":"",// The WHERE clause.
                "splitPk":"",// The shard key.
                "encoding":"UTF-8"// The encoding format.
            },
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"stream",
            "parameter":{},
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
```

```
            "errorLimit":{
                "record":"0"// The maximum number of dirty data records allowed.
            },
            "speed":{
                "throttle":false,// Specifies whether to enable bandwidth throttling. The val
ue false indicates that bandwidth throttling is disabled, and the value true indicates th
at bandwidth throttling is enabled. The mbps parameter takes effect only when the throttl
e parameter is set to true.
                "concurrent":1,// The maximum number of parallel threads.
            }
        },
        "order":{
            "hops":[
                {
                    "from":"Reader",
                    "to":"Writer"
                }
            ]
        }
}
```

- Configure SAP HANA Reader to read data from a database or table that is sharded

  > **Note** When you configure a synchronization node to read data from a sharded database or table, you can select multiple SAP HANA tables with the same schema.

```
{
    "type": "job",
    "version": "1.0",
    "configuration": {
        "reader": {
            "plugin": "saphana",
            "parameter": {
                "connection": [
                    {
                        "table": [
                            "tbl1",
                            "tbl2",
                            "tbl3"
                        ],
                        "datasource": "datasourceName1"
                    },
                    {
                        "table": [
                            "tbl4",
                            "tbl5",
                            "tbl6"
                        ],
                        "datasource": "datasourceName2"
                    }
                ],
                "singleOrMulti": "multi",
                "splitPk": "db_id",
                "column": [
                    "id", "name", "age"
                ],
                "where": "1 < id and id < 100"
            }
        },
        "writer": {
        }
    }
}
```

3.6.3.25. ClickHouse Reader

This topic describes the parameters that are supported by ClickHouse Reader and how to configure ClickHouse Reader by using the code editor.

Limits

- ClickHouse Reader connects to an ApsaraDB for ClickHouse database by using Java Database Connectivity (JDBC) and can read data from a source table only by using JDBC Statement.
- ClickHouse Reader allows you to read data from the specified columns in an order different from that specified in the schema of the source table.
- You must make sure that the driver version is compatible with your ClickHouse database. ClickHouse Reader supports only the following version of the ClickHouse database driver:

```
<dependency>
    <groupId>ru.yandex.clickhouse</groupId>
    <artifactId>clickhouse-jdbc</artifactId>
    <version>0.2.4.ali2-SNAPSHOT</version>
</dependency>
```

Background information

ClickHouse Reader is designed for extract, transform, load (ETL) developers to read data from ApsaraDB for ClickHouse databases. ClickHouse Reader connects to a remote ApsaraDB for ClickHouse database by using JDBC, generates an SQL statement based on your configurations to read data from the database, matches the protocol of each writer of Data Integration, and writes data to other engines by using a write API that is provided by each engine.

Parameters

Parameter	Description	Required	Default
datasource	The name of the data source. It must be the same as the name of the added data source. You can add data sources by using the code editor.	Yes	None
table	The name of the table from which you want to read data. The table contains data in the JSON format. **Note** The table parameter must be included in the connection parameter.	Yes	None
column	The names of the columns to which you want to write data in the destination table. Separate the names with commas (,), such as "column": ["id", "name", "age"]. **Note** You must specify the column parameter.	Yes	None
jdbcUrl	The JDBC URL of the source database. The jdbcUrl parameter must be included in the connection parameter. • You can configure only one JDBC URL for a database. • The value format of the jdbcUrl parameter must be in accordance with the official specifications of ClickHouse. You can also specify additional JDBC connection properties in the value of this parameter. Example: jdbc:clickhouse://localhost:3306/test?user=root&password=&useUnicode=true&characterEncoding=gbk &autoReconnect=true&failOverReadOnly=false.	Yes	None
username	The username that you can use to connect to the database.	Yes	None
password	The password that you can use to connect to the database.	Yes	None

Parameter	Description	Required	Default
splitPk	The field that is used for data sharding when ClickHouse Reader reads data. If you specify this parameter, the source table is sharded based on the value of this parameter. Data Integration then runs parallel threads to read data. This way, data can be synchronized more efficiently.	No	None
where	The WHERE clause. For example, you can set this parameter to `gmt_create > $bizdate` to read the data that is generated on the current day. You can use the WHERE clause to read incremental data. If the where parameter is not provided or is left empty, ClickHouse Reader reads all data.	No	None

Configure ClickHouse Reader by using the codeless UI

This method is not supported.

Configure ClickHouse Reader by using the code editor

In the following code, a synchronization node is configured to read data from ApsaraDB for ClickHouse. For more information about the parameters, see the preceding parameter description.

> **Note** Delete the comments from the code before you run the code.

```
{
    "type": "job",
    "version": "2.0",
    "steps": [
        {
            "stepType": "clickhouse", // The reader type.
            "parameter": {
                "datasource": "example",
                "column": [   // The names of the columns from which you want to read data.

                    "id",
                    "name"
                ],
                "where": "",    // The condition that is used to filter data you want to re
ad.
                "splitPk": "",  // The shard key.
                "table": ""    // The name of the table from which you want to read data.
            },
            "name": "Reader",
            "category": "reader"
        },
        {
            "stepType": "clickhouse",
            "parameter": {
                "postSql": [
                    "update @table set db_modify_time = now() where db_id = 1"
```

```
                    update @table set db_modify_time = now() where db_id = 1
                ],
                "datasource": "example",    // The name of the data source.
                "batchByteSize": "67108864",
                "column": [
                    "id",
                    "name"
                ],
                "writeMode": "insert",
                "encoding": "UTF-8",
                "batchSize": 1024,
                "table": "ClickHouse_table",
                "preSql": [
                    "delete from @table where db_id = -1"
                ]
            },
            "name": "Writer",
            "category": "writer"
        }
    ],
    "setting": {
        "executeMode": null,
        "errorLimit": {
            "record": "0"  // The maximum number of dirty data records allowed.
        },
        "speed": {
         "throttle":true,// Specifies whether to enable bandwidth throttling. A value of fa
lse indicates that bandwidth throttling is disabled, and a value of true indicates that ban
dwidth throttling is enabled. The mbps parameter takes effect only when the throttle parame
ter is set to true.
            "concurrent":1 // The maximum number of parallel threads.
            "mbps":"12",// The maximum transmission rate.
        }
    },
    "order": {
        "hops": [
            {
                "from": "Reader",
                "to": "Writer"
            }
        ]
    }
}
```

3.6.3.26. TSDB Reader

TSDB Reader allows you to read data from Time Series Database (TSDB). You can synchronize data from TSDB in batch mode by using TSDB Reader. This topic describes the data types and parameters that are supported by TSDB Reader and how to configure TSDB Reader by using the code editor.

Background information

TSDB is a high-performance, cost-effective, stable, and reliable online time series database service. TSDB features high read and write performance and provides a high compression ratio for data storage. TSDB also enables the interpolation and aggregation of time series data. TSDB can monitor devices and business and send alerts in real time. It is widely used in Internet and IoT fields.

TSDB Reader connects to a TSDB instance by sending an HTTP request and obtains data points by using the `/api/query` or `/api/mquery` HTTP API endpoint. TSDB Reader splits a synchronization node into multiple tasks based on time series and a specific time range.

Limits

- DataWorks supports only batch synchronization of TSDB data by using the code editor.
- When TSDB Reader reads data, the specified start time and end time are automatically converted to on-the-hour time. For example, if you set the time range to [3:35, 4:55) on April 18, 2019, the time range of [3:35, 4:55) is converted to [3:00, 4:00).

Usage notes

To ensure that synchronization nodes can run as expected, you must change the memory size of the Java Virtual Machine (JVM) to an appropriate value in the following scenarios:

- If the amount of data that is reported for a metric within an hour is greater than a specific threshold, you must modify the value of the -j parameter to change the memory size of the JVM.
- If the data write speed of the downstream TSDB Writer is lower than the data read speed of TSDB Reader, a task backlog may occur. Therefore, you must change the memory size of the JVM to an appropriate value.

For example, to extract data from an Alibaba Cloud TSDB database to an on-premises data source, you can run the following command in which the JVM memory size is specified: You can change the memory size of the JVM based on your requirements.

```
python datax/bin/datax.py tsdb2stream.json -j "-Xms4096m -Xmx4096m"
```

Data types

TSDB Reader can convert TSDB data to the STRING type supported by Data Integration. The following table describes the mapping between the data type mapping.

TSDB data type	Data Integration data type
The string to which a data point in TSDB is serialized, including timestamp, metric, tag, field, and value.	STRING

Parameters

The following table describes the parameters that you must set when you synchronize TSDB data by using the code editor.

Parameter	Description	Default
name	The name of TSDB Reader.	*tsdbreader*

Parameter	Description	Default
sinkDbType	The type of the destination database from which you want to read data. Valid values of sinkDbType: *TSDB* and *RDB*. • The value *TSDB* indicates that the source database is an Alibaba Cloud TSDB database, an OpenTSDB database, a Prometheus database, or a Timescale database. • The value *RDB* indicates that the source database is a relational database, such as an AnalyticDB for MySQL database, an Oracle database, a MySQL database, a PostgreSQL database, or a Distributed Relational Database Service (DRDS) database.	*TSDB*
endpoint	The HTTP endpoint of the source TSDB database. Specify the endpoint in the format of `http://IP:Port`.	None
username	The username used to connect to the source TSDB database.	None
password	The password used to connect to the source TSDB database.	None
column	The names of the columns from which you want to read data. The value of this parameter varies based on the value of the sinkDbType parameter. • If the value of the sinkDbType parameter is TSDB, set the column parameter to the names of the metrics that you want to read. • If the value of the sinkDbType parameter is RDB, set the column parameter to the table fields from which you want to read data. You can also specify the __metric__, __ts__, and __value__ fields in the column parameter based on your business requirements. ○ __metric__: used to read metrics. ○ __ts__: used to read timestamps. ○ __value__: used to read single-value data. To read multi-value data, specify the field parameter.	None
metric	The metrics that you want to read. This parameter is valid only when the sinkDbType parameter is set to RDB.	None
field	The fields that you want to read. This parameter is valid only for multi-value data synchronization.	None
tag	The tags that you want to read. Specify the tags in the format of key-value pairs. These tags are used to filter time series.	None
splitIntervalMs	The interval at which TSDB Reader reads data from the source database. A synchronization node is split into multiple tasks to read data based on the interval. Unit: milliseconds.	None

Parameter	Description	Default
beginDateTime	The start time of the time range of the data points that you want to read. Specify the start time in the format of `yyyy-MM-dd HH:mm:ss`. The beginDateTime and endDateTime parameters must be used in pairs. **Note** When TSDB Reader reads data, the specified start time and end time are automatically converted to on-the-hour time. For example, if you set the time range to [3:35, 4:55) on April 18, 2019, the time range of [3:35, 4:55) is converted to [3:00, 4:00).	None
endDateTime	The end time of the time range of the data points that you want to read. Specify the end time in the format of `yyyy-MM-dd HH:mm:ss`. The beginDateTime and endDateTime parameters must be used in pairs. **Note** When TSDB Reader reads data, the specified start time and end time are automatically converted to on-the-hour time. For example, if you set the time range to [3:35, 4:55) on April 18, 2019, the time range of [3:35, 4:55) is converted to [3:00, 4:00).	None
combine	Specifies whether to combine multiple rows into a single row in the output. If the source table is a sparse table, you can set this parameter to true for TSDB Reader to combine multiple rows into a single row in the output. **Note** • However, TSDB Reader combines only rows that have the same timestamp. • If the combine parameter is set to true, TSDB Reader exports values by using __metric__.xxx. In this case, you do not need to specify the __value__ field. • To ensure that metrics can be obtained, specify the __metric__.xxx field at the beginning of the array that is specified in the column parameter.	*false*

Configure TSDB Reader by using the code editor

For more information about how to configure TSDB Reader by using the code editor, see Create a synchronization node by using the code editor. The following sample code provides examples on how to configure TSDB Reader in different scenarios:

- In this example, a synchronization node is configured to read time series data from an Alibaba Cloud for TSDB database and write the data to an on-premises data source.

  The following code provides sample time series data:

  ```
  {"metric":"m","tags":{"app":"a19","cluster":"c5","group":"g10","ip":"i999","zone":"z1"},"
  timestamp":1546272263,"value":1}
  ```

  The following code provides the configuration of the synchronization node:

```
{
    "type":"job",
    "version":"2.0", // The version number.
    "steps":[
        {
            "stepType":"tsdb",
              "parameter": {
                "sinkDbType": "TSDB", // The type of the destination database.
                "endpoint": "http://localhost:8242",
                "username": "The username used to log on to TSDB",
                "password": "The password used to log on to TSDB",
                "column": [
                  "m"
                ],
                "combine":false,
                "splitIntervalMs": 60000,
                "beginDateTime": "2019-01-01 00:00:00",
                "endDateTime": "2019-01-01 01:00:00"
              },
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"stream",
            "parameter":{
            },
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0"// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":true,// Specifies whether to enable bandwidth throttling. A value
of false indicates that bandwidth throttling is disabled, and a value of true indicates t
hat bandwidth throttling is enabled. The mbps parameter takes effect only when the thrott
le parameter is set to true.
            "concurrent":1, // The maximum number of parallel threads.
            "mbps":"12"// The maximum transmission rate.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

- In this example, a synchronization node is configured to read relational data from an Alibaba Cloud TSDB database and write the data to an on-premises data source.

  The following code provides sample relational data:

```
m 1546272125 a1    c1 g2 i3021 z4 1.0
```

  The following code provides the configuration of the synchronization node:

```
{
    "type":"job",
    "version":"2.0", // The version number.
    "steps":[
        {
            "stepType":"tsdb",
              "parameter": {
                "sinkDbType": "RDB", // The type of the destination database.
                "endpoint": "http://localhost:8242",
                "column": [
                  "__metric__",
                  "__ts__",
                  "app",
                  "cluster",
                  "group",
                  "ip",
                  "zone",
                  "__value__"
                ],
                "metric": [
                  "m"
                ],
                "splitIntervalMs": 60000,
                "beginDateTime": "2019-01-01 00:00:00",
                "endDateTime": "2019-01-01 01:00:00"
              },
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"stream",
            "parameter":{
            },
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0"// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":true,// Specifies whether to enable bandwidth throttling. A value
of false indicates that bandwidth throttling is disabled, and a value of true indicates t
hat bandwidth throttling is enabled. The mbps parameter takes effect only when the thrott
le parameter is set to true.
```

```
            "concurrent":1, // The maximum number of parallel threads.
            "mbps":"12"// The maximum transmission rate.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

- In this example, a synchronization node is configured to read single-value data from an Alibaba Cloud TSDB database and write the data to an AnalyticDB for MySQL database.

```
{
    "type":"job",
    "version":"2.0", // The version number.
    "steps":[
        {
            "stepType":"tsdb",
              "parameter": {
                "sinkDbType": "RDB",
                "endpoint": "http://localhost:8242",
                "column": [
                  "__metric__",
                  "__ts__",
                  "app",
                  "cluster",
                  "group",
                  "ip",
                  "zone",
                  "__value__"
                ],
                "metric": [
                  "m"
                ],
                "splitIntervalMs": 60000,
                "beginDateTime": "2019-01-01 00:00:00",
                "endDateTime": "2019-01-01 01:00:00"
              },
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"stream",
            "parameter":{
            },
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
```

```
    "setting":{
        "errorLimit":{
            "record":"0"// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":true,// Specifies whether to enable bandwidth throttling. A value
of false indicates that bandwidth throttling is disabled, and a value of true indicates t
hat bandwidth throttling is enabled. The mbps parameter takes effect only when the thrott
le parameter is set to true.
            "concurrent":1, // The maximum number of parallel threads.
            "mbps":"12"// The maximum transmission rate.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

- In this example, a synchronization node is configured to read multi-value data from an Alibaba Cloud TSDB database and write the data to an AnalyticDB for MySQL database.

```
{
    "type":"job",
    "version":"2.0", // The version number.
    "steps":[
        {
            "stepType":"tsdb",
              "parameter": {
                "sinkDbType": "RDB",
                "endpoint": "http://localhost:8242",
                "username": "The username used to log on to TSDB",
                "password": "The password used to log on to TSDB",
                "column": [
                  "__metric__",
                  "__ts__",
                  "app",
                  "cluster",
                  "group",
                  "ip",
                  "zone",
                  "load",
                  "memory",
                  "cpu"
                ],
                "metric": [
                  "m_field"
                ],
                "field": {
                  "m_field": [
                    "load",
```

```
                    "memory",
                    "cpu"
                  ]
                },
                "splitIntervalMs": 60000,
                "beginDateTime": "2019-01-01 00:00:00",
                "endDateTime": "2019-01-01 01:00:00"
              },
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"ads",
              "parameter": {
                "username": "******",
                "password": "******",
                "column": [
                  "`metric`",
                  "`ts`",
                  "`app`",
                  "`cluster`",
                  "`group`",
                  "`ip`",
                  "`zone`",
                  "`load`",
                  "`memory`",
                  "`cpu`"
                ],
                "url": "http://localhost:3306",
                "schema": "datax_test",
                "table": "datax_test_multi_field",
                "writeMode": "insert",
                "opIndex": "0",
                "batchSize": "2"
              },
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0"// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":true,// Specifies whether to enable bandwidth throttling. A value
of false indicates that bandwidth throttling is disabled, and a value of true indicates t
hat bandwidth throttling is enabled. The mbps parameter takes effect only when the thrott
le parameter is set to true.
            "concurrent":1, // The maximum number of parallel threads.
            "mbps":"12"// The maximum transmission rate.
        }
    },
    "order":{
        "hops":[
```

```
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

- In this example, a synchronization node is configured to read single-value data in a specific time range from an Alibaba Cloud TSDB database and write the data to an AnalyticDB for MySQL database.

```
{
    "type":"job",
    "version":"2.0", // The version number.
    "steps":[
        {
            "stepType":"tsdb",
              "parameter": {
                "sinkDbType": "RDB",
                "endpoint": "http://localhost:8242",
                "username": "The username used to log on to TSDB",
                "password": "The password used to log on to TSDB",
                "column": [
                  "__metric__",
                  "__ts__",
                  "app",
                  "cluster",
                  "group",
                  "ip",
                  "zone",
                  "__value__"
                ],
                "metric": [
                  "m"
                ],
                "tag": {
                  "m": {
                    "app": "a1",
                    "cluster": "c1"
                  }
                },
                "splitIntervalMs": 60000,
                "beginDateTime": "2019-01-01 00:00:00",
                "endDateTime": "2019-01-01 01:00:00"
              },
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"ads",
              "parameter": {
                "username": "******",
                "password": "******",
                "column": [
```

```
                    "`metric`",
                    "`ts`",
                    "`app`",
                    "`cluster`",
                    "`group`",
                    "`ip`",
                    "`zone`",
                    "`value`"
                  ],
                  "url": "http://localhost:3306",
                  "schema": "datax_test",
                  "table": "datax_test",
                  "writeMode": "insert",
                  "opIndex": "0",
                  "batchSize": "2"
                },
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0"// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":true,// Specifies whether to enable bandwidth throttling. A value
of false indicates that bandwidth throttling is disabled, and a value of true indicates t
hat bandwidth throttling is enabled. The mbps parameter takes effect only when the thrott
le parameter is set to true. .
            "concurrent":1, // The maximum number of parallel threads.
            "mbps":"12"// The maximum transmission rate.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

- In this example, a synchronization node is configured to read multi-value data in a specific time range from an Alibaba Cloud TSDB database and write the data to an AnalyticDB for MySQL database.

```
{
    "type":"job",
    "version":"2.0", // The version number.
    "steps":[
        {
            "stepType":"tsdb",
              "parameter": {
                "sinkDbType": "RDB",
```

```
                "endpoint": "http://localhost:8242",
                "username": "The username used to log on to TSDB",
                "password": "The password used to log on to TSDB",
                "column": [
                  "__metric__",
                  "__ts__",
                  "app",
                  "cluster",
                  "group",
                  "ip",
                  "zone",
                  "load",
                  "memory",
                  "cpu"
                ],
                "metric": [
                  "m_field"
                ],
                "field": {
                  "m_field": [
                    "load",
                    "memory",
                    "cpu"
                  ]
                },
                "tag": {
                  "m_field": {
                    "ip": "i999"
                  }
                },
                "splitIntervalMs": 60000,
                "beginDateTime": "2019-01-01 00:00:00",
                "endDateTime": "2019-01-01 01:00:00"
              },
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"ads",
              "parameter": {
                "username": "******",
                "password": "******",
                "column": [
                  "`metric`",
                  "`ts`",
                  "`app`",
                  "`cluster`",
                  "`group`",
                  "`ip`",
                  "`zone`",
                  "`load`",
                  "`memory`",
                  "`cpu`"
                ],
                "url": "http://localhost:3306"
```

```
                "jdbcUrl": "http://localhost:3306",
                "schema": "datax_test",
                "table": "datax_test_multi_field",
                "writeMode": "insert",
                "opIndex": "0",
                "batchSize": "2"
              },
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0"// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":true,// Specifies whether to enable bandwidth throttling. A value
of false indicates that bandwidth throttling is disabled, and a value of true indicates t
hat bandwidth throttling is enabled. The mbps parameter takes effect only when the thrott
le parameter is set to true.
            "concurrent":1, // The maximum number of parallel threads.
            "mbps":"12"// The maximum transmission rate.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

- In this example, a synchronization node is configured to read single-value data from an Alibaba Cloud TSDB database and write the data to another Alibaba Cloud TSDB database.

```
{
    "type":"job",
    "version":"2.0", // The version number.
    "steps":[
        {
            "stepType":"tsdb",
              "parameter": {
                "sinkDbType": "TSDB",
                "endpoint": "http://localhost:8242",
                "username": "The username used to log on to TSDB",
                "password": "The password used to log on to TSDB",
                "column": [
                  "m"
                ],
                "splitIntervalMs": 60000,
                "beginDateTime": "2019-01-01 00:00:00",
                "endDateTime": "2019-01-01 01:00:00"
              },
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"tsdb",
              "parameter": {
                "endpoint": "http://localhost:8240"
              },
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0"// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":true,// Specifies whether to enable bandwidth throttling. A value
of false indicates that bandwidth throttling is disabled, and a value of true indicates t
hat bandwidth throttling is enabled. The mbps parameter takes effect only when the thrott
le parameter is set to true.
            "concurrent":1, // The maximum number of parallel threads.
            "mbps":"12"// The maximum transmission rate.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

- In this example, a synchronization node is configured to read multi-value data from an Alibaba Cloud

TSDB database and write the data to another Alibaba Cloud TSDB database.

```
{
    "type":"job",
    "version":"2.0", // The version number.
    "steps":[
        {
            "stepType":"tsdb",
              "parameter": {
                "sinkDbType": "TSDB",
                "endpoint": "http://localhost:8242",
                "username": "The username used to log on to TSDB",
                "password": "The password used to log on to TSDB",
                "column": [
                  "m_field"
                ],
                "field": {
                  "m_field": [
                    "load",
                    "memory",
                    "cpu"
                  ]
                },
                "splitIntervalMs": 60000,
                "beginDateTime": "2019-01-01 00:00:00",
                "endDateTime": "2019-01-01 01:00:00"
              },
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"tsdb",
              "parameter": {
                "multiField": true,
                "endpoint": "http://localhost:8240"
              },
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0"// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":true,// Specifies whether to enable bandwidth throttling. A value
of false indicates that bandwidth throttling is disabled, and a value of true indicates t
hat bandwidth throttling is enabled. The mbps parameter takes effect only when the thrott
le parameter is set to true.
            "concurrent":1, // The maximum number of parallel threads.
            "mbps":"12"// The maximum transmission rate.
        }
    },
    "order":{
```

```
        "nops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

3.6.4. Configure the writer

3.6.4.1. Configure AnalyticDB for MySQL 2.0 Writer

This topic describes the data types and parameters supported by AnalyticDB for MySQL 2.0 Writer and how to configure it by using the codeless UI and code editor.

Prerequisites

Data Integration can import data to AnalyticDB for MySQL 2.0 in real time. This method requires you to create real-time tables, which are fact tables, in the destination AnalyticDB for MySQL 2.0 database in advance. In real-time import mode, data is imported efficiently and the process is simple.

You must configure a connection before you configure AnalyticDB for MySQL 2.0 Writer.

Data types

The following table lists the data types supported by AnalyticDB for MySQL 2.0 Writer.

Category	AnalyticDB for MySQL 2.0 data type
Integer	INT, TINYINT, SMALLINT, and BIGINT
Floating point	FLOAT and DOUBLE
String	VARCHAR
Date and time	DATE and TIMESTAMP
Boolean	BOOLEAN

Parameters

Parameter	Description	Required	Default value
connectionUrl	The URL for connecting to the AnalyticDB for MySQL 2.0 database. Specify the parameter in the IP address:Port format.	Yes	None
database	The name of the AnalyticDB for MySQL 2.0 database.	Yes	None
Access Id	The AccessKey ID that you can use to connect to the AnalyticDB for MySQL 2.0 database.	Yes	None

Parameter	Description	Required	Default value
Access Key	The AccessKey secret that you can use to connect to the AnalyticDB for MySQL 2.0 database.	Yes	None
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None
table	The name of the destination table.	Yes	None
partition	The partition name of the destination table. If the destination table is partitioned, this parameter is required.	No	None
writeMode	The write mode. Set the value to insert. In this mode, if a primary key conflict occurs, the conflicting rows are overwritten.	Yes	None
column	The columns in the destination table to which data is written. Separate the columns with commas (,), for example, ["a","b","c"]. Set the value to an asterisk (*) if data is written to all the columns in the destination table.	Yes	None
suffix	Optional. The suffix to the AnalyticDB for MySQL 2.0 URL that is in the format of `IP address:Port`. This suffix is a custom connection string. After this parameter is set, the URL changes to a JDBC connection string for connecting to AnalyticDB for MySQL 2.0. For example, set the suffix parameter to `autoReconnect=true&failOverReadOnly=false&maxReconnects=10`.	No	None
batchSize	The number of data records to write at a time. This parameter is available only when the writeMode parameter is set to insert.	Required only when the writeMode parameter is set to insert	None

Parameter	Description	Required	Default value
bufferSize	The size of the Data Integration data buffer, which is designed to improve the performance of AnalyticDB for MySQL 2.0. Data from the source database is sorted in the buffer before the data is committed to AnalyticDB for MySQL 2.0. The data in the buffer is sorted based on the partition key columns in AnalyticDB for MySQL 2.0. In this way, the data is organized in an order that can improve the performance of the AnalyticDB for MySQL 2.0 server. Data in the buffer is committed to AnalyticDB for MySQL 2.0 in batches based on the batchSize parameter. We recommend that you set the bufferSize value to a multiple of batchSize. This parameter is available only when the writeMode parameter is set to insert.	Required only when the writeMode parameter is set to insert	Disabled

Configure AnalyticDB for MySQL 2.0 Writer by using the codeless UI

1. Configure the connections.

   Configure the connections to the source and destination data stores for the sync node.

Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and then select a connection name that you have configured in DataWorks.
Table	The table parameter in the preceding parameter description.
Write Method	The writeMode parameter in the preceding parameter description.

2. Configure field mapping. It is equivalent to setting the column parameter in the preceding parameter description. Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right.

GUI element	Description
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	Click Auto Layout to sort the fields based on specified rules.

3. Configure channel control policies.

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read data from and write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.

Parameter	Description
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure AnalyticDB for MySQL 2.0 Writer by using the code editor

```
{
    "type":"job",
    "version":"2.0",
    "steps":[
        {
            "stepType":"stream",
            "parameter":{
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"ads",// The writer type.
            "parameter":{
                "partition:"",// The partition name of the destination table.
                "datasource":"",// The connection name.
                "column":[// The columns to which data is written.
                     "id"
                ],
                "writeMode":"insert",// The write mode.
                "batchSize":"256",// The number of data records to write at a time.
                "table":"",// The name of the destination table.
                "overWrite":"true"// Specifies whether to overwrite the destination table w
hen data is written to AnalyticDB for MySQL 2.0. A value of true indicates that the destina
tion table is overwritten. A value of false indicates that the destination table is not ove
rwritten and the new data is appended to the existing data. This value takes effect only wh
en the writeMode parameter is set to load.
            },
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0"// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false,// Specifies whether to enable bandwidth throttling. A value o
f false indicates that the bandwidth is not throttled. A value of true indicates that the b
andwidth is throttled. The maximum transmission rate takes effect only if you set this para
meter to true.
            "concurrent":1,// The maximum number of concurrent threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

3.6.4.2. Configure DataHub Writer

This topic describes the data types and parameters supported by DataHub Writer and how to configure it by using the code editor.

DataHub is a real-time data distribution platform designed to process streaming data. You can publish and subscribe applications to streaming data in DataHub and distribute the data to other platforms. This allows you to easily analyze streaming data and build applications based on the streaming data.

Based on the Apsara system of Alibaba Cloud, DataHub features high availability, low latency, high scalability, and high throughput. Seamlessly integrated with Realtime Compute, DataHub allows you to easily use SQL to analyze streaming data. DataHub can also distribute streaming data to Alibaba Cloud services such as MaxCompute and OSS.

> **Note** Strings can only be UTF-8 encoded. The size of each string must not exceed 1 MB.

Parameter configuration

The source is connected to the destination through a single channel. Therefore, the channel type configured for the writer must be the same as that configured for the reader. Generally, channels are categorized into two types: memory andfile. The following configuration sets the channel type to file:

```
"agent.sinks.dataXSinkWrapper.channel": "file"
```

Parameters

Parameter	Description	Required	Default value
accessId	The AccessKey ID for accessing DataHub.	Yes	None
accessKey	The AccessKey secret for accessing DataHub.	Yes	None
endpoint	The endpoint of DataHub.	Yes	None
maxRetryCount	The maximum number of retries if a task fails.	No	None
mode	The mode for writing strings.	Yes	None
parseContent	The data that has been parsed.	Yes	None
project	The organizational unit in DataHub. Each project contains one or more topics. **Note** DataHub projects are independent from MaxCompute projects. Projects created in MaxCompute cannot be used in DataHub.	Yes	None
topic	The minimum unit for data subscription and publication. You can use topics to distinguish different types of streaming data.	Yes	None

Parameter	Description	Required	Default value
maxCommitSize	The amount of data, in MB, that DataHub Writer buffers before sending it to the destination. This mechanism aims to improve writing efficiency. The default value is 1048576, in KB, that is, 1 MB.	No	*1048576*
batchSize	The number of data records that DataHub Writer buffers before sending them to the destination. This mechanism aims to improve writing efficiency. The default value is 1024.	No	*1,024*
maxCommitInterval	The maximum interval at which DataHub Writer sends data to the destination. When an interval ends, DataHub Writer sends buffered data even if the data amount does not reach the preceding two thresholds. The default value is 30000, in milliseconds, that is, 30 seconds.	No	*30,000*
parseMode	The mode for parsing log entries. Valid values: *default* and *csv*. The value default indicates that no log parsing is required. The value csv indicates that a delimiter is inserted between fields for each log entry.	No	*default*

Configure DataHub Writer by using the codeless UI

Currently, the codeless UI is not supported for DataHub Writer.

Configure DataHub Writer by using the code editor

In the following code, a node is configured to read data from the memory and then write the data to DataHub.

```
{
    "type": "job",
    "version": "2.0", // The version number.
    "steps": [
        {
            "stepType": "stream",
            "parameter": {},
            "name": "Reader",
            "category": "reader"
        },
        {
            "stepType": "datahub", // The writer type.
            "parameter": {
                "datasource": "", // The connection name.
                "topic": "",// The minimum unit for data subscription and publication. You
can use topics to distinguish different types of streaming data.
                "maxRetryCount":500,// The maximum number of retries if a task fails.
                "maxCommitSize": 1048576// The amount of data, in MB, that DataHub Writer b
uffers before sending it to the destination.
                "shardId": "xxxxxx" // The shard of the DataHub topic.
            },
            "name": "Writer",
            "category": "writer"
        }
    ],
    "setting": {
        "errorLimit": {
            "record": "" // The maximum number of dirty data records allowed.
        },
        "speed": {
            "concurrent": 20, // The maximum number of concurrent threads.
            "throttle": false, // The value false indicates that the bandwidth is not throt
tled. The value true indicates that the bandwidth is throttled. The maximum transmission ra
te takes effect only if you set this parameter to true.
        }
    },
    "order": {
        "hops": [
            {
                "from": "Reader",
                "to": "Writer"
            }
        ]
    }
}
```

3.6.4.3. Configure the DB2 writer

The DB2 writer enables writing data to tables stored on Db2 databases. To write data into a Db2 table, the DB2 writer connects to the remote Db2 database through JDBC, and runs `INSERT INTO` statements. Data is written into the Db2 table in batches.

The DB2 writer is designed for ETL developers to import data from data warehouses to Db2 databases. It also serves as a data migration tool for database administrators and other users.

The DB2 writer reads data from the channel, connects to a remote Db2 database through JDBC, and then runs `INSERT INTO` statements. The rows that violate the unique index constraint or primary key constraint cannot be written into the Db2 database. To improve performance, the DB2 writer makes `batch updates with the PreparedStatement method` and sets `rewriteBatchedStatements=true`. In this way, the DB2 writer buffers data, and submits a write request when the amount of data in the buffer reaches a specific threshold.

> **Note** The INSERT INTO privilege is required for data synchronization tasks with the DB2 writer. Whether other permissions are required depends on the SQL statements specified in the preSql and postSql parameters.

The DB2 writer supports most Db2 data types. Since still some of the Db2 data types are not supported, verify that your data types are supported.

The following table lists data types supported by the DB2 writer.

Data Integration data type	Db2 data type
Integer	SMALLINT
Floating point	DECIMAL, REAL, and DOUBLE
String	CHAR, CHARACTER, VARCHAR, GRAPHIC, VARGRAPHIC, LONG VARCHAR, CLOB, LONG VARGRAPHIC, and DBCLOB
Date and time	DATE, TIME, and TIMESTAMP
Boolean	N/A
Binary	BLOB

Parameters

Parameter	Description	Required	Default value
jdbcUrl	The JDBC connectivity URL, used to connect to the Db2 database. In accordance with Db2 official specifications, the URL format must be jdbc:db2://ip:port/database. You can also specify the information of the attachment facility.	Yes	None
username	The username used to connect to the data source.	Yes	None
password	The password used to connect to the data source.	Yes	None
table	The name of the destination table.	Yes	None

Parameter	Description	Required	Default value
column	The columns in the destination table to which data is written. Separate the columns with a comma (,). Example: "column": ["id", "name", "age"]. Set the value to an asterisk (*) if data is written to all the columns in the destination table. Example: "column": ["*"].	Yes	None
preSql	The SQL statement runs before the data synchronization task starts. Currently, you can run only one SQL statement. For example, you can run a statement to clear outdated data.	No	None
postSql	The SQL statement runs after the data synchronization task ends. Currently, you can run only one SQL statement in wizard mode but multiple SQL statements in script mode. For example, you can run a statement to add a timestamp.	No	None
batchSize	The number of data records to write per batch. Setting this parameter can greatly reduce the interactions between Data Integration and the Db2 database over the network, and increase the throughput. However, an excessively large value may cause the running Data Integration process to become out of memory (OOM).	No	1024

Configure the DB2 writer in wizard mode

Currently, wizard mode is not supported for the DB2 writer.

Configure the DB2 writer in script mode

In the following script, a task is configured to write data to a Db2 database.

```
{
    "type":"job",
    "version":"2.0", // The version number.
    "steps":[
        { // The following template is used to configure the reader. For more information,
see the corresponding section.
            "stepType":"stream",
            "parameter":{},
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"db2", // The writer type.
            "parameter":{
                "postSql":[], // The SQL statement runs after the data synchronization task
ends.
                "password":"", // The password.
                "jdbcUrl":"jdbc:db2://ip:port/database", //The JDBC connectivity URL, used
to connect to the Db2 database.
                "column":[
                    "id"
                ],
                "batchSize":1024, // The number of data records to write per batch.
                "table":"", // The table name.
                "username":"", // The username.
                "preSql": [] // The SQL statement runs before the data synchronization task
starts.
            },
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0" // The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false, // The value false means that the bandwidth is not throttled.
The value true means that the bandwidth is throttled. The maximum transmission rate takes e
ffect only if you specify this parameter as true.
            "concurrent":1, // The maximum number of concurrent threads.
            "dmu":1 // The number of DMUs.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

3.6.4.4. Configure DRDS Writer

This topic describes the data types and parameters supported by DRDS Writer and how to configure it by using the codeless UI and code editor.

DRDS Writer allows you to write data to tables stored in DRDS databases. DRDS Writer connects to the proxy of a remote DRDS database by using JDBC, and executes a `REPLACE INTO` statement to write data to the DRDS database.

> **Note**
> - To execute the `REPLACE INTO` statement, make sure that your table has the primary key or a unique index to avoid replicated data.
> - You must configure a connection before you configure DRDS Writer.

DRDS Writer is designed for ETL developers to import data from data warehouses to DRDS databases. DRDS Writer can also be used as a data migration tool by users such as DBAs.

DRDS Writer obtains data from a Data Integration reader, and writes the data to the destination database by executing the `REPLACE INTO` statement. If no primary key conflict or unique index conflict occurs, the action is the same as that of the `INSERT INTO` statement. If a conflict occurs, original rows are replaced by new rows. DRDS Writer sends data to the DRDS proxy when the amount of buffered data reaches a specific threshold. The proxy determines whether to write the data to one or more tables and how to route the data when it is written to multiple tables.

> **Note** A sync node that uses DRDS Writer must have at least the permission to execute the REPLACE INTO statement. Whether other permissions are required depends on the SQL statements specified in the preSql and postSql parameters.

Similar to MySQL Writer, DRDS Writer supports most MySQL data types. Make sure that your data types are supported.

The following table lists the data types supported by DRDS Writer.

Category	DRDS data type
Integer	INT, TINYINT, SMALLINT, MEDIUMINT, BIGINT, and YEAR
Floating point	FLOAT, DOUBLE, and DECIMAL
String	VARCHAR, CHAR, TINYTEXT, TEXT, MEDIUMTEXT, and LONGTEXT
Date and time	DATE, DATETIME, TIMESTAMP, and TIME
Boolean	BIT and BOOLEAN
Binary	TINYBLOB, MEDIUMBLOB, BLOB, LONGBLOB, and VARBINARY

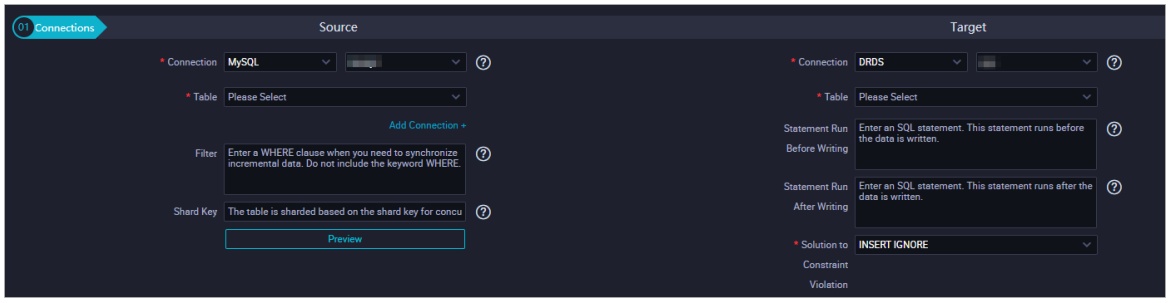
Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None
table	The name of the destination table.	Yes	None
writeMode	The write mode. Valid values: *insert into*, *on duplicate key update*, and *replace into*. • *insert into*: If a primary key conflict or unique index conflict occurs, data cannot be written to the conflicting rows and is regarded as dirty data. • *on duplicate key update*: If no primary key conflict or unique index conflict occurs, the action is the same as that of `insert into`. If a conflict occurs, specified fields in original rows are updated. • *replace into*: If no primary key conflict or unique index conflict occurs, the action is the same as that of `insert into`. If a conflict occurs, original rows are deleted and new rows are inserted. This means that all the field values of the original rows are replaced.	No	*insert*
column	The columns in the destination table to which data is written. Separate the columns with commas (,), for example, "column": ["id","name","age"]. To write data to all the columns in the destination table, set the value to an asterisk (*), for example, "column":["*"].	Yes	None
preSql	The SQL statement to execute before the sync node is run. For example, you can clear outdated data before data synchronization. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor.	No	None
postSql	The SQL statement to execute after the sync node is run. For example, you can add a timestamp after data synchronization. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor.	No	None
batchSize	The number of data records to write at a time. Setting this parameter can greatly reduce the interactions between Data Integration and the DRDS database over the network, and increase the throughput. However, an excessively large value may lead to the OOM error during the data synchronization process.	No	*1,024*

Configure DRDS Writer by using the codeless UI

1. Configure the connections.

Configure the connections to the source and destination data stores for the sync node.



Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and then select a connection name that you have configured in DataWorks.
Table	The table parameter in the preceding parameter description.
Statement Run Before Writing	The preSql parameter in the preceding parameter description. Enter an SQL statement to execute before the sync node is run.
Statement Run After Writing	The postSql parameter in the preceding parameter description. Enter an SQL statement to execute after the sync node is run.
Solution to Constraint Violation	The writeMode parameter in the preceding parameter description. Select the required write mode.

2. Configure field mapping. It is equivalent to setting the column parameter in the preceding parameter description. Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right.

GUI element	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	Click Auto Layout to sort the fields based on specified rules.

3. Configure channel control policies.

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read data from and write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.

Parameter	Description
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure DRDS Writer by using the code editor

In the following code, a node is configured to write data to a DRDS database.

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"stream",
            "parameter":{},
            "name":"Reader",
            "category":"reader"
                },
        {
            "stepType":"drds",// The writer type.
            "parameter":{
                "postSql":[],// The SQL statement to execute after the sync node is run.
                "datasource":"",// The connection name.
                "column":[// The columns to which data is written.
                "id"
                ],
                "writeMode":"insert ignore",
                "batchSize":"1024",// The number of data records to write at a time.
                "table":"test",// The name of the destination table.
                "preSql":[]// The SQL statement to execute before the sync node is run.
                },
            "name":"Writer",
            "category":"writer"
                }
                ],
    "setting":{
        "errorLimit":{
        "record":"0"// The maximum number of dirty data records allowed.
            },
        "speed":{
            "throttle":false,// Specifies whether to enable bandwidth throttling. A value o
f false indicates that the bandwidth is not throttled. A value of true indicates that the b
andwidth is throttled. The maximum transmission rate takes effect only if you set this para
meter to true.
            "concurrent":1 // The maximum number of concurrent threads.
                }
            },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
                }
            ]
        }
    }
```

3.6.4.5. Configure the FTP writer

The FTP writer allows you to write one or more files in CSV format into a remote FTP file. At the underlying level, this writer converts the data that is readable by the Data Integration service to CSV files, and writes these files into the remote FTP server using FTP network protocols. You must configure the data source before configuring the FTP writer.

> **Note** For more information, see Add FTP data sources.

The FTP writer can only write data into FTP files that store logical two-dimensional tables, for example, text information in the CSV format.

This writer enables you to convert data that is readable by the Data Integration service to FTP files. FTP files stores non-structured data. The advantages and disadvantages of the FTP writer are described as follows:

- Only supports text files and the schema in the text file must be a two-dimensional table. It does not support the blob type, such as video data.
- Supports CSV and text files with custom delimiters.
- Does not support text compression when data is written to the destination table.
- Supports multi-thread writing, with each thread performing write operations on a subfile.

Currently, the FTP writer does not support the following two features:

- Concurrent writing for a single file.
- Providing varying data types. The FTP does not provide data types, and the FTP writer writes data of the string type into FTP files.

Parameters

Parameter	Description	Required.	Default value
datasource	The name of the data source. You can add a data source either in wizard or script mode. The value of this parameter must be identical to the real data source name.	Yes	None
timeout	The timeout period for the connection to the FTP server, measured in milliseconds.	No	60000 (1 minute)
path	The path of the FTP file system. The write can write data into multiple files in the path.	Yes	None
FileName	The name of the file into which data is written. A random suffix is added to the file name to form the actual name of the file into which the data is written on each thread.	Yes	None

Parameter	Description	Required.	Default value
writeMode	The mode in which the FTP writer clears existing data before writing data. Valid values: • truncate: The writer clears all the files prefixed by fileName in the path before writing data. • append: No processing is performed on the file before the FTP writer imports data into this file. In the Data Integration service, the FTP writer uses the original file name in the data source. No duplicate file names are allowed. • nonConflict: An error is reported if a file prefixed by fileName exists in the path.	Yes	None
fieldDelimiter	The column delimiter of the file to be written.	Yes. A single character is used.	None
compress	The compress option. The gzip and bzip2 compression options are supported.	No	No
encoding	The encoding of the file to be read.	No	UTF-8
nullFormat	The string that represents null. Since no standard strings can represent null in text files, Data Integration provides the nullFormat parameter to define which string represents a null pointer. For example, if you specify `nullFormat:"null"`, Data Integration considers "null" as a null pointer.	No	None
dateFormat	The date format, for example, "dateFormat": "yyyy-MM-dd".	No	None
fileFormat	The file format, including CSV and text. For the CSV format, if you want to write the data that includes column delimiters, the delimiters are escaped with quotation marks. For text format, the data to be written is separated by column delimiters without being escaped.	No	text
header	The header used when a txt file is written, for example, ['id', 'name', 'age'].	No	None
Markdonefilename	The name of the file marked as "done". After a synchronization task is completed, a MarkDoneFile is generated, based on which you can determine whether the task is executed successfully.	No	None

Configure the FTP writer in wizard mode

1. Select data sources.

Configure the source and destination for the data synchronization task.

Parameter	Description
Data Source	The datasource parameter provided in the preceding table. Select a data source type, and enter the name of a data source that has been configured in DataWorks.
File Path	The path parameter provided in the preceding table.
Column Delimiter	The fieldDelimiter parameter provided in the preceding table. Default value: a comma (,)
Encoding	The encoding parameter provided in the preceding table. Default value: UTF-8.
Null String	The nullFormat parameter provided in the preceding table, which defines a string that represents null.
Compression Format	The nullFormat parameter provided in the preceding table. Default value: No.
Include Header	The skipHeader parameter in the preceding table. Default value: No.
Prefix Conflict	The writeMode parameter provided in the preceding table, which defines a string that represents null.

2. Configure field mappings. It is equivalent to setting the column parameter provided in the preceding table.

   You can map the left-side source table fields to the right-side destination table fields. You can also click **Add** to add a field or click the **Delete** icon to delete a field in the source table.

   After you click **Map Fields in the Same Line**, each source table field is mapped to the destination table field in the same line if exists. Ensure that the conversion between data types is feasible.

3. Configure the channel.

Parameter	Description
DMU	The data processing capabilities. A data migration unit (DMU) represents the data processing capabilities for data integration, given a minimum configuration of the CPU, memory, network, and other resources.
Concurrent Jobs	The maximum number of concurrent threads to read and write data to data storage within the data synchronization task.
Transmission Rate	You can throttle the bandwidth and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Parameter	Description
Task Resource Group	The servers on which tasks are run. If an excessively large number of tasks are run on the default resource group, some tasks may be delayed due to insufficient resources. In this case, you can configure additional servers.

Configure the FTP writer in script mode

In the following script, a task is configured to write data to an FTP database.

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        { // The following template is used to configure the reader. For more information,
see the corresponding section.
            "stepType":"stream",
            "Parameter ":{},
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"ftp",// The plug-in name.
            "parameter":{
                "path":""// The file path.
                "fileName": "",// The file name.
                "nullFormat": "null", // The string that represents null.
                "dateFormat":"yyyy-MM-dd HH:mm:ss", // The time format.
                "datasource": "", // The data source.
                "writeMode": "",// The writing method.
                "fieldDelimiter": "," ,// The column delimiter.
                 "encoding":"",// The encoding.
                "fileFormat": "",// The file type.
            },
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0"// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false,// The value false means that the bandwidth is not throttled.
The value true means that the bandwidth is throttled. The maximum transmission rate takes e
ffect only if you specify this parameter as true.
            "concurrent":"1",// The maximum number of concurrent threads.
            "dmu":1// The number of DMUs.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

3.6.4.6. Configure HBase Writer

This topic describes the features, data types, and parameters supported by HBase Writer and how to configure it by using the code editor.

HBase Writer allows you to write data to HBase data stores. Specifically, HBase Writer connects to a remote HBase data store through the Java client of HBase. Then, HBase Writer uses the PUT method to write data to the HBase data store.

Features

- HBase 0.94.x and 1.1.x are supported.
  - If you use HBase 0.94.x, set the hbaseVersion parameter to 094x for the writer.

    ```
    "writer": {
            "hbaseVersion": "094x"
        }
    ```

  - If you use HBase 1.1.x, set the hbaseVersion parameter to 11x for the writer.

    ```
    "writer": {
            "hbaseVersion": "11x"
        }
    ```

    > **Note** Currently, HBase Writer for HBase 1.1.x is compatible with HBase 2.0. If you have any issues in using HBase Writer with HBase 2.0, submit a ticket.

- You can use concatenated fields as a rowkey.

  Currently, HBase Writer supports concatenating multiple fields to generate the rowkey of an HBase table.

- You can set the version of each HBase cell.

  The information that can be used as the version of an HBase cell includes:

  - Current time
  - Specified source column
  - Specified time

Data types

The following table lists the data types supported by HBase Writer.

> **Note**
> - The types of the specified columns must be the same as those in the HBase table.
> - Data types that are not listed in the table are not supported.

Category	HBase data type
Integer	Int, Long, and Short
Floating point	Float and Double
Boolean	Boolean

Category	HBase data type
String	String

Parameters

Parameter	Description	Required	Default value
haveKerberos	Specifies whether Kerberos authentication is required. A value of true indicates that Kerberos authentication is required. **Note** • If the value is true, the following five Kerberos-related parameters must be specified: ◦ kerberosKeytabFilePath ◦ kerberosPrincipal ◦ hbaseMasterKerberosPrincipal ◦ hbaseRegionserverKerberosPrincipal ◦ hbaseRpcProtection • If the value is false, Kerberos authentication is not required and you do not need to specify the preceding parameters.	No	*false*
hbaseConfig	The properties of the HBase cluster, in JSON format. The hbase.zookeeper.quorum parameter is required. It specifies the ZooKeeper ensemble servers. You can also configure other properties, such as those related to the cache and batch for scan operations.	Yes	None
mode	The mode in which data is written to the HBase data store. Currently, only the normal mode is supported. The dynamic column selection mode is coming soon.	Yes	None
table	The name of the HBase table to which data is written. The name is case-sensitive.	Yes	None
encoding	The encoding format in which a string is converted through byte[]. Currently, UTF-8 and GBK are supported.	No	*utf-8*
column	The HBase columns to which data is written. • index: the ID of the column in the source table, starting from 0. • name: the name of the column in the HBase table, in the columnFamily:column format. • type: the type of the data written, which is used by the byte[] constructor.	Yes	None

Parameter	Description	Required	Default value
maxVersion	The number of versions read by HBase Reader when multiple versions are available. Valid values: -1 and integers greater than 1. A value of -1 indicates that all versions are read.	Required in multiVersionFixedColumn mode	None
range	The rowkey range that HBase Reader reads. • startRowkey: the start rowkey. • endRowkey: the end rowkey. • isBinaryRowkey: the operation called by byte[] to convert the specified start and end rowkeys. Default value: false. If the value is true, Bytes.toBytesBinary(rowkey) is called. If the value is false, Bytes.toBytes(rowkey) is called. Example: `"range": { "startRowkey": "aaa", "endRowkey": "ccc", "isBinaryRowkey":false }` Example: `"column": [ { "index":1, "name": "cf1:q1", "type": "string" }, { "index":2, "name": "cf1:q2", "type": "string" } ]`	No	None

Parameter	Description	Required	Default value
rowkeyColumn	The rowkey of each HBase cell. • index: the ID of the column in the source table, starting from 0. If the column is a constant, set the value to -1. • type: the type of the data written, which is used by the byte[] constructor. • value: a constant, which is usually used as the delimiter between fields. HBase Writer sequentially concatenates all columns specified in this parameter to a string, and uses the string as the rowkey. The specified columns cannot be all constants. Example: `"rowkeyColumn": [ { "index":0, "type":"string" }, { "index":-1, "type":"string", "value":"_" } ]`	Yes	None
versionColumn	The version of each HBase cell. You can use the current time, a specified source column, or a specified time as the version. If you do not specify this parameter, the current time is used. • index: the ID of the column in the source table, starting from 0. Make sure that the value can be properly converted to the Long type. • type: the data type. If the type is Date, HBase Writer converts the date to yyyy-MM-dd HH:mm:ss or yyyy-MM-dd HH:mm:ss SSS. If you want to use a specified time as the version, set the value to -1. • value: the specified time of the Long type. Example: `"versionColumn":{ "index":1 }` `"versionColumn":{ "index":-1, "value":123456789 }`	No	None

Parameter	Description	Required	Default value
nullMode	The method of processing null values. Valid values: • skip: HBase Writer does not write null values to the HBase data store. • empty: HBase Writer writes HConstants.EMPTY_BYTE_ARRAY (new byte [0]) to the HBase data store instead of null values.	No	skip
walFlag	Specifies whether to enable write ahead logging (WAL) for HBase. If the value is true, all edits requested by an HBase client for all Regions carried by the RegionServer are recorded first in the WAL (that is, the HLog). After the edits are successfully recorded in the WAL, they are implemented to the Memstore and a success indication is sent to the HBase client. If edits fail to be recorded in the WAL, a failure indication is sent to the HBase client without implementing the edits. If the value is false, WAL is disabled but writing efficiency is improved.	No	*false*
writeBufferSize	The write buffer size, in bytes, of the HBase client. If you specify this parameter, you must also specify the autoflush parameter. autoflush: • If the value is true, the HBase client sends a PUT request each time it receives an edit. • If the value is false, the HBase client sends a PUT request only when its write buffer is full.	No	8 MB

Configure HBase Writer by using the codeless UI

Currently, the codeless user interface (UI) is not supported for HBase Writer.

Configure HBase Writer by using the code editor

In the following code, a node is configured to write data to an HBase 1.1.x data store.

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"stream",
            "parameter":{},
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"hbase",// The writer type.
            "parameter":{
                "mode":"normal",// The mode in which data is written to the HBase data stor
e.
                "walFlag":"false",// WAL is disabled for HBase.
                "hbaseVersion":"094x",// The HBase version.
```

```
                "rowkeyColumn":[// The rowkey of each HBase cell.
                    {
                        "index":"0",// The ID of the column in the source table.
                        "type":"string"// The data type.
                    },
                    {
                        "index":"-1",
                        "type":"string",
                        "value":"_"
                    }
                ],
                "nullMode":"skip",// The method of processing null values.
                "column":[// The HBase columns to which data is written.
                    {
                        "name":"columnFamilyName1:columnName1",// The name of the HBase col
umn.
                        "index":"0",// The ID of the column in the source table.
                        "type":"string"// The data type.
                    },
                    {
                        "name":"columnFamilyName2:columnName2",
                        "index":"1",
                        "type":"string"
                    },
                    {
                        "name":"columnFamilyName3:columnName3",
                        "index":"2",
                        "type":"string"
                    }
                ],
                "writeMode":"api",// The write mode.
                "encoding":"utf-8",// The encoding format.
                "table":"",// The name of the destination table.
                "hbaseConfig":{// The properties of the HBase cluster, in JSON format.
                    "hbase.zookeeper.quorum":"hostname",
                    "hbase.rootdir":"hdfs: //ip:port/database",
                    "hbase.cluster.distributed":"true"
                }
            },
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0"// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false,// Specifies whether to enable bandwidth throttling. A value o
f false indicates that the bandwidth is not throttled. A value of true indicates that the b
andwidth is throttled. The maximum transmission rate takes effect only if you set this para
meter to true.
            "concurrent":1,// The maximum number of concurrent threads.
        }
```

```
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

3.6.4.7. Configure HBase11xsql Writer

This topic describes the features, data types, and parameters supported by HBase11xsql Writer and how to configure it by using the code editor.

Background information

HBase11xsql Writer allows you to write data in batches to HBase tables created by using Phoenix. Phoenix can encode the primary key to rowkey. If you directly use the HBase API to write data to an HBase table that is created by using Phoenix, you must manually convert data, which is troublesome and error-prone. HBase11xsql Writer allows you to write data to HBase tables that packs all values into a single cell per column family.

HBase11xsql Writer connects to a remote HBase data store by using JDBC, and executes an UPSERT statement to write data to the HBase data store.

Limits

- The column order specified in the writer must match that specified in the reader. When you configure the column order in the reader, you specify the order of columns in each row for the output data. When you configure the column order in the writer, you specify the expected order of columns for the input data. Example:

  Column order specified in the reader: c1, c2, c3, c4.

  Column order specified in the writer: x1, x2, x3, x4.

  In this case, the value of column c1 is assigned to column x1 in the writer. If the column order specified in the writer is x1, x2, x4, x3, the value of column c3 is assigned to column x4 and the value of column c4 is assigned to column x3.
- HBase11xsql Writer can write data only to HBase 1.x.
- HBase11xsql Writer can write data only to tables created by using Phoenix but not native HBase tables.
- HBase11xsql Writer cannot write data with timestamps.

Features

HBase11xsql Writer can write data of an indexed table and synchronously update all indexed tables.

How it works

HBase11xsql Writer connects to an HBase data store by using Phoenix, which is a JDBC driver, and executes an UPSERT statement to write data in batches to the destination table. Phoenix allows to synchronously update indexed tables when you write data.

Parameters

Parameter	Description	Required	Default value
plugin	The writer type. Set this value to hbase11xsql.	Yes	None
table	The name of the destination table. The name is case-sensitive. Generally, the name of a table that is created by using Phoenix consists of uppercase letters.	Yes	None
column	The name of the column. The name is case-sensitive. Generally, the name of each column in a table that is created by using Phoenix consists of uppercase letters. **Note** • HBase11xsql Writer writes data strictly in accordance with the order of the columns obtained from the reader. • You do not need to specify the data type for each column. HBase11xsql Writer automatically obtains the metadata of columns from Phoenix.	Yes	None
hbaseConfig	The properties of the HBase cluster. The hbase.zookeeper.quorum parameter is required. It specifies the ZooKeeper ensemble servers. **Note** • Separate the IP addresses with commas (,), for example, ip1,ip2,ip3. • The zookeeper.znode.parent parameter is optional. Default value: /hbase.	Yes	None
batchSize	The number of data records to write at a time.	No	*256*
nullMode	The method of processing null values. Valid values: • *skip*: HBase11xsql Writer does not write null values to the HBase data store. • *empty*: HBase11xsql Writer writes 0 or an empty string instead of null values to the HBase data store. For a column of the numeric type, HBase11xsql Writer writes 0. For a column of the VARCHAR type, HBase11xsql Writer writes an empty string.	No	*skip*

Configure HBase11xsql Writer by using the code editor

In the following code, a node is configured to write data to an HBasae database.

```
{
  "type": "job",
  "version": "1.0",
  "configuration": {
    "setting": {
      "errorLimit": {
        "record": "0"
      },
      "speed": {
        "mbps": "1",
        "concurrent": "1"
      }
    },
    "reader": {
      "plugin": "odps",
      "parameter": {
        "datasource": "",
        "table": "",
        "column": [],
        "partition": ""
      }
    },
    "plugin": "hbase11xsql",
    "parameter": {
      "table": "The case-sensitive name of the destination table",
      "hbaseConfig": {
        "hbase.zookeeper.quorum": "The IP addresses of ZooKeeper ensemble servers of the de
stination HBase cluster. Obtain the IP addresses from product engineers (PEs).",
        "zookeeper.znode.parent": "The root znode of the destination HBase cluster. Obtain
the IP addresses from PEs."
      },
      "column": [
        "columnName"
      ],
      "batchSize": 256,
      "nullMode": "skip"
    }
  }
}
```

FAQ

Q: What is the proper number of concurrent threads? Can I increase the number of concurrent threads to speed up the synchronization?

A: In the data import process, the default size of a JVM heap is 2 GB. Concurrent synchronization requires multiple threads. However, excessive threads sometimes cannot speed up the synchronization and may even deteriorate the performance because of frequent garbage collection (GC). We recommend that you set the number of concurrent threads within the range from 5 to 10.

Q: What is the proper value for the batchSize parameter?

A: The default value of the batchSize parameter is 256. You can set a proper value for the batchSize parameter based on the data volume of each row. Generally, the data volume of each write operation is 2 MB to 4 MB. You can set the value to the data volume of a write operation divided by the data volume of a row.

3.6.4.8. Configure HDFS Writer

This topic describes the data types and parameters supported by HDFS Writer and how to configure it by using the code editor.

HDFS Writer allows you to write text, ORC, or Parquet files to the specified directory in HDFS. In addition, you can associate the fields in the files with those in Hive tables. You must configure a connection before you configure HDFS Writer.

How it works

HDFS Writer writes files to HDFS in the following way:

1. Creates a temporary directory that does not exist in HDFS based on the path parameter you specified.

   The name of the temporary directory is in the format of path_Random suffix.
2. Writes files that are read by a Data Integration reader to the temporary directory.
3. Moves the files from the temporary directory to the specified directory in HDFS after all the files are written. HDFS Writer ensures that the file names do not conflict with existing files in HDFS when it moves the files.
4. Deletes the temporary directory. If the deletion is interrupted because HDFS Writer fails to connect to HDFS, you must manually delete the temporary directory.

> **Note** To synchronize data, use an administrator account with the read and write permissions.

Limits

- HDFS Writer can write only text, ORC, and Parquet files that store logical two-dimensional tables to HDFS.
- HDFS is a distributed file system and does not have a schema. Therefore, you cannot write only some of the columns in a file to HDFS.
- HDFS Writer supports only the following Hive data types:
  - Numeric: TINYINT, SMALLINT, INT, BIGINT, FLOAT, and DOUBLE
  - String: STRING, VARCHAR, and CHAR
  - Boolean: BOOLEAN
  - Date and time: DATE and TIMESTAMP
- HDFS Writer does not support other Hive data types, such as DECIMAL, BINARY, ARRAY, MAP, STRUCT, or UNION.
- HDFS Writer can write data to only one partition in a partitioned Hive table at a time.
- To write a text file to HDFS, make sure that the delimiter in the file is the same as that in the Hive table to be associated with the file. Otherwise, you cannot associate the fields in the file stored in HDFS with those in the Hive table.
- HDFS Writer can be used in the environment where Hive 1.1.1 and Hadoop 2.7.1 (JDK version: 1.7) are

installed. HDFS Writer can write files to HDFS properly in testing environments where Hadoop 2.5.0, Hadoop 2.6.0, or Hive 1.2.0 is installed.

Data types

HDFS Writer supports most Hive data types. Make sure that your data types are supported.

The following table lists the Hive data types supported by HDFS Writer.

> **Note** The types of the specified columns must be the same as those of columns in the Hive table.

Category	Hive data type
Integer	TINYINT, SMALLINT, INT, and BIGINT
Floating point	FLOAT and DOUBLE
String	CHAR, VARCHAR, and STRING
Boolean	BOOLEAN
Date and time	DATE and TIMESTAMP

Parameters

Parameter	Description	Required	Default value
defaultFS	The address of the HDFS NameNode, for example, `hdfs://127.0.0.1:9000`. The default resource group does not support configuring advanced Hadoop parameters related to the high availability feature.	Yes	None
fileType	The format of the files to be written to HDFS. Valid values: • *text*: the text file format. • *orc*: the ORC file format. • *parquet*: the common Parquet file format.	Yes	None

Parameter	Description	Required	Default value
path	The directory in HDFS to which the files are written. HDFS Writer concurrently writes multiple files to the directory based on the concurrency setting. To associate the fields in a file with those in a Hive table, set the path parameter to the storage path of the Hive table in HDFS. Assume that the storage path specified for the data warehouse of Hive is `/user/hive/warehouse/`. The storage path of the hello table created in the test database is `/user/hive/warehouse/test.db/hello`.	Yes	None
fileName	The name prefix of the files to be written to HDFS. A random suffix is appended to the specified prefix to form the actual file name used by each thread.	Yes	None
column	The columns to be written to HDFS. You cannot write only some of the columns in a file to HDFS. To associate the fields in a file with those in a Hive table, specify the name and type parameters for each field. You can also specify the column parameter in the following way: `"column": [ { "name": "userName", "type": "string" }, { "name": "age", "type": "long" } ]`	Yes (Not required if the fileType parameter is set to parquet)	None

Parameter	Description	Required	Default value
writeMode	The mode in which HDFS Writer writes the files. Valid values: • *append*: writes the files based on the specified file name prefix and ensures that the actual file names do not conflict with those of existing files. • *nonConflict*: returns an error if a file with the specified file name prefix exists in the destination directory. **Note** Parquet files do not support the append mode. They support only the nonConflict mode.	Yes	None
fieldDelimiter	The column delimiter used in the files to be written to HDFS. Make sure that you use the same delimiter as that in the Hive table. Otherwise, you cannot query data in the Hive table.	Yes (Not required if the fileType parameter is set to parquet)	None
compress	The compression format of the files to be written to HDFS. By default, this parameter is left empty, that is, files are not compressed. For a text file, the GZIP and BZIP2 compression formats are supported. For an ORC file, the SNAPPY compression format is supported. To compress an ORC file, you must install SnappyCodec.	No	None
encoding	The encoding format of the files to be written to HDFS.	No	None

Parameter	Description	Required	Default value
parquetSchema	The schema of the files to be written to HDFS. This parameter is required only when the fileType parameter is set to parquet. Format: `message messageTypeName {` `required, dataType, columnName;` `..... ;` `}` Parameter description: • messageTypeName: the name of the MessageType object. • required: specifies whether the field is required. We recommend that you set the parameter to optional for all fields. • dataType: the type of the field. Valid values: BOOLEAN, INT32, INT64, INT96, FLOAT, DOUBLE, BINARY, and FIXED_LEN_BYTE_ARRAY. Set this parameter to BINARY if the field stores strings. **Note** Each line, including the last one, must end with a semicolon (;). Example: `message m {` `optional int64 id;` `optional int64 date_id;` `optional binary datetimestring;` `optional int32 dspId;` `optional int32 advertiserId;` `optional int32 status;` `optional int64 bidding_req_num;` `optional int64 imp;` `optional int64 click_num;` `}`	No	None

Parameter	Description	Required	Default value
hadoopConfig	The advanced parameter settings of Hadoop, such as those related to high availability. The default resource group does not support configuring advanced Hadoop parameters related to the high availability feature. `"hadoopConfig":{ "dfs.nameservices": "testDfs", "dfs.ha.namenodes.testDfs": "namenode1,namenode2", "dfs.namenode.rpc-address.youkuDfs.namenode1": "", "dfs.namenode.rpc-address.youkuDfs.namenode2": "", "dfs.client.failover.proxy.provider.testDfs": "org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider" }`	No	None
	The synchronization mode for Parquet files. If the dataxParquetMode parameter is set to fields, you can write data of complex types, such as ARRAY, MAP, and STRUCT. Valid values: fields and columns. If the dataxParquetMode parameter is set to fields, HDFS Writer supports HDFS over OSS. HDFS uses OSS as the storage service and HDFS Writer writes Parquet files to OSS. In this case, you can add the following OSS-related parameters in the hadoopConfig parameter: • fs.oss.accessKeyId: the AccessKey ID for connecting to OSS. • fs.oss.accessKeySecret: the AccessKey secret for connecting to OSS. • fs.oss.endpoint: the endpoint for connecting to OSS. Example:		

Parameter	Description	Required	Default value
dataxParquetMode	```json "writer": { "name": "hdfswriter", "parameter": { "defaultFS": "oss://test-bucket", "fileType": "parquet", "path": "/datasets/oss_demo/kpt", "fileName": "test", "writeMode": "truncate", "compress": "SNAPPY", "encoding": "UTF-8", "hadoopConfig": { "fs.oss.accessKeyId": "the-access-id", "fs.oss.accessKeySecret": "the-access-key", "fs.oss.endpoint": "oss-cn-hangzhou.aliyuncs.com" }, "parquetSchema": "message test {\n required int64 id;\n optional binary name (UTF8);\n optional int64 gmt_create;\n required group map_col (MAP) {\n repeated group key_value {\n required binary key (UTF8);\n required binary value (UTF8);\n }\n }\n required group array_col (LIST) {\n repeated group list {\n required binary element (UTF8);\n }\n }\n required group struct_col {\n required int64 id;\n required binary name (UTF8);\n }\n}", "dataxParquetMode": "fields" } } ```	No	*columns*

Parameter	Description	Required	Default value
haveKerberos	Specifies whether Kerberos authentication is required. Default value: *false*.	If you set this parameter to *true*, you must also set the kerberosKeytabFilePath and kerberosPrincipal parameters.	*false*
kerberosKeytabFilePath	The absolute path of the keytab file for Kerberos authentication.	Required if the haveKerberos parameter is set to true	None
kerberosPrincipal	The Kerberos principal to which Kerberos can assign tickets. Example: ****/hadoopclient@**.***. **Note** The absolute path of the keytab file is required for Kerberos authentication. Therefore, you can configure Kerberos authentication only on a custom resource group. Example: `"haveKerberos":true, "kerberosKeytabFilePath":"/opt/datax/**.keytab", "kerberosPrincipal":"**/hadoopclient@**. **"`	Required if the haveKerberos parameter is set to true	None

Configure HDFS Writer by using the codeless UI

The codeless UI is not supported for HDFS Writer.

Configure HDFS Writer by using the code editor

In the following code, a node is configured to write files to HDFS. For more information about the parameters, see the preceding parameter description.

```
{
    "type": "job",
    "version": "2.0",// The version number.
    "steps": [
        {
            "stepType": "stream",
            "parameter": {},
            "name": "Reader",
            "category": "reader"
        },
        {
            "stepType": "hdfs",// The writer type.
            "parameter": {
```

```
                "path": "",// The directory in HDFS to which the files are written.
                "fileName": "",// The name prefix of the files to be written to HDFS.
                "compress": "",// The compression format of the files.
                "datasource": "",// The connection name.
                "column": [
                    {
                        "name": "col1",// The name of the column.
                        "type": "string"// The data type of the column.
                    },
                    {
                        "name": "col2",
                        "type": "int"
                    },
                    {
                        "name": "col3",
                        "type": "double"
                    },
                    {
                        "name": "col4",
                        "type": "boolean"
                    },
                    {
                        "name": "col5",
                        "type": "date"
                    }
                ],
                "writeMode": "",// The write mode.
                "fieldDelimiter": ",",// The column delimiter.
                "encoding": "",// The encoding format.
                "fileType": "text"// The file format.
            },
            "name": "Writer",
            "category": "writer"
        }
    ],
    "setting": {
        "errorLimit": {
            "record": ""// The maximum number of dirty data records allowed.
        },
        "speed": {
            "concurrent": 3,// The maximum number of concurrent threads.
            "throttle": false // Specifies whether to enable bandwidth throttling. A value
of false indicates that the bandwidth is not throttled. A value of true indicates that the
bandwidth is throttled. The maximum transmission rate takes effect only if you set this par
ameter to true.
        }
    },
    "order": {
        "hops": [
            {
                "from": "Reader",
                "to": "Writer"
            }
        ]
```

```
        }
}
```

3.6.4.9. Configure MaxCompute Writer

This topic describes the data types and parameters supported by MaxCompute Writer and how to configure it by using the codeless UI and code editor.

MaxCompute Writer is designed for developers to insert data to or update data in MaxCompute. MaxCompute Writer is suitable for importing data at the GB or TB level to MaxCompute.

> **Note** You must configure a connection before you configure MaxCompute Writer.

Based on the specified information such as the source project, table, partition, and field, MaxCompute Writer writes data to MaxCompute by using Tunnel.

Data types

The following table lists the data types supported by MaxCompute Writer.

Category	MaxCompute data type
Integer	BIGINT
Floating point	DOUBLE and DECIMAL
String	STRING
Date and time	DATETIME
Boolean	BOOLEAN

Parameters

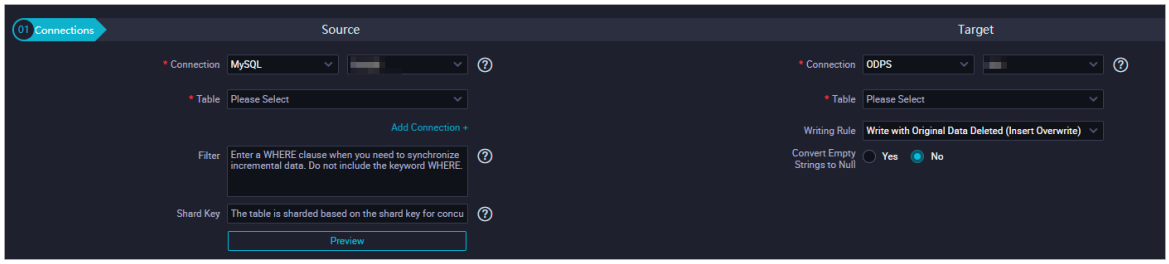
Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None
table	The name of the destination table. The name is not case-sensitive. You can specify only one table as the destination table.	Yes	None

Parameter	Description	Required	Default value
partition	The partition to which data is written. The last-level partition must be specified. For example, if you want to write data to a three-level partitioned table, set the partition parameter to a value that contains the third-level partition information, for example, `pt=20150101, type=1, biz=2`. • To write data to a non-partitioned table, do not set this parameter. The data is directly written to the destination table. • MaxCompute Writer does not support writing data based on the partition route. To write data to a partitioned table, make sure that data is written to the last-level partition.	Required only for writing data to a partitioned table	None
column	The columns in the destination table to which data is written. To write data to all the columns in the destination table, set the value to an asterisk (*), for example, `"column":["*"]`. Set the value to the specified columns if data is written to only some of the columns in the destination table. Separate the columns with commas (,). Example: `"column":["id","name"]`. • MaxCompute Writer can filter columns and change the order of columns. For example, a MaxCompute table has three columns: a, b, and c. If you want to write data only to column c and column b, you can set the column parameter in the format `"column":["c","b"]`. During data synchronization, column a is automatically set to null. • The column parameter must explicitly specify a set of columns to which data is written. The parameter cannot be left empty.	Yes	None
truncate	To ensure the idempotence of write operations, set the truncate parameter in the format `"truncate":"true"`. When a failed sync node is rerun due to a write failure, MaxCompute Writer deletes the data that has been written before it imports the source data again. This ensure that the same data is written for each rerun. MaxCompute Writer uses MaxCompute SQL to delete data. MaxCompute SQL cannot ensure the atomicity. Therefore, the truncation operation is not an atomic operation. Conflicts may occur when concurrent nodes delete data from the same table or partition. To avoid this issue, we recommend that you do not run concurrent DDL nodes to write data to the same partition. You can create different partitions for nodes that need to run concurrently.	Yes	None

Configure MaxCompute Writer by using the codeless UI

1. Configure the connections.

Configure the connections to the source and destination data stores for the sync node.



Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and then select a connection name that you have configured in DataWorks.
Table	The table parameter in the preceding parameter description.
Partition Key Column	To write data to all the columns in the destination table, set "column": ["*"] for the column parameter in the preceding parameter description. The partition parameter allows you to use wildcards and specify one or more partitions. ○ `"partition":"pt=20140501/ds=*"` specifies that data is written to all ds partitions with pt=20140501. ○ `"partition":"pt=top?"` specifies that data is written to the partitions with pt=top and pt=to. You can specify the partition key columns to which data is written. Assume that the partition key column of a MaxCompute table is pt=${bdp.system.bizdate}. You can configure the column to which data is written to pt. Ignore it if the column is marked as unidentified. ○ To write data to all partitions, enter pt=*. ○ To write data to some of the partitions, specify the corresponding dates.

Parameter	Description
Writing Rule	◦ **Write with Original Data Deleted (Insert Overwrite)**: All data in the table or partition is deleted before data import. This rule is equivalent to the `INSERT OVERWRITE` statement. ◦ **Write with Original Data Retained (Insert Into)**: No data is deleted before data import. New data is always appended upon each run. This rule is equivalent to the `INSERT INTO` statement. **Note** ◦ MaxCompute Reader reads data by using Tunnel. Sync nodes do not support data filtering. Instead, they must read all the data in a specific table or partition. ◦ MaxCompute Writer writes data by using Tunnel instead of the INSERT INTO statement. You can view the complete data in the destination table only after a sync node is run. Pay attention to the node dependencies.
Convert Empty Strings to Null	Specifies whether to convert empty strings to null. Default value: *No*.

2. Configure field mapping. It is equivalent to setting the column parameter in the preceding parameter description. Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right.

GUI element	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	Click Auto Layout to sort the fields based on specified rules.

3. Configure channel control policies.

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read data from and write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.

Parameter	Description
Dirty Data Records Allowed	The maximum number of dirty data records allowed.
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure MaxCompute Writer by using the code editor

In the following code, a node is configured to write data to a MaxCompute project. For more information about the parameters, see the preceding parameter description.

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"stream",
            "parameter":{},
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"odps",// The writer type.
            "parameter":{
                "partition":"",// The partition information.
                "truncate":true,// The write rule.
                "compress":false,// Specifies whether to enable compression.
                "datasource":"odps_first",// The connection name.
            "column":[// The columns to which data is written.
                "id",
                "name",
                "age",
                "sex",
                "salary",
                "interest"
                ],
                "emptyAsNull":false,// Specifies whether to convert empty strings to null.
                "table":""// The name of the destination table.
            },
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0"// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false,// Specifies whether to enable bandwidth throttling. A value o
f false indicates that the bandwidth is not throttled. A value of true indicates that the b
andwidth is throttled. The maximum transmission rate takes effect only if you set this para
meter to true.
            "concurrent":1 // The maximum number of concurrent threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

Additional instructions

- Column filter

  By configuring MaxCompute Writer, you can perform operations that MaxCompute does not support, for example, filter columns, reorder columns, and set empty fields to null. To write data to all the columns in the destination table, set the column parameter in the format `"column":["*"]`.

  For example, a MaxCompute table has three columns: a, b, and c. If you want to write data only to column c and column b, you can set the column parameter in the format `"column":["c","b"]`. The first column and the second column of the source data are written to column c and column b in the MaxCompute table respectively. During data synchronization, column a is automatically set to null.

- Column configuration error handling

  To avoid losing the data of redundant columns and ensure high data reliability, MaxCompute Writer returns an error message if the number of columns to be written is more than that in the destination table. For example, if a MaxCompute table contains columns a, b, and c, MaxCompute Writer returns an error message if more than three columns are to be written to the table.

- Partition configuration

  MaxCompute Writer can write data only to the last-level partition, and cannot write data to the specified partition based on a field. To write data to a partitioned table, specify the last-level partition. For example, if you want to write data to a three-level partitioned table, set the partition parameter to a value that contains the third-level partition information, for example, `pt=20150101, type=1, biz=2`. The data cannot be written if you set the partition parameter to `pt=20150101, type=1` or `pt=20150101`.

- Node rerunning

  To ensure the idempotence of write operations, set the `truncate` parameter to true. When a failed sync node is rerun due to a write failure, MaxCompute Writer deletes the data that has been written before it imports the source data again. This ensures that the same data is written for each rerun. If a sync node is interrupted due to other exceptions, the data cannot be rolled back and the node cannot be rerun automatically. You can ensure the idempotence of write operations and the data integrity By setting the truncate parameter to true.

  > **Note** If the truncate parameter is set to true, all data of the specified partition or table is deleted before a rerun. Exercise caution when you set this parameter to true.

3.6.4.10. Configure Memcache Writer

This topic describes the data types and parameters supported by Memcache Writer and how to configure it by using the code editor.

ApsaraDB for Memcache is a distributed in-memory database service with high performance, reliability, and scalability. Based on the Apsara distributed operating system and high-performance storage technologies, ApsaraDB for Memcache provides a complete database solution with hot standby, fault recovery, business monitoring, and data migration features.

ApsaraDB for Memcache is immediately available after an instance is created. It relieves the load on databases from dynamic websites and applications by caching data in the memory and therefore improves the response speed of websites and applications.

Same as on-premises Memcached databases, ApsaraDB for Memcache databases are compatible with the Memcached protocol. ApsaraDB for Memcache databases can be directly used in your environments. The difference is that the data, hardware infrastructure, network security, and system maintenance services used by ApsaraDB for Memcache databases are all deployed in the cloud. These services are billed based on the pay-as-you-go billing method.

Memcache Writer writes data to ApsaraDB for Memcache databases based on the Memcached protocol.

Memcache Writer writes data only in text format. The method of converting data types varies based on the format of writing data.

- text: Memcache Writer uses the specified column delimiter to serialize source data to a string.
- binary: This format is not supported.

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None
writeMode	The write mode. Valid values: • *set*: stores the source data. • *add*: stores the source data only when its key does not exist in the destination ApsaraDB for Memcache database. This mode is not supported now. • *replace*: uses the source data to replace the data record with the same key in the destination ApsaraDB for Memcache database. This mode is not supported now. • *append*: adds the value of the source data to the end of the value of an existing data record with the same key in the destination ApsaraDB for Memcache database, but does not update the expiration time of the existing data record. This mode is not supported now. • *prepend*: adds the value of the source data to the beginning of the value of an existing data record with the same key in the destination ApsaraDB for Memcache database, but does not update the expiration time of the existing data record. This mode is not supported now.	Yes	None

Parameter	Description	Required	Default value
writeFormat	The format in which Memcache Writer writes the source data. Currently, only the text format is supported. text: serializes the source data to the text format. Memcache Writer uses the first column of the source data as the key and serializes the subsequent columns to the value by using the specified delimiter. Then, Memcache Writer writes the key-value pair to ApsaraDB for Memcache. Assume that the following source data exists: `| ID | NAME | COUNT|` `| ---- |:------|:-----|` `| 23 | "CDP" | 100 |` If you set the column delimiter to a backslash and a caret (\^), data is written to ApsaraDB for Memcache in the following format: `| KEY (OCS) | VALUE(OCS) |` `| --------- |:---------- |` `| 23 | CDP\^100 |`	No	None
expireTime	The expiration time of the source data to be cached in ApsaraDB for Memcache. ApsaraDB for Memcache supports the following two types of expiration time: • unixtime: the UNIX timestamp, indicating a specific time point in the future when the data expires. The UNIX timestamp represents the number of seconds that have elapsed since 00:00:00 on January 1, 1970. • seconds: the relative time in seconds starting from the current time point. It specifies the period during which data is valid. **Note** If the specified time exceeds 30 days, the server identifies the time as the UNIX timestamp.	No	*0*, indicating that the data never expires
batchSize	The number of data records to write at a time. Setting this parameter can greatly reduce the interactions between Data Integration and the ApsaraDB for Memcache database over the network, and increase the throughput. However, an excessively large value may lead to the OOM error during the data synchronization process.	No	*1,024*

Configure Memcache Writer by using the codeless UI

The codeless UI is not supported for Memcache Writer.

Configure Memcache Writer by using the code editor

In the following code, a node is configured to write data to an ApsaraDB for Memcache database.

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"stream",
            "parameter":{},
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"ocs",// The writer type.
            "parameter":{
                "writeFormat":"text",// The format in which Memcache Writer writes the sour
ce data.
                "expireTime":1000,// The expiration time of the source data to be cached in
ApsaraDB for Memcache.
                "indexes":0,
                "datasource":"",// The connection name.
                "writeMode":"set",// The write mode.
                "batchSize":"256"// The number of data records to write at a time.
            },
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0" // The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false,// Specifies whether to enable bandwidth throttling. A value o
f false indicates that the bandwidth is not throttled. A value of true indicates that the b
andwidth is throttled. The maximum transmission rate takes effect only if you set this para
meter to true.
            "concurrent":1 // The maximum number of concurrent threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

3.6.4.11. Configure MongoDB Writer

This topic describes the data types and parameters supported by MongoDB Writer and how to configure it by using the code editor.

MongoDB Writer connects to a remote MongoDB database by using the Java client named MongoClient and writes data to the database. The latest version of MongoDB has improved the locking feature from database locks to document locks. The powerful index functionalities of MongoDB enable MongoDB Writer to efficiently write data to MongoDB databases. If you want to update data, specify the primary key.

> **Note**
> - You must configure a connection before you configure MongoDB Writer.
> - If you use ApsaraDB for MongoDB, the MongoDB database has a root account by default.
> - For security concerns, Data Integration only supports access to a MongoDB database by using a MongoDB database account. When you add a MongoDB connection, do not use the root account for access.

MongoDB Writer obtains data from a Data Integration reader, and converts the data types to those supported by MongoDB. Data Integration does not support arrays. MongoDB supports arrays and the array index is useful.

To use MongoDB arrays, you can convert strings to MongoDB arrays by configuring a parameter and write the arrays to a MongoDB database.

Data types

MongoDB Writer supports most MongoDB data types. Make sure that your data types are supported.

The following table lists the data types supported by MongoDB Writer.

Category	MongoDB data type
Integer	INT and LONG
Floating point	DOUBLE
String	STRING and ARRAY
Date and time	DATE
Boolean	BOOL
Binary	BYTES

> **Note** When data of the DATE type is written to a MongoDB database, the type of the data is converted to DATETIME.

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None
collectionName	The name of the MongoDB collection.	Yes	None
column	The columns in MongoDB. • name: the name of the column. • type: the data type of the column. • splitter: the delimiter. Specify this field only when you want to convert the string to an array. The string is split based on the specified delimiter, and the split strings are saved in a MongoDB array.	Yes	None
writeMode	Specifies whether to overwrite data. • isReplace: If you set this parameter to true, MongoDB Writer overwrites the data in the destination table with the same primary key. If you set this parameter to false, the data is not overwritten. • replaceKey: the primary key for each record. Data is overwritten based on this primary key. The primary key must be unique.	No	None

Parameter	Description	Required	Default value
preSql	The action to perform before the sync node is run. For example, you can clear outdated data before data synchronization. If the preSql parameter is left empty, no action is performed before data synchronization. Make sure that the value of the preSql parameter complies with the JSON syntax. The format requirements for the preSql parameter are as follows: • Configure the type field to specify the action type. Valid values: drop and remove. Example: `"preSql":{"type":"remove"}`. ◦ *drop*: deletes the collection specified by the collectionName parameter and the data in the collection. ◦ *remove*: deletes data based on conditions. ◦ *json*: the conditions for deleting data. Example: `"preSql":{"type":"remove", "json":"{'operationTime':{'$gte':ISODate('${last_day}T00:00:00.424+0800')}}"}`. In the preceding JSON string, `${last_day}` is a scheduling parameter of DataWorks. The format is `$[yyyy-mm-dd]`. You can use comparison operators (such as $gt, $lt, $gte, and $lte), logical operators (such as $and and $or), and functions (such as max, min, sum, avg, and ISODate) supported by MongoDB as needed. For more information, see the MongoDB query syntax. Data Integration uses the following standard MongoDB API to query and delete the specified data: `query=(BasicDBObject) com.mongodb.util.JSON.parse(json);` `col.deleteMany(query);` **Note** If you want to delete data based on conditions, we recommend that you specify the conditions in JSON format preferentially. ◦ *item*: the name, condition, and value for filtering data. Example: `"preSql":{"type":"remove","item":[{"name":"pv","value":"100","condition":"$gt"},{"name":"pid","value":"10"}]}`. Data Integration sets query conditions based on the value of the item field and deletes data by using the standard MongoDB API. Example: `col.deleteMany(query);`. • If the value of the preSql parameter cannot be recognized, no action is performed.	No	None

Configure MongoDB Writer by using the codeless UI

The codeless UI is not supported for MongoDB Writer.

Configure MongoDB Writer by using the code editor

In the following code, a node is configured to write data to a MongoDB database. For more information about the parameters, see the preceding parameter description.

```
{
    "type": "job",
    "version": "2.0",// The version number.
    "steps": [
        {
            "stepType": "stream",
            "parameter": {},
            "name": "Reader",
            "category": "reader"
        },
        {
            "stepType": "mongodb",// The writer type.
            "parameter": {
                "datasource": "",// The connection name.
                "column": [
                    {
                        "name": "_id",// The name of the column to which data is written.
                        "type": "ObjectId"// The data type of the column to which data is w
ritten. If the replacekey parameter is set to _id, set the type parameter to ObjectId. If y
ou set the type parameter to String, the data cannot be overwritten.
                    },
                    {
                        "name": "age",
                        "type": "int"
                    },
                    {
                        "name": "id",
                        "type": "long"
                    },
                    {
                        "name": "wealth",
                        "type": "double"
                    },
                    {
                        "name": "hobby",
                        "type": "array",
                        "splitter": " "
                    },
                    {
                        "name": "valid",
                        "type": "boolean"
                    },
                    {
                        "name": "date_of_join",
                        "format": "yyyy-MM-dd HH:mm:ss",
                        "type": "date"
                    }
                ],
                "writeMode": {// The write mode.
```

```
                    "isReplace": "true",
                    "replaceKey": "_id"
                },
                "collectionName": "datax_test"// The name of the MongoDB collection.
            },
            "name": "Writer",
            "category": "writer"
        }
    ],
    "setting": {
        "errorLimit": {// The maximum number of dirty data records allowed.
            "record": "0"
        },
        "speed": {
            "jvmOption": "-Xms1024m -Xmx1024m",
            "throttle": true,// Specifies whether to enable bandwidth throttling. A value o
f false indicates that the bandwidth is not throttled. A value of true indicates that the b
andwidth is throttled. The maximum transmission rate takes effect only if you set this para
meter to true.
            "concurrent":1,// The maximum number of concurrent threads.
            "mbps": "1"// The maximum transmission rate.
        }
    },
    "order": {
        "hops": [
            {
                "from": "Reader",
                "to": "Writer"
            }
        ]
    }
}
```

3.6.4.12. Configure MySQL Writer

This topic describes the data types and parameters supported by MySQL Writer and how to configure it by using the codeless UI and code editor.

MySQL Writer allows you to write data to tables stored in MySQL databases. MySQL Writer connects to a remote MySQL database by using JDBC, and executes the `INSERT INTO` or `REPLACE INTO` statement to write data to the MySQL database. MySQL uses the InnoDB engine so that data is written to the database in batches.

> **Note**
>
> - You must configure a connection before you configure MySQL Writer.
> - MySQL Writer does not support MySQL 8.0 or later.

MySQL Writer can be used as a data migration tool by users such as DBAs. MySQL Writer obtains data from a Data Integration reader, and writes the data to the destination database based on value of the writeMode parameter.

> **Note** A sync node that uses MySQL Writer must have at least the permission to execute the `INSERT INTO or REPLACE INTO` statement. Whether other permissions are required depends on the SQL statements specified in the preSql and postSql parameters when you configure the node.

Data types

MySQL Writer supports most MySQL data types. Make sure that your data types are supported.

The following table lists the data types supported by MySQL Writer.

Category	MySQL data type
Integer	INT, TINYINT, SMALLINT, MEDIUMINT, BIGINT, and YEAR
Floating point	FLOAT, DOUBLE, and DECIMAL
String	VARCHAR, CHAR, TINYTEXT, TEXT, MEDIUMTEXT, and LONGTEXT
Date and time	DATE, DATETIME, TIMESTAMP, and TIME
Boolean	BOOL
Binary	TINYBLOB, MEDIUMBLOB, BLOB, LONGBLOB, and VARBINARY

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None
table	The name of the destination table.	Yes	None
writeMode	The write mode. Valid values: • *insert into*: If a primary key conflict or unique index conflict occurs, data cannot be written to the conflicting rows and is regarded as dirty data. • *on duplicate key update*: If no primary key conflict or unique index conflict occurs, the action is the same as that of `insert into`. If a conflict occurs, specified fields in original rows are updated. • *replace into*: If no primary key conflict or unique index conflict occurs, the action is the same as that of `insert into`. If a conflict occurs, original rows are deleted and new rows are inserted. This means that all the field values of the original rows are replaced.	No	*insert*

Parameter	Description	Required	Default value
column	The columns in the destination table to which data is written. Separate the columns with commas (,), for example, `"column":["id","name","age"]`. To write data to all the columns in the destination table, set the value to an asterisk (*), for example, `"column":["*"]`.	Yes	None
preSql	The SQL statement to execute before the sync node is run. For example, you can clear outdated data before data synchronization. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor. **Note** If you specify multiple SQL statements in the code editor, the system may not execute them in the same transaction.	No	None
postSql	The SQL statement to execute after the sync node is run. For example, you can add a timestamp after data synchronization. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor. **Note** If you specify multiple SQL statements in the code editor, the system may not execute them in the same transaction.	No	None
batchSize	The number of data records to write at a time. Setting this parameter can greatly reduce the interactions between Data Integration and the MySQL database over the network, and increase the throughput. However, an excessively large value may lead to the OOM error during the data synchronization process.	No	*1,024*

Configure MySQL Writer by using the codeless UI

1. Configure the connections.

   Configure the connections to the source and destination data stores for the sync node.

Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and then select a connection name that you have configured in DataWorks.
Table	The table parameter in the preceding parameter description.
Statement Run Before Writing	The preSql parameter in the preceding parameter description. Enter an SQL statement to execute before the sync node is run.

Parameter	Description
Statement Run After Writing	The postSql parameter in the preceding parameter description. Enter an SQL statement to execute after the sync node is run.
Solution to Primary Key Violation	The writeMode parameter in the preceding parameter description. Select the required write mode.

2. Configure field mapping. It is equivalent to setting the column parameter in the preceding parameter description. Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right.

GUI element	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	Click Auto Layout to sort the fields based on specified rules.

3. Configure channel control policies.

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read data from and write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure MySQL Writer by using the code editor

In the following code, a node is configured to write data to a MySQL database. For more information about the parameters, see the preceding parameter description.

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"stream",
            "parameter":{},
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"mysql",// The writer type.
            "parameter":{
                "postSql":[],// The SQL statement to execute after the sync node is run.
                "datasource":"",// The connection name.
                "column":[// The columns to which data is written.
                    "id",
                    "value"
                ],
                "writeMode":"insert",// The write mode.
                "batchSize":1024,// The number of data records to write at a time.
                "table":"",// The name of the destination table.
                "preSql":[]// The SQL statement to execute before the sync node is run.
            },
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{// The maximum number of dirty data records allowed.
            "record":"0"
        },
        "speed":{
            "throttle":false,// Specifies whether to enable bandwidth throttling.
            "concurrent":1 // The maximum number of concurrent threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

3.6.4.13. Configure Oracle Writer

This topic describes the data types and parameters supported by Oracle Writer and how to configure it by using the codeless UI and code editor.

Oracle Writer allows you to write data to tables stored in primary Oracle databases. Oracle Writer connects to a remote Oracle database by using JDBC, and executes an `INSERT INTO` statement to write data to the Oracle database.

> **Note** You must configure a connection before you configure Oracle Writer.

Oracle Writer is designed for ETL developers to import data from data warehouses to Oracle databases. Oracle Writer can also be used as a data migration tool by users such as DBAs.

Oracle Writer obtains data from a Data Integration reader, connects to a remote Oracle database by using JDBC, and then executes an SQL statement to write data to the Oracle database.

Data types

Oracle Writer supports most Oracle data types. Make sure that your data types are supported.

The following table lists the data types supported by Oracle Writer.

Category	Oracle data type
Integer	NUMBER, ROWID, INTEGER, INT, and SMALLINT
Floating point	NUMERIC, DECIMAL, FLOAT, DOUBLE PRECISION, and REAL
String	LONG, CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, CLOB, NCLOB, CHARACTER, CHARACTER VARYING, CHAR VARYING, NATIONAL CHARACTER, NATIONAL CHAR, NATIONAL CHARACTER VARYING, NATIONAL CHAR VARYING, and NCHAR VARYING
Date and time	TIMESTAMP and DATE
Boolean	BIT and BOOLEAN
Binary	BLOB, BFILE, RAW, and LONG RAW

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None
table	The name of the destination table.	Yes	None

Parameter	Description	Required	Default value
writeMode	The write mode. Valid values: *insert into*, *on duplicate key update*, and *replace into*. • *insert into*: If a primary key conflict or unique index conflict occurs, data cannot be written to the conflicting rows and is regarded as dirty data. • *on duplicate key update*: If no primary key conflict or unique index conflict occurs, the action is the same as that of `insert into`. If a conflict occurs, specified fields in original rows are updated. • *replace into*: If no primary key conflict or unique index conflict occurs, the action is the same as that of `insert into`. If a conflict occurs, original rows are deleted and new rows are inserted. This means that all the field values of the original rows are replaced.	No	*insert*
column	The columns in the destination table to which data is written. Separate the columns with commas (,), for example, `"column": ["id","name","age"]`. To write data to all the columns in the destination table, set the value to an asterisk (*), for example, `"column":["*"]`.	Yes	None
preSql	The SQL statement to execute before the sync node is run. For example, you can clear outdated data before data synchronization. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor.	No	None
postSql	The SQL statement to execute after the sync node is run. For example, you can add a timestamp after data synchronization. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor.	No	None
batchSize	The number of data records to write at a time. Setting this parameter can greatly reduce the interactions between Data Integration and the Oracle database over the network, and increase the throughput. However, an excessively large value may lead to the OOM error during the data synchronization process.	No	*1,024*

Configure Oracle Writer by using the codeless UI

1. Configure the connections.

   Configure the connections to the source and destination data stores for the sync node.

Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and then select a connection name that you have configured in DataWorks.
Table	The table parameter in the preceding parameter description.

Parameter	Description
Statement Run Before Writing	The preSql parameter in the preceding parameter description. Enter an SQL statement to execute before the sync node is run.
Statement Run After Writing	The postSql parameter in the preceding parameter description. Enter an SQL statement to execute after the sync node is run.

2. Configure field mapping. It is equivalent to setting the column parameter in the preceding parameter description. Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right.

GUI element	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	Click Auto Layout to sort the fields based on specified rules.

3. Configure channel control policies.

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read data from and write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure Oracle Writer by using the code editor

In the following code, a node is configured to write data to an Oracle database.

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"stream",
            "parameter":{},
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"oracle",// The writer type.
            "parameter":{
                "postSql":[],// The SQL statement to execute after the sync node is run.
                "datasource":"",
                "session":[],// The settings of the session to the database.
                "column":[// The columns to which data is written.
                    "id",
                    "name"
                ],
                "encoding":"UTF-8",// The encoding format.
                "batchSize":1024,// The number of data records to write at a time.
                "table":"",// The name of the destination table.
                "preSql":[]// The SQL statement to execute before the sync node is run.
            },
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0"// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false,// Specifies whether to enable bandwidth throttling. A value o
f false indicates that the bandwidth is not throttled. A value of true indicates that the b
andwidth is throttled. The maximum transmission rate takes effect only if you set this para
meter to true.
            "concurrent":1 // The maximum number of concurrent threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

3.6.4.14. Configure OSS Writer

This topic describes the data types and parameters supported by OSS Writer and how to configure it by using the codeless UI and code editor.

OSS Writer allows you to write one or more CSV-like files to OSS.

> **Note** You must configure a connection before you configure OSS Writer.

OSS Writer can write files that store logical two-dimensional tables, such as CSV files that store text data, to OSS.

OSS Writer allows you to convert data obtained from a Data Integration reader to files and write the files to OSS. The OSS files store unstructured data only. OSS Writer supports the following features:

- Writes only files that store text data. The text data must be logical two-dimensional tables.
- Writes CSV-like files with custom delimiters.
- Uses concurrent threads to write files. Each thread writes a file.
- Supports file rotation. OSS Writer can write data to another file when the size of the current file exceeds a specific value. OSS Writer can also write data to another file when the number of rows in the current file exceeds a specific value.

OSS Writer does not support the following features:

- Uses concurrent threads.to write a single file.
- Distinguishes between data types. OSS does not distinguish between data types. Therefore, OSS Writer writes all data as strings to files in OSS.

The following table lists the data types supported by OSS Writer.

Category	OSS data type
Integer	LONG
Floating point	DOUBLE
String	STRING
Boolean	BOOLEAN
Date and time	DATE

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None

Parameter	Description	Required	Default value
object	The name prefix of the files to be written to OSS as objects. OSS simulates the directory effect by adding separators to object names. You can set the object parameter based on the following rules: • `"object": "datax"`: The names of the files start with datax, which is followed by a random string as the suffix. • `"object": "cdo/datax"`: The names of the files start with `/cdo/datax`, which is followed by a random string as the suffix. OSS uses backslashes (/) in objects to simulate the directory effect. If you do not want to add a random universally unique identifier (UUID) as the suffix, we recommend that you set the writeSingleObject parameter to `true`.	Yes	None
writeMode	The mode in which OSS Writer writes the files. Valid values: • *truncate*: deletes all existing objects with the specified object name prefix before writing files to OSS. For example, if you set the object parameter to `abc`, all objects whose names start with abc are deleted. • *append*: writes all files and ensures that the actual file names do not conflict with those of existing objects by suffixing the file names with random UUIDs. For example, if you set the object parameter to DI, the actual names of the files written to OSS are in the following format: DI_****_****_****. • *nonConflict*: returns an error message if an object with the specified object name exists. For example, if you set the `object` parameter to abc and the object named abc123 exists, an error message is returned.	Yes	None
fileFormat	The format in which the files are written to OSS. Valid values: csv and text. • If a file is written as a CSV file, the file strictly follows CSV specifications. If the data in the file contains the column delimiter, the column delimiter is escaped by using double quotation marks (" "). • If a file is written as a text file, the data in the file is separated with the column delimiter. If the data in the file contains the column delimiter, the column delimiter is not escaped.	No	*text*
fieldDelimiter	The column delimiter that is used in the files to be written to OSS.	No	,
encoding	The encoding format of the files to be written to OSS.	No	*utf-8*

Parameter	Description	Required	Default value
nullFormat	The string that represents null. No standard strings can represent null in text files. Therefore, Data Integration provides the nullFormat parameter to define which string represents a null pointer. For example, if you specify `nullFormat="null"`, Data Integration considers null as a null pointer.	No	None
header (advanced parameter, which cannot be set on the codeless UI)	The table header in the files to be written to OSS, for example, ['id','name','age'].	No	None
maxFileSize (advanced parameter, which cannot be set on the codeless UI)	The maximum size of a single file that can be written to OSS. Default value: 100000. Unit: MB. File rotation based on this maximum size is similar to log rotation of Log4j. When a file is uploaded to OSS in multiple parts, the minimum size of a part is 10 MB. This size is the minimum granularity for file rotation. That is, if you set the maxFileSize parameter to less than 10 MB, the minimum size of a file is still 10 MB. Each call of the InitiateMultipartUploadRequest operation supports writing up to 10,000 parts. If file rotation occurs, suffixes, such as _1, _2, and _3, are appended to the new file names that consist of file name prefixes and random UUIDs.	No	100,000MB
suffix (advanced parameter, which cannot be set on the codeless UI)	The file name extension of the files to be written to OSS. For example, if you set the suffix parameter to .csv, the final name of a file written to OSS is in the format fileName****.csv.	No	None

Configure OSS Writer by using the codeless UI

1. Configure the connections.

   Configure the connections to the source and destination data stores for the sync node.

Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and then select a connection name that you have configured in DataWorks.

Parameter	Description
Object Name Prefix	The object parameter in the preceding parameter description. Enter the path of the directory for storing the files. Do not include the name of the OSS bucket in the path.
File Type	The fileFormat parameter in the preceding parameter description. Valid values: *csv* and *text*.
Field Delimiter	The fieldDelimiter parameter in the preceding parameter description. The default delimiter is comma (,).
Encoding	The encoding parameter in the preceding parameter description. Default value: *UTF-8*.
Null String	The nullFormat parameter in the preceding parameter description. Enter a string that represents null. If the data in the source data store contains the string, the string is replaced with null.
Time Format	The format in which the data of the DATE type is serialized in an object, for example, `"dateFormat": "yyyy-MM-dd"`.
Solution to Duplicate Prefixes	The solution to take when a prefix conflict occurs. If an object with the specified name prefix exists, replace the object with the new object, insert the new object, or return an error message.

2. Configure field mapping. Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right.

GUI element	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.

3. Configure channel control policies.

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read data from and write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.

Parameter	Description
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure OSS Writer by using the code editor

In the following code, a node is configured to write files to OSS. For more information about the parameters, see the preceding parameter description.

```
{
    "type":"job",
    "version":"2.0",
    "steps":[
        {
            "stepType":"stream",
            "parameter":{},
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"oss",// The writer type.
            "parameter":{
                "nullFormat":"",// The string that represents null.
                "dateFormat":"",// The format in which the data of the DATE type is seriali
zed in an object.
                "datasource":"",// The connection name.
                "writeMode":"",// The write mode.
                "encoding":"",// The encoding format.
                "fieldDelimiter":","// The column delimiter.
                "fileFormat":"",// The format in which the files are written to OSS.
                "object":""// The name prefix of the files to be written to OSS as objects.
            },
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0"// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false,// Specifies whether to enable bandwidth throttling. A value o
f false indicates that the bandwidth is not throttled. A value of true indicates that the b
andwidth is throttled. The maximum transmission rate takes effect only if you set this para
meter to true.
            "concurrent":1 // The maximum number of concurrent threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

3.6.4.15. Configure PostgreSQL Writer

This topic describes the data types and parameters supported by PostgreSQL Writer and how to configure it by using the codeless UI and code editor.

PostgreSQL Writer allows you to write data to a PostgreSQL database. PostgreSQL Writer connects to a remote PostgreSQL database by using JDBC, and executes an SQL statement to write data to the PostgreSQL database.

> **Note** You must configure a connection before you configure PostgreSQL Writer.

- PostgreSQL Writer generates the SQL statement based on the table, column, and where parameters that you specified, and sends the generated SQL statement to the PostgreSQL database.
- If you specify the querySql parameter, PostgreSQL Writer directly sends the value of this parameter to the PostgreSQL database.

Data types

PostgreSQL Writer supports most PostgreSQL data types. Make sure that your data types are supported.

The following table lists the data types supported by PostgreSQL Writer.

Data Integration data type	PostgreSQL data type
LONG	BIGINT, BIGSERIAL, INTEGER, SMALLINT, and SERIAL
DOUBLE	DOUBLE, PRECISION, MONEY, NUMERIC, and REAL
STRING	VARCHAR, CHAR, TEXT, BIT, and INET
DATE	DATE, TIME, and TIMESTAMP
BOOLEAN	BOOL
BYTES	BYTEA

> **Note**
> - Data types that are not listed in the table are not supported.
> - You can convert the MONEY, INET, and BIT types by using syntax such as `a_inet::varchar`.

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None
table	The name of the destination table.	Yes	None

Parameter	Description	Required	Default value
writeMode	The write mode. Valid values: insert and copy. • *insert*: executes the `INSERT INTO` statement to write data to the PostgreSQL database. If a primary key conflict or unique index conflict occurs, data cannot be written to the conflicting rows and is regarded as dirty data. We recommend that you select the insert mode. • *copy*: copies data between tables and the standard input or output file. Data Integration supports the `COPY FROM` command, which allows you to copy data from files to tables. We recommend that you try this mode when performance issues occur.	No	*insert*
column	The columns in the destination table to which data is written. Separate the columns with commas (,), for example, `"column":["id","name","age"]`. To write data to all the columns in the destination table, set the value to an asterisk (*), for example, `"column":["*"]`.	Yes	None
preSql	The SQL statement to execute before the sync node is run. For example, you can clear outdated data before data synchronization. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor.	No	None
postSql	The SQL statement to execute after the sync node is run. For example, you can add a timestamp after data synchronization. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor.	No	None
batchSize	The number of data records to write at a time. Setting this parameter can greatly reduce the interactions between Data Integration and the PostgreSQL database over the network, and increase the throughput. However, an excessively large value may lead to the OOM error during the data synchronization process.	No	*1,024*

Parameter	Description	Required	Default value
pgType	The PostgreSQL configuration for converting data types. Valid values: bigint[], double[], text[], jsonb, and json. Example: `{ "job": { "content": [{ "reader": {...}, "writer": { "parameter": { "column": [// The columns in the destination table to which data is written. "bigint_arr", "double_arr", "text_arr", "jsonb_obj", "json_obj"], "pgType": { // The PostgreSQL configuration for converting data types. In each key-value pair, the key specifies the name of a field in the destination table, and the value specifies the data type of the field. "bigint_arr": "bigint[]", "double_arr": "double[]", "text_arr": "text[]", "jsonb_obj": "jsonb", "json_obj": "json" }] } } }] } }`	No	None

Configure PostgreSQL Writer by using the codeless UI

1. Configure the connections.

   Configure the connections to the source and destination data stores for the sync node.

Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and then select a connection name that you have configured in DataWorks.
Table	The table parameter in the preceding parameter description.

Parameter	Description
Statement Run Before Writing	The preSql parameter in the preceding parameter description. Enter an SQL statement to execute before the sync node is run.
Statement Run After Writing	The postSql parameter in the preceding parameter description. Enter an SQL statement to execute after the sync node is run.
Write Method	The writeMode parameter in the preceding parameter description. Valid values: *Insert* and *Copy*.

2. Configure field mapping. It is equivalent to setting the column parameter in the preceding parameter description. Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right.

Button	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	Click Auto Layout to sort the fields based on specified rules.

3. Configure channel control policies.

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read data from and write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure PostgreSQL Writer by using the code editor

In the following code, a node is configured to write data to a PostgreSQL database. For more information about the parameters, see the preceding parameter description.

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"stream",
            "parameter":{},
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"postgresql",// The writer type.
            "parameter":{
                "postSql":[],// The SQL statement to execute after the sync node is run.
                "datasource":"// The connection name.
                    "col1",
                    "col2"
                ],
                "table":"",// The name of the destination table.
                "preSql":[]// The SQL statement to execute before the sync node is run.
            },
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0"// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false,// Specifies whether to enable bandwidth throttling. A value o
f false indicates that the bandwidth is not throttled. A value of true indicates that the b
andwidth is throttled. The maximum transmission rate takes effect only if you set this para
meter to true.
            "concurrent":1 // The maximum number of concurrent threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

3.6.4.16. Configure Redis Writer

Redis Writer is a writer that is developed based on the Data Integration framework. It can be used to import data from data stores such as data warehouses to Redis databases.

Redis is a network-enabled key-value storage system that is either in-memory or permanent. It supports logs and delivers high performance. It can be used as a database, cache, and message broker. Redis supports diverse data types for values, including STRING, LIST, SET, ZSET (sorted set), and HASH.

Redis Writer interacts with a Redis server by using Jedis. As a preferred Java client development kit provided by Redis, Jedis supports almost all the features of Redis.

Note

- You must configure a connection before you configure Redis Writer.
- If you write values of the LIST type to Redis by using Redis Writer, the result of rerunning a sync node is not idempotent. If the data type of the values is LIST, you must manually clear the corresponding data on Redis when you rerun a sync node.

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None
keyIndexes	The columns used as the key. The index of the first column is 0. For example, if you want to set the first and second columns of the source data as the key, set the keyIndexes parameter to [0,1]. **Note** After you specify the keyIndexes parameter, Redis Writer specifies the remaining columns as the value. If you do not want to synchronize all the columns, filter columns when you configure the reader.	Yes	None
keyFieldDelimiter	The delimiter used to separate keys when data is written to Redis. Example: key=key1\u0001id. If multiple keys need to be concatenated, this parameter is required. If only one key exists, this parameter is not required.	No	*\u0001*
batchSize	The number of data records to write at a time. Setting this parameter can greatly reduce the interactions between Data Integration and the Redis database over the network, and increase the throughput. However, an excessively large value may lead to the OOM error during the data synchronization process.	No	*1,000*

Parameter	Description	Required	Default value
expireTime	The expiration time of the values to be cached in Redis. Unit: seconds. The data is valid permanently if you do not specify this parameter. • *seconds*: the relative time in seconds starting from the current time point. It specifies the time range during which data is valid. • *unixtime*: the UNIX timestamp, indicating that data is invalid at a specific time point in the future. The UNIX timestamp represents the number of seconds that have elapsed since 00:00:00 on January 1, 1970. **Note** If the specified expiration time is larger than 30 days, the server identifies the time as the UNIX timestamp.	No	*0*, indicating that the values never expire
timeout	The timeout period to connect to Redis when data is written to Redis. Unit: milliseconds.	No	*30,000*
dateFormat	The format in which the data of the DATE type is written to Redis. Set the value to yyyy-MM-dd HH:mm:ss.	No	None
writeMode	The write mode. Redis supports diverse data types for values, including STRING, LIST, SET, ZSET (sorted set), and HASH. Redis Writer allows you to write values of the preceding types to Redis. The value of the writeMode parameter varies based on the specified data type of the values. **Note** When you configure Redis Writer, you can choose only one of the five data types described in the following table. If you do not specify a data type, the data type is STRING by default.	No	*string*

The following table lists the data types supported by Redis Writer.

Type	Parameter	Description	Required
String `"writeMode":{ "type": "string", "mode": "set", "valueFieldDelimiter": "\u0001" }`	type	The data type of the values is STRING.	Yes
	mode	The mode in which data of the STRING type is written to Redis.	Yes. Valid value: set (overwrites the existing data).
	valueFieldDelimiter	This parameter is required if two or more columns are specified as the values. This parameter is not required if only one column is specified as the values. The delimiter used to separate values if the data is of the STRING type. Example: value1\u0001value2\u0001value3.	No. Default value: \u0001.
LIST `"writeMode":{ "type": "list", "mode": "lpush|rpush", "valueFieldDelimiter": "\u0001" }`	type	The data type of the values is LIST.	Yes
	mode	The mode in which data of the LIST type is written to Redis.	Yes. Valid values: lpush (stores the data at the leftmost of the list) and rpush (stores the data at the rightmost of the list).
	valueFieldDelimiter	The delimiter used to separate values if the data is of the STRING type. Example: value1\u0001value2\u0001value3.	No. Default value: *\u0001*.
SET	type	The data type of the values is SET.	Yes
	mode	The mode in which data of the SET type is written to Redis.	Yes. Valid value: sadd (stores the data to a set, or overwrites the existing data).

Type	Parameter	Description	Required
"writeMode":{ "type": "set", "mode": "sadd", "valueFieldDelimiter": "\u0001" }	valueFieldDelimiter	The delimiter used to separate values if the data is of the STRING type. Example: value1\u0001value2\u0001value3.	No. Default value: *\u0001*.
ZSET (sorted set) "writeMode":{ "type": "zset", "mode": "zadd" }	type	The data type of the values is ZSET. **Note** If the data type of the values is ZSET, each data record must follow the following standard: Except for the key, a data record can contain only one score and one value. The score must be placed before the value. In this way, Redis Writer can identify which column is the score and which column is the value.	Yes
	mode	The mode in which data of the ZSET type is written to Redis.	Yes. Valid value: zadd (stores data to a sorted set, or overwrites the existing data).

Type	Parameter	Description	Required
HASH `"writeMode":{ "type": "hash", "mode": "hset" }`	type	The data type of the values is HASH. **Note** If the data type of the values is HASH, each data record must follow the following standards: Except for the key, a data record can contain only one attribute and one value. The attribute must be placed before the value. In this way, Redis Writer can identify which column is the attribute and which column is the value.	Yes
	mode	The mode in which data of the HASH type is written to Redis.	Yes. Valid value: hmset (stores data to a hash sorted set, or overwrites the existing data). If you do not specify a data type, the data type is STRING by default.

Configure Redis Writer by using the codeless UI

The codeless UI is not supported for Redis Reader.

Configure Redis Writer by using the code editor

In the following code, a node is configured to write data to Redis. For more information about parameters, see the preceding parameter description.

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"stream",
            "parameter":{},
            "name":"Reader",
            "category":"reader"
```

```
        },
        {
            "stepType":"redis",// The writer type.
            "parameter":{
                "expireTime":{// The expiration time of the values to be cached in Redis.
                    "seconds":"1000"
                },
                "keyFieldDelimiter":"u0001",// The delimiter used to separate keys when dat
a is written to Redis.
                "dateFormat":"yyyy-MM-dd HH:mm:ss",// The format in which the data of the D
ATE type is written to Redis.
                "datasource":"",// The connection name.
                "writeMode":{// The write mode.
                    "mode":"",// The write mode used to write data of a specified data type
.
                    "valueFieldDelimiter":"",// The delimiter used to separate values.
                    "type":""// The data type of the values.
                },
                "keyIndexes":[// The columns used as the key.
                    0,
                    1
                ],
                "batchSize":"1000"// The number of data records to write at a time.
            },
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0"// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false,// Specifies whether to enable bandwidth throttling. A value o
f false indicates that the bandwidth is not throttled. A value of true indicates that the b
andwidth is throttled. The maximum transmission rate takes effect only if you set this para
meter to true.
            "concurrent":1 // The maximum number of concurrent threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}Writer"
            }
        ]
    }
}
```

3.6.4.17. Configure SQL Server Writer

This topic describes the data types and parameters supported by SQL Server Writer and how to configure it by using the codeless UI and code editor.

SQL Server Writer allows you to write data to tables stored in primary SQL Server databases. SQL Server Writer connects to a remote SQL Server database by using JDBC, and executes an `INSERT INTO` statement to write data to the SQL Server database. Internally, data is submitted to the database in batches.

SQL Server Writer is designed for ETL developers to import data from data warehouses to SQL Server databases. SQL Server Writer can also be used as a data migration tool by users such as DBAs.

SQL Server Writer obtains data from a Data Integration reader, and generates the `INSERT INTO` statement. If a primary key conflict or unique index conflict occurs, data cannot be written to the conflicting rows. To improve performance, SQL Server Writer makes batch updates with the `PreparedStatement` method and sets `rewriteBatchedStatements` to true. In this way, SQL Server Writer buffers data, and submits a write request when the amount of data in the buffer reaches a specific threshold.

> Note

- Data can be written only to tables stored in the primary SQL Server database.
- A sync node that uses SQL Server Writer must have at least the permission to execute the INSERT INTO statement. Whether other permissions are required depends on the SQL statements specified in the preSql and postSql parameters when you configure the node.

Data types

SQL Server Writer supports most SQL Server data types. Make sure that your data types are supported.

The following table lists the data types supported by SQL Server Writer.

Category	SQL Server data type
Integer	BIGINT, INT, SMALLINT, and TINYINT
Floating point	FLOAT, DECIMAL, REAL, and NUMERIC
String	CHAR, NCHAR, NTEXT, NVARCHAR, TEXT, VARCHAR, NVARCHAR (MAX), and VARCHAR (MAX)
Date and time	DATE, TIME, and DATETIME
Boolean	BIT
Binary	BINARY, VARBINARY, VARBINARY (MAX), and TIMESTAMP

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None
table	The name of the destination table.	Yes	None
column	The columns in the destination table to which data is written. Separate the columns with commas (,). Example: `"column":["id","name","age"]`. To write data to all the columns in the destination table, set the value to an asterisk (*), for example, `"column":["*"]`.	Yes	None
preSql	The SQL statement to execute before the sync node is run. For example, you can clear outdated data before data synchronization. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor.	No	None
postSql	The SQL statement to execute after the sync node is run. For example, you can add a timestamp after data synchronization. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor.	No	None
writeMode	The write mode. Valid value: *insert*. When a data record violates the primary key constraint or unique index constraint, Data Integration considers it dirty and retains the original data.	No	*insert*
batchSize	The number of data records to write at a time. Setting this parameter can greatly reduce the interactions between Data Integration and the SQL Server database over the network, and increase the throughput. However, an excessively large value may lead to the OOM error during the data synchronization process.	No	*1,024*

Configure SQL Serve Writer by using the codeless UI

1. Configure the connections.

   Configure the connections to the source and destination data stores for the sync node.

Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and then select a connection name that you have configured in DataWorks.
Table	The table parameter in the preceding parameter description.
Statement Run Before Writing	The preSql parameter in the preceding parameter description. Enter an SQL statement to execute before the sync node is run.
Statement Run After Writing	The postSql parameter in the preceding parameter description. Enter an SQL statement to execute after the sync node is run.

Parameter	Description
Solution to Primary Key Violation	The writeMode parameter in the preceding parameter description. Select the required write mode.

2. Configure field mapping. It is equivalent to setting the column parameter in the preceding parameter description.

   Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right. You can click **Add** to add a field, or move the pointer over a field and click the **Delete** icon to delete the field.

GUI element	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	Click Auto Layout to sort the fields based on specified rules.
Change Fields	Click the Change Fields icon. In the Change Fields dialog box, you can manually edit fields in the source table. Each field occupies a row. The first and the last blank rows are included, whereas other blank rows are ignored.
Add	◦ Click Add to add a field. You can enter constants. Each constant must be enclosed in single quotation marks (' '), for example, 'abc' and '123'. ◦ You can use scheduling parameters such as ${bizdate}. ◦ You can enter functions supported by relational databases, for example, now() and count(1). ◦ If the value you entered cannot be parsed, the type is displayed as Unidentified.

3. Configure channel control policies.

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read data from and write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Parameter	Description
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure SQL Server Writer by using the code editor

In the following code, a node is configured to write data to an SQL Server database. For more information about the parameters, see the preceding parameter description.

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"stream",
            "parameter":{},
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"sqlserver",// The writer type.
            "parameter":{
                "postSql":[],// The SQL statement to execute after the sync node is run.
                "datasource":"",// The connection name.
                "column":[// The columns to which data is written.
                    "id",
                    "name"
                ],
                "table":"",// The name of the destination table.
                "preSql":[]// The SQL statement to execute before the sync node is run.
            },
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0"// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false,// Specifies whether to enable bandwidth throttling. A value o
f false indicates that the bandwidth is not throttled. A value of true indicates that the b
andwidth is throttled. The maximum transmission rate takes effect only if you set this para
meter to true.
            "concurrent":1 // The maximum number of concurrent threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

3.6.4.18. Configure Elasticsearch Writer

This topic describes the data types and parameters supported by Elasticsearch Writer and how to configure it by using the code editor.

Elasticsearch is an open-source product that complies with the Apache open standards. It is the mainstream search engine for enterprise data. Elasticsearch is a Lucene-based data search and analysis tool that provides distributed services. The mappings between Elasticsearch core concepts and database core concepts are as follows:

```
Relational database (instance) -> database -> table -> row -> column
Elasticsearch -> index -> type -> document -> field
```

Elasticsearch can contain multiple indexes (databases). Each index can contain multiple types (tables). Each type can contain multiple documents (rows). Each document can contain multiple fields (columns). Elasticsearch Writer uses the RESTful API of Elasticsearch to write multiple data records retrieved by a reader to Elasticsearch at a time.

Parameters

Parameter	Description	Required	Default value
endpoint	The endpoint for accessing Elasticsearch, in the format of `http://xxxx.com:9999`.	No	None
accessId	The AccessKey ID for accessing Elasticsearch, which is used for authorization when a connection with Elasticsearch is established. **Note** The accessId and accessKey parameters are required. If you do not set the parameters, an error is returned. If you use on-premises Elasticsearch for which basic authentication is not configured, the AccessKey ID and AccessKey secret are not required. In this case, you can set the accessId and accessKey parameters to random values.	No	None
accessKey	The AccessKey secret for accessing Elasticsearch.	No	None
index	The index name in Elasticsearch.	No	None
indexType	The type name in the index of Elasticsearch.	No	*Elasticsearch*
cleanup	Specifies whether to clear existing data in the index. The method used to clear the data is to delete and rebuild the corresponding index. The default value false indicates that the existing data in the index is retained.	No	*false*
batchSize	The number of data records to write at a time.	No	*1000*

Parameter	Description	Required	Default value
trySize	The number of retries after a failure.	No	*30*
timeout	The connection timeout of the client. Unit: milliseconds.	No	*600000*
discovery	Specifies whether to enable Node Discovery. When Node Discovery is enabled, the server list in the client is polled and regularly updated.	No	*false*
compression	Specifies whether to enable compression for an HTTP request.	No	*true*
multiThread	Specifies whether to use multiple threads for an HTTP request.	No	*true*
ignoreWriteError	Specifies whether to ignore write errors and proceed with writing without retries.	No	*false*
ignoreParseError	Specifies whether to ignore format parsing errors and proceed with writing.	No	*true*
alias	The alias of the index. The alias feature of Elasticsearch is similar to the view feature of a traditional database. For example, if you create an alias named my_index_alias for the index my_index, the operations on my_index_alias also take effect on my_index. Configuring alias means that after the data import is completed, an alias is created for the specified index.	No	None
aliasMode	The mode in which an alias is added after the data is imported. Valid values: *append* and *exclusive*.	No	append
settings	The delimiter (-,-) for splitting the source data if you are inserting an array to Elasticsearch. Example: The source column stores data `a-,-b-,-c-,-d` of the String type. Elasticsearch Writer uses the delimiter (-,-) to split the source data and obtains the array `["a", "b", "c", "d"]`. Then, Elasticsearch Writer writes the array to the corresponding field in Elasticsearch.	No	*-,-*
	The fields of the document. The parameters for each field include basic parameters such as name and type and advanced parameters such as analyzer, format, and array. The field types supported by Elasticsearch are as follows:		

Parameter	Description	Required	Default value
column	- id // The id type corresponds to the _id type in Elasticsearch, and can be considered as the unique primary key. Data with the same ID will be overwritten and not indexed. - string - text - keyword - long - integer - short - byte - double - float - date - boolean - binary - integer_range - float_range - long_range - double_range - date_range - geo_point - geo_shape - ip - token_count - array - object - nested • When the field type is Text, you can specify the analyzer, norms, and index_options parameters. Example: { "name": "col_text", "type": "text", "analyzer": "ik_max_word" } • When the field type is date, you can specify the format and timezone parameters, indicating the date serialization format and the time zone, respectively. Example: { "name": "col_date", "type": "date", "format": "yyyy-MM-dd HH:mm:ss", "timezone": "UTC" } • When the field type is ge_shape, you can specify the tree (geohash or quadtree) and precision	Yes	None

Parameter	Description	Required	Default value
	parameters. Example: { "name": "col_geo_shape", "type": "geo_shape", "tree": "quadtree", "precision": "10m" } If you specify the array parameter for a field and set the array parameter to *true*, the field is an array column. Elasticsearch Writer uses the delimiter specified by the splitter to split the source data, converts the data to an array of strings, and writes the array to the destination. Only one delimiter is supported for one node. Example: { "name": "col_integer_array", "type": "integer", "array": true }		
dynamic	Specifies whether to use the mapping configuration of Elasticsearch. A value of *true* indicates that the mapping configuration of Elasticsearch, instead of the mapping configuration of Data Integration, is used.	No	*false*

Parameter	Description	Required	Default value
actionType	The type of the action for writing data to Elasticsearch. Currently, Data Integration supports only the following action types: *index* and *update*. Default value: *index*. • *index*: Data Integration uses Index.Builder of the Elasticsearch SDK to construct a request for writing multiple data records at a time. In *index* mode, Elasticsearch first checks whether an ID is specified for the document to be inserted. ◦ If the ID is not specified, Elasticsearch generates a unique ID by default. In this case, the document is directly inserted to Elasticsearch. ◦ If the ID is specified, Elasticsearch replaces the existing document with the document to be inserted. Note In this case, you cannot modify specific fields in the document. • *update*: Data Integration uses Update.Builder of the Elasticsearch SDK to construct a request for writing multiple data records at a time. In *update* mode, Elasticsearch calls the get method of InternalEngine to obtain the information of the original document for each update. In this way, you can modify specific fields. In update mode, you must obtain the information of the original document for each update, which greatly affects the performance. However, you can modify specific fields in this mode. If the original document does not exist, the new document is directly inserted.	No	*index*

Configure Elasticsearch Writer by using the code editor

In the following code, a node is configured to write data to Elasticsearch. For more information about the parameters, see the preceding parameter description.

```
{
    "order": {
        "hops": [
            {
                "from": "Reader",
                "to": "Writer"
            }
        ]
    },
    "setting": {
        "errorLimit": {
            "record": "0"
        },
        "speed": {
```

```
            "concurrent": 1,
            "throttle": false
        }
    },
    "steps": [
        {
            "category": "reader",
            "name": "Reader",
            "parameter": {
            },
            "stepType": "stream"
        },
        {
            "category": "writer",
            "name": "Writer",
            "parameter": {
                "endpoint": "http://xxxx.com:9999",
                "accessId": "xxxx",
                "accessKey": "yyyy",
                "index": "test-1",
                "type": "default",
                "cleanup": true,
                "settings": {
                    "index": {
                        "number_of_shards": 1,
                        "number_of_replicas": 0
                    }
                },
                "discovery": false,
                "batchSize": 1000,
                "splitter": ",",
                "column": [
                    {
                        "name": "pk",
                        "type": "id"
                    },
                    {
                        "name": "col_ip",
                        "type": "ip"
                    },
                    {
                        "name": "col_double",
                        "type": "double"
                    },
                    {
                        "name": "col_long",
                        "type": "long"
                    },
                    {
                        "name": "col_integer",
                        "type": "integer"
                    },
                    {
                        "name": "col_keyword",
```

```
                        "type": "keyword"
                    },
                    {
                        "name": "col_text",
                        "type": "text",
                        "analyzer": "ik_max_word"
                    },
                    {
                        "name": "col_geo_point",
                        "type": "geo_point"
                    },
                    {
                        "name": "col_date",
                        "type": "date",
                        "format": "yyyy-MM-dd HH:mm:ss"
                    },
                    {
                        "name": "col_nested1",
                        "type": "nested"
                    },
                    {
                        "name": "col_nested2",
                        "type": "nested"
                    },
                    {
                        "name": "col_object1",
                        "type": "object"
                    },
                    {
                        "name": "col_object2",
                        "type": "object"
                    },
                    {
                        "name": "col_integer_array",
                        "type": "integer",
                        "array": true
                    },
                    {
                        "name": "col_geo_shape",
                        "type": "geo_shape",
                        "tree": "quadtree",
                        "precision": "10m"
                    }
                ]
            },
            "stepType": "elasticsearch"
        }
    ],
    "type": "job",
    "version": "2.0"
}
```

> **Note** Currently, Elasticsearch that is deployed in a Virtual Private Cloud (VPC) supports only custom resource groups. A sync node that is run on the default resource group may fail to connect to Elasticsearch.

3.6.4.19. Configure LogHub Writer

This topic describes the data types and parameters supported by LogHub Writer and how to configure it by using the code editor.

LogHub Writer allows you to transfer data from a Data Integration reader to LogHub through Log Service Java SDK.

> **Note** LogHub does not guarantee idempotence. Rerunning a node after the node fails may result in redundant data.

LogHub Writer obtains data from a Data Integration reader and converts the data types supported by Data Integration to String. When the number of the data records reaches the value specified for the batchSize parameter, LogHub Writer sends the data records to LogHub at a time through Log Service Java SDK. LogHub Writer sends 1,024 data records at a time by default. The batchSize parameter can be set to 4096 at most.

Data types

The following table lists the data types supported by LogHub Writer.

Data Integration data type	LogHub data type
LONG	STRING
DOUBLE	STRING
STRING	STRING
DATE	STRING
BOOLEAN	STRING
BYTES	STRING

Parameters

Parameter	Description	Required	Default value
endpoint	The endpoint for accessing Log Service.	Yes	None
accessKeyId	The AccessKey ID for accessing Log Service.	Yes	None
accessKeySecret	The AccessKey secret for accessing Log Service.	Yes	None

Parameter	Description	Required	Default value
project	The name of the destination Log Service project.	Yes	None
logstore	The name of the destination Logstore.	Yes	None
topic	The name of the destination topic.	No	Empty string
batchSize	The number of data records to write at a time.	No	*1024*
column	The columns in each data record.	Yes	None

Configure LogHub Writer by using the codeless UI

Currently, the codeless user interface (UI) is not supported for LogHub Writer.

Configure LogHub Writer by using the code editor

In the following code, a node is configured to write data to LogHub. For more information about the parameters, see the preceding parameter description.

```
{
    "type": "job",
    "version": "2.0",// The version number.
    "steps": [
        { //
            "stepType": "stream",
            "parameter": {},
            "name": "Reader",
            "category": "reader"
        },
        {
            "stepType": "loghub",// The writer type.
            "parameter": {
                "datasource": "",// The connection name.
                "column": [// The columns in each data record.
                    "col0",
                    "col1",
                    "col2",
                    "col3",
                    "col4",
                    "col5"
                ],
                "topic": "",// The name of the destination topic.
                "batchSize": "1024",// The number of data records to write at a time.
                "logstore": ""// The name of the destination Logstore.
            },
            "name": "Writer",
            "category": "writer"
        }
    ],
    "setting": {
        "errorLimit": {
            "record": ""// The maximum number of dirty data records allowed.
        },
        "speed": {
            "concurrent": 3,// The maximum number of concurrent threads.
            "throttle": false,// Specifies whether to enable bandwidth throttling. A value
of false indicates that the bandwidth is not throttled. A value of true indicates that the
bandwidth is throttled. The maximum transmission rate takes effect only if you set this par
ameter to true.
        }
    },
    "order": {
        "hops": [
            {
                "from": "Reader",
                "to": "Writer"
            }
        ]
    }
}
```

3.6.4.20. Configure Open Search Writer

This topic describes the data types and parameters supported by Open Search Writer and how to configure it by using the code editor.

How it works

Open Search Writer allows you to insert data to or update data in Open Search. Open Search Writer is designed for developers to import data to Open Search so that the data can be searched.

Specifically, Open Search Writer uses the search API provided by Open Search to import data.

> Note

- Open Search V3 uses internal dependent databases, with POM of `com.aliyun.opensearch aliyun-sdk-opensearch 2.1.3`.
- To use Open Search Writer, you must install JDK 1.6-32 or later. You can run the `java-version` command to view the JDK version.

Features

The columns in Open Search are unordered. Open Search Writer writes data strictly in accordance with the order of the specified columns. If the number of specified columns is less than that in Open Search, redundant columns in Open Search are set to the default value or null.

Assume that an Open Search table contains columns a, b, and c, and you only need to write data to columns b and c. You can set the column parameter to ["c","b"]. In this case, Open Search Writer imports the first and second columns of the source data obtained from a reader to columns c and b in the Open Search table respectively. Column a in the Open Search table is set to the default value or null.

Additional instructions:

- Handling of column configuration errors

  To avoid losing the data of redundant columns and ensure high data reliability, Open Search Writer returns an error message if the number of columns to be written is more than that in the destination Open Search table. For example, if an Open Search table contains columns a, b, and c, Open Search Writer returns an error if more than three columns are to be written to the table.

- Table configuration

  Open Search Writer can write data to only one table at a time.

- Node rerunning

  After a node is rerun, data is automatically overwritten based on IDs. Therefore, the data written to Open Search must contain one ID column. An ID is a unique identifier of a row in Open Search. The existing data with the same ID as the new data will be overwritten.

- Node rerunning

  After a node is rerun, data is automatically overwritten based on IDs.

Data types

Open Search Writer supports most Open Search data types. Make sure that your data types are supported.

The following table lists the data types supported by Open Search Writer.

Category	Open Search data type
Integer	INT
Floating point	DOUBLE and FLOAT
String	TEXT, LITERAL, and SHORT_TEXT
Date and time	INT
Boolean	LITERAL

Parameters

Parameter	Description	Required	Default value
accessId	The AccessKey ID for connecting to the Open Search database.	Yes	None
accessKey	The AccessKey secret for connecting to the Open Search database.	Yes	None
host	The endpoint for connecting to Open Search. You can view the endpoint in the Apsara Stack console.	Yes	None
indexName	The name of the Open Search project.	Yes	None
table	The name of the table to which data is written. You can specify only one table name because Data Integration does not support importing data to multiple tables at a time.	Yes	None
column	The columns in the destination table to which data is written. To write data to all the columns in the destination table, set the value to an asterisk (*), for example, `"column":["*"]`. Separate the columns with a comma (,) if data is written to some of the columns in the destination table. Example: `"column":["id","name"]`. Open Search Writer supports filtering columns and changing the order of columns. Assume that an Open Search table has three columns: a, b, and c. If you want to write data only to columns c and b, you can set the column parameter in the format `"column":["c","b"]`. During data synchronization, column a is automatically set to null.	Yes	None

Parameter	Description	Required	Default value
batchSize	The number of data records to write at a time. Data is written to Open Search in batches. The advantage of Open Search is data query. The transactions per second (TPS) of Open Search is generally not high. Set this parameter based on the resources available for the account that is used to connect to Open Search. Generally, the size of a data record must be less than 1 MB, and the size of the data records to write at a time must be less than 2 MB.	Required only for writing data to a partitioned table	*300*
writeMode	The write mode. To ensure the idempotence of write operations, set the writeMode parameter to add/update when you configure Open Search Writer. • add: deletes the existing data record and inserts the new data record to Open Search, which is an atomic operation. • update: updates the existing data record based on the new data record, which is an atomic operation. **Note** Writing data to Open Search in batches is not an atomic operation. Part of the data may fail to be written. Exercise caution when you set the writeMode parameter. Open Search V3 does not support the update mode.	Yes	None
ignoreWriteError	Specifies whether to ignore failed write operations. Example: `"ignoreWriteError":true`. If data is written to Open Search in batches, this parameter specifies whether to ignore failed write operations in the current batch. If you set the parameter to true, Open Search Writer continues to perform other write operations. If you set the parameter to false, the sync node ends and an error message is returned. We recommend that you use the default value.	No	*false*
version	The version of Open Search, for example, `"version":"v3"`. We recommend that you use Open Search V3 because the push operation faces many constraints in Open Search V2.	No	*v2*

Configure Open Search Writer by using the code editor

In the following code, a node is configured to write data to Open Search.

```
{
    "type": "job",
    "version": "1.0",
    "configuration": {
        "reader": {},
        "writer": {
            "plugin": "opensearch",
            "parameter": {
                "accessId": "*********",
                "accessKey": "********",
                "host": "http://yyyy.aliyuncs.com",
                "indexName": "datax_xxx",
                "table": "datax_yyy",
                "column": [
                "appkey",
                "id",
                "title",
                "gmt_create",
                "pic_default"
                ],
                "batchSize": 500,
                "writeMode": add,
                "version":"v2",
                "ignoreWriteError": false
            }
        }
    }
}
```

3.6.4.21. Configure Tablestore Writer

This topic describes the data types and parameters supported by Tablestore Writer and how to configure it by using the code editor.

Tablestore is a NoSQL database service that is built on the Apsara distributed operating system. The service allows you to store and access large volumes of structured data in real time. Tablestore organizes data into instances and tables. It uses data sharding and load balancing technologies to seamlessly expand the data scale.

Tablestore Writer connects to the Tablestore server by using Tablestore SDK for Java and writes data to the server by using the SDK. Tablestore Writer greatly optimizes the write process, including retry after write timeouts, retry after exceptions, and batch submission.

Tablestore Writer writes data to Tablestore in one of the following modes:

- PutRow: the PutRow API operation for Tablestore, which is used to insert data to a specified row. If this row does not exist, a new row is added. Otherwise, the original row is overwritten.
- UpdateRow: the UpdateRow API operation for Tablestore, which is used to update the data of a specified row. If this row does not exist, a new row is added. Otherwise, the values of the specified columns are added, modified, or removed as requested.

Tablestore Writer supports all Tablestore data types. The following table lists the data types.

Category	Tablestore data type
Integer	INTEGER
Floating point	DOUBLE
String	STRING
Boolean	BOOLEAN
Binary	BINARY

Note To write INTEGER-type data, set the data type to INT in the code editor. Then, DataWorks converts the INT type into the INTEGER type. If you directly set the data type to INTEGER, an error is reported in the log, and the node fails.

Parameters

Parameter	Description	Required	Default value
datasource	The name of the data source. It must be the same as the name of the added data source. You can add data sources by using the code editor.	Yes	None
endPoint	The endpoint of the Tablestore server.	Yes	None
accessId	The AccessKey ID of the account that is used to access Tablestore.	Yes	None
accessKey	The AccessKey secret of the account that is used to access Tablestore.	Yes	None
instanceName	The name of the Tablestore instance. The instance is an entity for you to use and manage Tablestore. After you activate the Tablestore service, you must create an instance in the Tablestore console before you can create and manage tables. Instances are the basic units that you can use to manage Tablestore resources. Access control and resource measurement for applications are implemented at the instance level.	Yes	None
table	The name of the destination table. You can specify only one table as the destination table. Multi-table synchronization is not required for Tablestore.	Yes	None

Parameter	Description	Required	Default value
primaryKey	The primary key of the destination table in Tablestore. The primary keys are described in a JSON array. Tablestore is a NoSQL database service. You must specify the primary key of the destination table for Tablestore Writer to write data. **Note** The primary keys in Tablestore must be of the STRING or INT type. Therefore, you must set the data type of a primary key to STRING or INT in the code editor. Data Integration supports data type conversion. Tablestore Writer can convert data that is not of the STRING or INT type to the STRING or INT type. The following code provides a configuration example: `"primaryKey" : [ {"name":"pk1", "type":"string"}, {"name":"pk2", "type":"int"} ],`	Yes	None
column	The columns that you want to synchronize to the destination table. The columns are described in a JSON array. Specify this parameter in the following format: `{"name":"col2", "type":"INT"},` The name parameter specifies the name of the column to which data is written. The type parameter specifies the data type of the column. Data types supported by Tablestore include STRING, INT, DOUBLE, BOOLEAN, and BINARY.	Yes	None
writeMode	The write mode. Constants, functions, or custom statements are not supported during the write process. The following three modes are supported: • PutRow: the PutRow API operation for Tablestore, which is used to insert data to a specified row. If this row does not exist, a new row is added. Otherwise, the original row is overwritten. • UpdateRow: the UpdateRow API operation for Tablestore, which is used to update the data of a specified row. If this row does not exist, a new row is added. Otherwise, the values of the specified columns are added, modified, or removed as requested. • DeleteRow: deletes a row.	Yes	None

Configure Tablestore Writer by using the codeless UI

This method is not supported.

Configure Tablestore Writer by using the code editor

In the following code, a node is configured to write data to Tablestore:

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"stream",
            "parameter":{},
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"ots",// The writer type.
            "parameter":{
                "datasource":"",// The data source.
                "column":[// The columns to which data is written.
                    {
                        "name":"columnName1",// The name of the column.
                        "type":"INT"// The data type of the column.
                    },
                    {
                        "name":"columnName2",
                        "type":"STRING"
                    },
                    {
                        "name":"columnName3",
                        "type":"DOUBLE"
                    },
                    {
                        "name":"columnName4",
                        "type":"BOOLEAN"
                    },
                    {
                        "name":"columnName5",
                        "type":"BINARY"
                    }
                ],
                "writeMode":"",// The write mode.
                "table":"",// The name of the destination table.
                "primaryKey":[// The primary key of the destination table in Tablestore.
                    {
                        "name":"pk1",
                        "type":"STRING"
                    },
                    {
                        "name":"pk2",
                        "type":"INT"
                    }
                ]
            },
            "name":"Writer",
            "category":"writer"
```

```
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0"// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false,// Specifies whether to enable bandwidth throttling. The value
false indicates that bandwidth throttling is disabled, and the value true indicates that ba
ndwidth throttling is enabled. The concurrent parameter takes effect only when the throttle
parameter is set to true.
            "concurrent":1,// The maximum number of concurrent threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

3.6.4.22. Configure RDBMS Writer

This topic describes the data types and parameters supported by RDBMS Writer and how to configure it by using the code editor.

RDBMS Writer allows you to write data to tables stored in primary relational database management system (RDBMS) databases. Specifically, RDBMS Writer obtains data from a Data Integration reader, connects to a remote RDBMS database through Java Database Connectivity (JDBC), and then runs an `INSERT INTO` statement to write data to the RDBMS database. RDBMS Writer is a common writer for relational databases. To enable RDBMS Writer to support a new relational database, register the driver for the relational database.

RDBMS Writer is designed for extract-transform-load (ETL) developers to import data from data warehouses to RDBMS databases. RDBMS Writer can also be used as a data migration tool by users such as database administrators (DBAs).

Data types

RDBMS Writer supports most data types in relational databases, such as numbers and characters. Make sure that your data types are supported.

Parameters

Parameter	Description	Required	Default value

Parameter	Description	Required	Default value
jdbcUrl	The JDBC URL for connecting to the database. The format must be in accordance with official specifications. You can also specify the information of the attachment facility. The format varies with the database type. Data Integration selects an appropriate driver for data reading based on the format. • Format for DM databases: `jdbc:dm://ip:port/database` • Format for Db2 databases: `jdbc:db2://ip:port/database` • Format for PPAS databases: `jdbc:edb://ip:port/database`	Yes	None
username	The username for connecting to the database.	Yes	None
password	The password for connecting to the database.	Yes	None
table	The name of the destination table.	Yes	None
column	The columns in the destination table to which data is written. Separate the columns with a comma (,). **Note** We recommend that you do not use the default setting.	Yes	None
preSql	The SQL statement to run before the sync node is run. For example, you can clear outdated data before data synchronization. Currently, you can run only one SQL statement. **Note** If you specify multiple SQL statements in the code editor, the system does not guarantee that they are run in the same transaction.	No	None
postSql	The SQL statement to run after the sync node is run. For example, you can add a timestamp after data synchronization. Currently, you can run only one SQL statement. **Note** If you specify multiple SQL statements in the code editor, the system does not guarantee that they are run in the same transaction.	No	None

Parameter	Description	Required	Default value
batchSize	The number of data records to write at a time. Setting this parameter can greatly reduce the interactions between Data Integration and the RDBMS database over the network, and increase the throughput. However, an excessively large value may lead to the out of memory (OOM) error during the data synchronization process.	No	1024

Configure RDBMS Writer by using the code editor

In the following code, a node is configured to write data to an RDBMS database.

```
{
    "job": {
        "setting": {
            "speed": {
                "channel": 1
            }
        },
        "content": [
            {
                "reader": {
                    "name": "streamreader",
                    "parameter": {
                        "column": [
                            {
                                "value": "DataX",
                                "type": "string"
                            },
                            {
                                "value": 19880808,
                                "type": "long"
                            },
                            {
                                "value": "1988-08-08 08:08:08",
                                "type": "date"
                            },
                            {
                                "value": true,
                                "type": "bool"
                            },
                            {
                                "value": "test",
                                "type": "bytes"
                            }
                        ],
                        "sliceRecordCount": 1000
                    }
                },
                "writer": {
                    "name": "RDBMS Writer",
                    "parameter": {
```

```
                        "connection": [
                            {
                                "jdbcUrl": "jdbc:dm://ip:port/database",
                                "table": [
                                    "table"
                                ]
                            }
                        ],
                        "username": "username",
                        "password": "password",
                        "table": "table",
                        "column": [
                            "*"
                        ],
                        "preSql": [
                            "delete from XXX;"
                        ]
                    }
                }
            }
        ]
    }
}
```

You can enable RDBMS Writer to support a new database as follows:

1. Go to the directory of RDBMS Writer, *${DATAX_HOME}/plugin/writer/RDBMS Writer*. In the preceding directory, ${DATAX_HOME} indicates the main directory of Data Integration.
2. Add the driver of your database to the drivers array in the *plugin.json* file in the RDBMS Writer directory. RDBMS Writer automatically selects an appropriate driver for connecting to a database.

```
{
    "name": "RDBMS Writer",
    "class": "com.alibaba.datax.plugin.reader.RDBMS Writer.RDBMS Writer",
    "description": "useScene: prod. mechanism: Jdbc connection using the database, exec
ute select sql, retrieve data from the ResultSet. warn: The more you know about the dat
abase, the less problems you encounter.",
    "developer": "alibaba",
    "drivers": [
        "dm.jdbc.driver.DmDriver",
        "com.ibm.db2.jcc.DB2Driver",
        "com.sybase.jdbc3.jdbc.SybDriver",
        "com.edb.Driver"
    ]
}
```

3. Add the package of the driver to the libs directory in the RDBMS Writer directory.

```
$tree
.
|-- libs
|   |-- Dm7JdbcDriver16.jar
|   |-- commons-collections-3.0.jar
|   |-- commons-io-2.4.jar
|   |-- commons-lang3-3.3.2.jar
|   |-- commons-math3-3.1.1.jar
|   |-- datax-common-0.0.1-SNAPSHOT.jar
|   |-- datax-service-face-1.0.23-20160120.024328-1.jar
|   |-- db2jcc4.jar
|   |-- druid-1.0.15.jar
|   |-- edb-jdbc16.jar
|   |-- fastjson-1.1.46.sec01.jar
|   |-- guava-r05.jar
|   |-- hamcrest-core-1.3.jar
|   |-- jconn3-1.0.0-SNAPSHOT.jar
|   |-- logback-classic-1.0.13.jar
|   |-- logback-core-1.0.13.jar
|   |-- plugin-rdbms-util-0.0.1-SNAPSHOT.jar
|   `-- slf4j-api-1.7.10.jar
|-- plugin.json
|-- plugin_job_template.json
`-- RDBMS Writer-0.0.1-SNAPSHOT.jar
```

3.6.4.23. Configure Stream Writer

This topic describes the data types and parameters supported by Stream Writer and how to configure it by using the code editor.

Stream Writer allows you to display the data obtained from a Data Integration reader on the screen or discard the data. Stream Writer is mainly applicable to performance testing for data synchronization and basic functional testing.

Parameters

print

- Description: specifies whether to display the data obtained from the reader on the screen.
- Required: No
- Default value: true

Configure Stream Writer by using the codeless UI

Currently, the codeless user interface (UI) is not supported for Stream Writer.

Configure Stream Writer by using the code editor

In the following code, a node is configured to display the data obtained from a Data Integration reader on the screen.

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"stream",
            "parameter":{},
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"stream",// The writer type.
            "parameter":{
                "print": false, // Specifies whether to display data on the screen.
                "fieldDelimiter":","// The column delimiter.
            },
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0"// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false,// Specifies whether to enable bandwidth throttling. A value o
f false indicates that the bandwidth is not throttled. A value of true indicates that the b
andwidth is throttled. The maximum transmission rate takes effect only if you set this para
meter to true.
            "concurrent":1,// The maximum number of concurrent threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

3.6.4.24. Configure Hive Writer

Hive Writer allows you to write data to HDFS and load the data to Hive. This topic describes how Hive Writer works, its parameters, and how to configure it by using the codeless UI and code editor.

Background information

Hive is a Hadoop-based data warehouse tool that is used to process large amounts of structured logs. Hive maps structured data files to a table and allows you to execute SQL statements to query data in the table.

Essentially, Hive converts Hive Query Language (HQL) or SQL statements to MapReduce programs.

- Hive stores processed data in HDFS.
- Hive uses MapReduce programs to analyze data at the underlying layer.
- Hive runs MapReduce programs on Yarn.

How it works

Hive Writer accesses a Hive metastore, parses the configuration to obtain the file storage path, file format, and column delimiter of the file to which data is written, and then writes data to the HDFS file. Hive Writer loads data in the HDFS file to the destination Hive table by using JDBC.

The underlying logic of Hive Writer is the same as that of HDFS Writer. You can configure parameters of HDFS Writer in the parameters of Hive Writer. Data Integration transparently transmits the configured parameters to HDFS Writer.

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection.	Yes	None
column	The columns to which data is written. Example: `"column": ["id","name"]`. • Column pruning is supported. You can select specific columns to export. • The column parameter must explicitly specify a set of columns to which data is written. The parameter cannot be left empty. • The column order cannot be changed.	Yes	None
table	The name of the Hive table to which data is written. **Note** The name is case-sensitive.	Yes	None
partition	The partition in the Hive table to which data is written. • This parameter is required for a partitioned Hive table. The sync node writes data to the partition specified by the partition parameter. • This parameter is not required for a non-partitioned table.	No	None

Parameter	Description	Required	Default value
writeMode	The mode in which data is loaded to the Hive table. After data is written to the HDFS file, Hive Writer executes the `LOAD DATA INPATH (overwrite) INTO TABLE` statement to load data to the Hive table. The writeMode parameter specifies the data loading mode. • *truncate*: deletes existing data before loading the data to the Hive table. • *append*: retains the existing data and appends the data to the Hive table. • If the writeMode parameter is set to Other, the data is written to the HDFS file but not loaded to the Hive table. **Note** Setting the writeMode parameter is a high-risk operation. Pay attention to the destination directory and the value of this parameter to avoid deleting data incorrectly. This parameter must be used together with the hiveConfig parameter.	Yes	None

Parameter	Description	Required	Default value
hiveConfig	The extended parameters for Hive, including hiveCommand, jdbcUrl, username, and password. • hiveCommand: the full path of the Hive client. After you run the `hive -e` command, the `LOAD DATA INPATH` statement is executed to load data based on the mode specified by the writeMode parameter. The client specified by the hiveCommand parameter provides access information about Hive. • jdbcUrl, username, and password: the information that is required to connect to Hive by using JDBC. After Hive Writer connects to Hive by using JDBC, Hive Writer executes the `LOAD DATA INPATH` statement to load data based on the mode specified by the writeMode parameter. `"hiveConfig": { "hiveCommand": "", "jdbcUrl": "", "username": "", "password": "" }` • Hive Writer allows you to write data to HDFS files by using an HDFS client. You can use the hiveConfig parameter to specify advanced settings for the HDFS client.	Yes	None

Configure Hive Writer by using the codeless UI

On the DataStudio page, double-click a data sync node, and perform the following operations on the node configuration tab that appears:

1. Configure the connections.

   Configure the connections to the source and destination data stores for the sync node.

Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and then select a connection name that you have configured in DataWorks.
Table	The table parameter in the preceding parameter description.
Partition Key Column	The partition to which data is written. The last-level partition must be specified. Hive Writer can write data to only one partition.

Parameter	Description
Writing Rule	The writeMode parameter in the preceding parameter description.

2. Configure field mapping. It is equivalent to setting the column parameter in the preceding parameter description. Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right.

GUI Element	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	Click Auto Layout to sort the fields based on specified rules.

3. Configure channel control policies.

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read data from and write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure Hive Writer by using the code editor

In the following code, a node is configured to write data to Hive in JSON format.

```
{
    "type": "job",
    "steps": [
        {
            "stepType": "hive",
            "parameter": {
            },
            "name": "Reader",
            "category": "reader"
        },
        {
            "stepType": "hive",
            "parameter": {
                "partition": "year=a,month=b,day=c", // The partition to which data is writ
ten.
                "datasource": "hive_ha_shanghai", // The connection name.
                "table": "partitiontable2", // The name of the destination table.
                "column": [// The columns in the destination table to which data is written
.
                    "id",
                    "name",
                    "age"
                ],
                "writeMode": "append" // The write mode.
            },
            "name": "Writer",
            "category": "writer"
        }
    ],
    "version": "2.0",
    "order": {
        "hops": [
            {
                "from": "Reader",
                "to": "Writer"
            }
        ]
    },
    "setting": {
        "errorLimit": {
            "record": ""
        },
        "speed": {
            "throttle": false,
            "concurrent": 2
        }
    }
}
```

3.6.4.25. Configure Vertica Writer

Vertica is a column-oriented database using the MPP architecture. Vertica Writer allows you to write data to tables stored in Vertica databases. This topic describes how Vertica Writer works, its parameters, and how to configure it by using the code editor.

How it works

Vertica Writer connects to a remote Vertica database by using JDBC, and executes an `INSERT INTO` statement to write data to the Vertica database. Internally, data is submitted to the Vertica database in batches.

Vertica Writer is designed for ETL developers to import data from data warehouses to Vertica databases. Vertica Writer can also be used as a data migration tool by users such as DBAs.

Vertica Writer obtains data from a Data Integration reader, and generates the INSERT INTO statement based on your configurations.

- `INSERT INTO`: If a primary key conflict or unique index conflict occurs, data cannot be written to the conflicting rows.
- Data can be written only to tables stored in the primary Vertica database.

  > **Note** A sync node that uses Vertica Writer must have at least the permission to execute the `INSERT INTO` statement. Whether other permissions are required depends on the SQL statements specified in the preSql and postSql parameters when you configure the node.

- Vertica Writer does not support the writeMode parameter.
- Vertica Writer accesses a Vertica database by using the Vertica database driver. Confirm the compatibility between the driver version and your Vertica database. Vertica Writer uses the following version of the Vertica database driver:

```
<dependency>
    <groupId>com.vertica</groupId>
    <artifactId>vertica-jdbc</artifactId>
    <version>7.1.2</version>
</dependency>
```

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None

Parameter	Description	Required	Default value
jdbcUrl	The JDBC URL for connecting to the Vertica database. You do not need to set this parameter because the system automatically obtains the value from the connection parameter. • You can configure only one JDBC URL for a database. Vertica Writer cannot write data to a database with multiple primary databases. • The format must be in accordance with Vertica official specifications. You can also specify the information of the attachment facility. Example: `jdbc:vertica://127.0.0.1:3306/database`.	Yes	None
username	The username that you can use to connect to the database.	Yes	None
password	The password that you can use to connect to the database.	Yes	None
table	The names of the destination tables, which are described in a JSON array. **Note** You do not need to set this parameter because the system automatically obtains the value from the connection parameter.	Yes	None
column	The columns in the destination table to which data is written. Separate the columns with a comma (,), for example, `"column": ["id","name","age"]`.	Yes	None
preSql	The SQL statement to execute before the sync node is run. Use `@table` to specify the name of the destination table in the SQL statement. When you execute this SQL statement, DataWorks replaces @table with the name of the destination table.	No	None
postSql	The SQL statement to execute after the sync node is run.	No	None
batchSize	The number of data records to write at a time. Setting this parameter can greatly reduce the interactions between Data Integration and the Vertica database over the network, and increase the throughput. However, an excessively large value may lead to the OOM error during the data synchronization process.	No	*1,024*

Configure Vertica Writer by using the codeless UI

The codeless UI is not supported for Vertica Writer.

Configure Vertica Writer by using the code editor

In the following code, a node is configured to write data to a Vertica database.

```
{
    "type":"job",
    "version":"2.0",// The version number.
     "steps":[
        {
            "stepType":"stream",
            "parameter":{},
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"vertica",// The writer type.
             "parameter":{
                "datasource": "The connection name.",
                "username": "",
                "password": "",
                "column": [// The columns to which data is written.
                     "id",
                    "name"
                ],
                "connection": [
                    {
                        "table": [// The name of the destination table.
                             "vertica_table"
                        ],
                        "jdbcUrl": "jdbc:vertica://ip:port/database"
                    }
                ],
                "preSql": [ // The SQL statement to execute before the sync node is run.
                     "delete from @table where db_id = -1"
                ],
                "postSql": [// The SQL statement to execute after the sync node is run.
                     "update @table set db_modify_time = now() where db_id = 1"
                ]
            },
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0"// The maximum number of dirty data records allowed.
         },
        "speed":{
            "throttle":false,// Specifies whether to enable bandwidth throttling. A value o
f false indicates that the bandwidth is not throttled. A value of true indicates that the b
andwidth is throttled. The maximum transmission rate takes effect only if you set this para
meter to true.
             "concurrent":1 // The maximum number of concurrent threads.
         }
    },
    "order":{
```

```
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

3.6.4.26. Configure Gbase8a Writer

This topic describes the implementation principle and parameter configurations of Gbase8a Writer.

Gbase8a Writer allows you to write data to tables stored in Gbase8a databases. At the underlying implementation level, Gbase8a Writer connects to a remote Gbase8a database through the JDBC Driver and runs the relevant SQL statements to write data to the Gbase8a database.

> **Note** You must configure a connection before configuring Gbase8a Writer.

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None
table	The name of the table to be synchronized.	Yes	None
writeMode	The write mode. Valid values: *insert into*, *on duplicate key update*, and *replace into*. • *insert into*: If a primary key conflict or unique index conflict occurs, data cannot be written to the conflicting rows and is regarded as dirty data. • *on duplicate key update*: If no primary key conflict or unique index conflict occurs, the action is the same as that of `insert into`. If a conflict occurs, specified fields in original rows are updated. • *replace into*: If no primary key conflict or unique index conflict occurs, the action is the same as that of `insert into`. If a conflict occurs, original rows are deleted and new rows are inserted. That is, all fields of original rows are replaced.	No	*insert*
column	The columns in the destination table to which data is written. Separate the columns with a comma (,). Example: `"column": ["id", "name", "age"]`. Set the value to an asterisk (*) if data is written to all the columns in the destination table. That is, set the column parameter as follows: `"column":["*"]`.	Yes	None

Parameter	Description	Required	Default value
preSql	The SQL statement to run before the sync node is run. For example, you can clear outdated data before data synchronization. Currently, you can run only one SQL statement on the codeless user interface (UI), and multiple SQL statements in the code editor. **Note** If you specify multiple SQL statements in the code editor, the system does not guarantee that they are run in the same transaction.	No	None
postSql	The SQL statement to run after the sync node is run. For example, you can add a timestamp after data synchronization. Currently, you can run only one SQL statement on the codeless UI, and multiple SQL statements in the code editor. **Note** If you specify multiple SQL statements in the code editor, the system does not guarantee that they are run in the same transaction.	No	None
batchSize	The number of data records to write at a time. Setting this parameter can greatly reduce the interactions between Data Integration and the Gbase8a database over the network, and increase the throughput. However, an excessively large value may lead to the out of memory (OOM) error during the data synchronization process.	No	*1024*

Configure Gbase8a Writer by using the codeless UI

Currently, the codeless UI is not supported for Gbase8a Writer.

Configure Gbase8a Writer by using the code editor

In the following code, a node is configured to write data to the Gbase8a database. For more information about the parameters, see the preceding parameter description.

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"stream",
            "parameter":{},
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"gbase8a",// The writer type.
            "parameter":{
                "postSql":[],// The SQL statement to run after the sync node is run.
                "datasource":"",// The connection name.
                "column":[// The columns to be synchronized.
                    "id",
                    "value"
                ],
                "writeMode":"insert",// The write mode.
                "batchSize":1024,// The number of data records to write at a time.
                "table":"",// The name of the table to be synchronized.
                "preSql":[]// The SQL statement to run before the sync node is run.
            },
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{// The maximum number of dirty data records allowed.
            "record":"0"
        },
        "speed":{
            "throttle":false,// Specifies whether to enable bandwidth throttling.
            "concurrent":1,// The maximum number of concurrent threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

3.6.4.27. KingbaseES Writer

This topic describes the parameters that are supported by KingbaseES Writer and how to configure KingbaseES Writer by using the codeless user interface (UI) and code editor.

Context

KingbaseES Writer writes data to tables stored in KingbaseES databases. KingbaseES Writer connects to a remote KingbaseES database by using Java Database Connectivity (JDBC) and executes the `INSERT INTO` or `REPLACE INTO` statement to write data to the database. KingbaseES uses the InnoDB engine so that data is written to the database in batches.

KingbaseES Writer can also be used as a data migration tool by users such as database administrators. KingbaseES Writer obtains protocol data from a Data Integration reader, and writes the data to the destination database based on the value of the writeMode parameter.

> **Note** A synchronization node that uses kingbaseES Writer must have at least the permissions to execute the `INSERT INTO` or `REPLACE INTO` statement. Whether other permissions are required depends on the SQL statements that you specify in the preSql and postSql parameters when you configure the node.

Parameters

Parameter	Description	Required	Default value
datasource	The name of the data source. It must be the same as the name of the added data source. You can add data sources by using the code editor.	Yes	No default value
table	The name of the table to which you want to write data.	Yes	No default value
column	The names of the columns to which you want to write data. Separate the names with commas (,), such as `"column": ["id", "name", "age"]`. If you want to write data to all the columns in the destination table, set this parameter to an asterisk (*), such as `"column":["*"]`.	Yes	No default value
preSql	The SQL statement that you want to execute before the synchronization node is run. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor. For example, you can set this parameter to the following statement to delete outdated data before the synchronization node is run: `truncate table tablename` **Note** If you specify multiple SQL statements, they may not be executed in the same transaction.	No	No default value

Parameter	Description	Required	Default value
postSql	The SQL statement that you want to execute after the synchronization node is run. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor. For example, you can set this parameter to `alter table tablenameadd colname timestamp DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP` to add a timestamp after the synchronization node is run.	No	No default value
batchSize	The number of data records to write at a time. Set this parameter to an appropriate value based on your business requirements. This greatly reduces the interactions between Data Integration and the database and increases throughput. If you set this parameter to an excessively large value, an out of memory (OOM) error may occur during data synchronization.	No	*2048*

Configure KingbaseES Writer by using the codeless UI

1. Configure data sources.

   Log on to the DataWorks console. The **DataStudio** page appears. On the DataStudio page, move the pointer over the icon and choose **Data Integration > Batch Synchronization**. In the **Create Node** dialog box, configure the parameters to create a batch synchronization node.

   On the configuration tab of the batch synchronization node, configure Source and **Target** for the node.

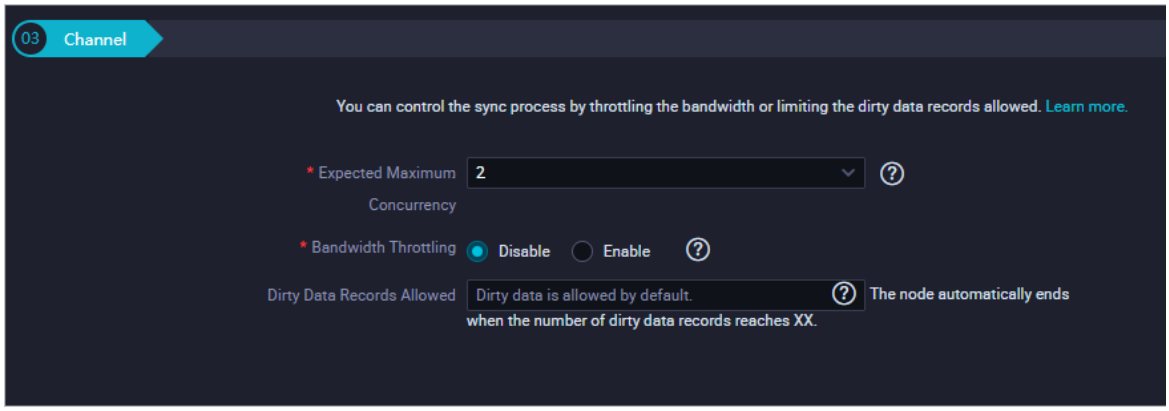
Parameter	Description
Connection	The name of the data source to which you want to write data. This parameter corresponds to the datasource parameter that is described in the preceding section. Select the name of a data source that you configured.
Table	The name of the table to which you want to write data. This parameter corresponds to the table parameter that is described in the preceding section.
Pre sql	The SQL statement that you want to execute before the synchronization node is run. This parameter corresponds to the preSql parameter that is described in the preceding section. Enter the SQL statement that you want to execute before the synchronization node is run.
Post sql	The SQL statement that you want to execute after the synchronization node is run. This parameter corresponds to the postSql parameter that is described in the preceding table. Enter the SQL statement that you want to execute after the synchronization node is run.

Parameter	Description
Data Records Per Write	The number of data records to write at a time. This parameter corresponds to the batchSize parameter that is described in the preceding section. Valid values: *2048*. You can specify this parameter based on your business requirements.

2. Configure field mappings. This operation is equivalent to setting the column parameter that is described in the preceding section. Fields in the source on the left have a one-to-one mapping with fields in the destination on the right.

Operation	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish mappings between fields with the same name. The data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish mappings between fields in the same row. The data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove the mappings that are established.
Auto Layout	Click Auto Layout. Then, the system automatically sorts the fields based on the specified rules.
Add	Click **Add** to add a field. Take note of the following rules when you add a field: ○ You can enter constants. Each constant must be enclosed in single quotation marks ('), such as 'abc' and '123'. ○ You can use scheduling parameters, such as ${bizdate}. ○ You can enter functions that are supported by relational databases, such as now() and count(1). ○ If the value that you entered cannot be parsed, the value of Type for the field is Unidentified.

3. Configure channel control policies.



Parameter	Description
Expected Maximum Concurrency	The maximum number of parallel threads that the synchronization node uses to read data from the source or write data to the destination. You can configure the parallelism for the synchronization node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workloads on the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure KingbaseES Writer by using the code editor

In the following code, a synchronization node is configured to write data to KingbaseES:

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"stream",
            "parameter":{},
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"kingbasees",// The writer type.
            "parameter":{
                "postSql":[],// The SQL statement that you want to execute after the synchr
onization node is run.
                "datasource":"",// The name of the data source.
                "column":[// The names of the columns to which you want to write data.
                    "id",
                    "value"
                ],
                "batchSize":2048,// The number of data records to write at a time.
                "table":"",// The name of the table to which you want to write data.
                "preSql":[
                     "delete from XXX;"// The SQL statement that you want to execute before
the synchronization node is run.
                  ]
            },
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{// The maximum number of dirty data records allowed.
            "record":"0"
        },
        "speed":{
            "throttle":false,// Specifies whether to enable bandwidth throttling.
            "concurrent":1// The maximum number of parallel threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

3.6.4.28. SAP HANA Writer

This topic describes the parameters that are supported by SAP HANA Writer and how to configure SAP HANA Writer by using the codeless user interface (UI) and code editor.

Context

SAP HANA Writer writes data to tables stored in SAP HANA databases. SAP HANA Writer connects to a remote SAP HANA database by using Java Database Connectivity (JDBC) and executes the `INSERT INTO` or `REPLACE INTO` statement to write data to the SAP HANA database. SAP HANA uses the InnoDB engine so that data is written to the database in batches.

SAP HANA Writer can also be used as a data migration tool by users such as database administrators.

> **Note** A synchronization node that uses SAP HANA Writer must have at least the permissions to execute the `INSERT INTO` or `REPLACE INTO` statement. Whether other permissions are required depends on the SQL statements that you specify in the preSql and postSql parameters when you configure the node.

Parameters

Parameter	Description	Required	Default value
datasource	The name of the data source. It must be the same as the name of the added data source. You can add data sources by using the code editor.	Yes	No default value
table	The name of the table to which you want to write data.	Yes	No default value
column	The names of the columns to which you want to write data. Separate the names with commas (,), such as `"column": ["id", "name", "age"]`. If you want to write data to all the columns in the destination table, set this parameter to an asterisk (*), such as `"column":["*"]`.	Yes	No default value
preSql	The SQL statement that you want to execute before the synchronization node is run. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor. For example, you can set this parameter to the following SQL statement that is used to delete outdated data: `truncate table tablename` **Note** If you specify multiple SQL statements in the code editor, the SQL statements cannot be executed in the same transaction.	No	No default value

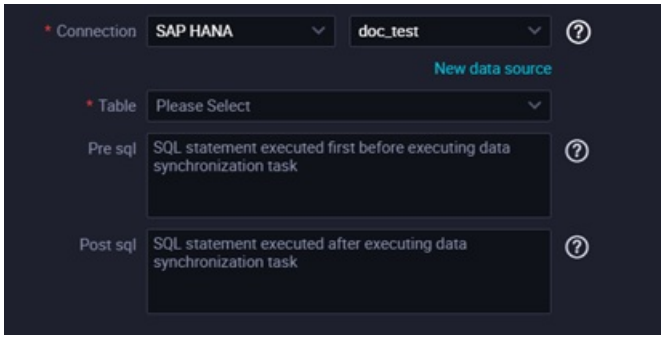
Parameter	Description	Required	Default value
postSql	The SQL statement that you want to execute after the synchronization node is run. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor. For example, you can set this parameter to `alter table tablenameadd colname timestamp DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP` that is used to add a timestamp.	No	No default value
batchSize	The number of data records to write at a time. Set this parameter to an appropriate value based on your business requirements. This greatly reduces the interactions between Data Integration and SAP HANA and increases throughput. If you set this parameter to an excessively large value, an out of memory (OOM) error may occur during data synchronization.	No	*2048*

Configure SAP HANA Writer by using the codeless UI

1. Configure data sources.

   Log on to the DataWorks console. The **DataStudio** page appears. On the DataStudio page, move the pointer over the icon and choose **Data Integration > Batch Synchronization**. In the **Create Node** dialog box, configure the parameters to create a batch synchronization node.

   Configure Source and **Target** for the synchronization node.



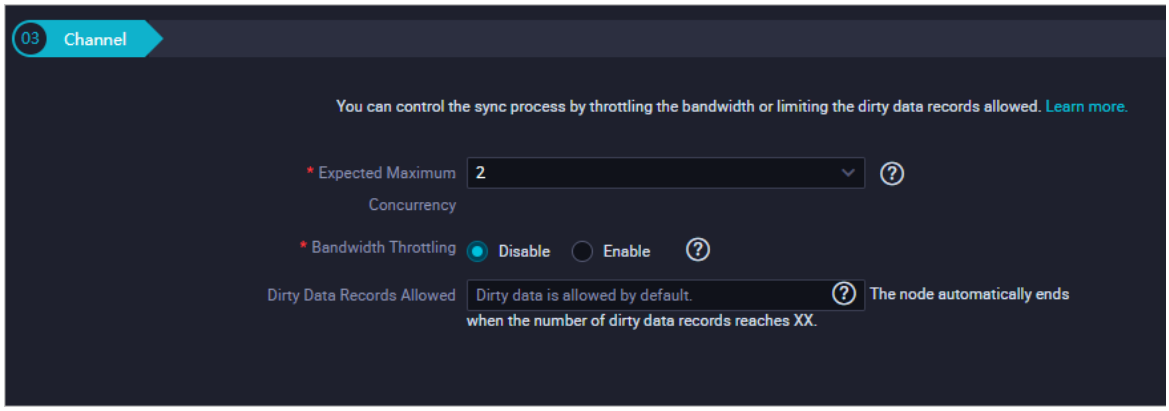
Parameter	Description
Connection	The name of the data source to which you want to write data. This parameter corresponds to the datasource parameter that is described in the preceding section.
Table	The name of the table to which you want to write data. This parameter corresponds to the table parameter that is described in the preceding section.
Pre sql	The SQL statement that you want to execute before the synchronization node is run. This parameter corresponds to the preSql parameter that is described in the preceding section.

Parameter	Description
Post sql	The SQL statement that you want to execute after the synchronization node is run. This parameter corresponds to the postSql parameter that is described in the preceding section.

2. Configure field mappings. This operation is equivalent to setting the column parameter that is described in the preceding section. Fields in the source on the left have a one-to-one mapping with fields in the destination on the right.

Operation	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish mappings between fields with the same name. The data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish mappings between fields in the same row. The data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove the mappings that are established.
Auto Layout	Click Auto Layout. Then, the system automatically sorts the fields based on specific rules.
Add	Click **Add** to add a field. Take note of the following rules when you add a field: ○ You can enter constants. Each constant must be enclosed in single quotation marks ('), such as 'abc' and '123'. ○ You can use scheduling parameters, such as ${bizdate}. ○ You can enter functions that are supported by relational databases, such as now() and count(1). ○ If the field that you entered cannot be parsed, the value of Type for the field is Unidentified.

3. Configure channel control policies.



Parameter	Description
Expected Maximum Concurrency	The maximum number of parallel threads that the synchronization node uses to read data from the source or write data to the destination. You can configure the parallelism for the synchronization node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workloads on the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to an appropriate value based on the configurations of the source.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure SAP HANA Writer by using the code editor

In the following code, a synchronization node is configured to write data to SAP HANA:

```
{
    "type":"job",
    "version":"2.0",// The version number.
    "steps":[
        {
            "stepType":"stream",
            "parameter":{},
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"saphana",// The writer type.
            "parameter":{
                "postSql":[],// The SQL statement that you want to execute after the synchr
onization node is run.
                "datasource":"",// The name of the data source.
                "column":[// The names of the columns to which you want to write data.
                    "id",
                    "value"
                ],
                "batchSize":1024,// The number of data records to write at a time.
                "table":"",// The name of the table to which you want to write data.
                "preSql":[
                     "delete from XXX;" // The SQL statement that you want to execute befor
e the synchronization node is run.
                   ]
            },
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{// The maximum number of dirty data records allowed.
            "record":"0"
        },
        "speed":{
            "throttle":false,// Specifies whether to enable bandwidth throttling. The value
false indicates that bandwidth throttling is disabled, and the value true indicates that ba
ndwidth throttling is enabled. The mbps parameter takes effect only when the throttle param
eter is set to true.
            "concurrent":1 // The maximum number of parallel threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

3.6.4.29. Configure ClickHouse Writer

ClickHouse is an open source column-oriented database management system (DBMS) for online analytical processing (OLAP) of queries. This topic describes how ClickHouse Writer works, the parameters that are supported by ClickHouse Writer, and how to configure ClickHouse Writer by using the codeless user interface (UI) and code editor.

Limits

- ClickHouse Writer connects to a ClickHouse database by using Java Database Connectivity (JDBC) and can write data to a destination table in the ClickHouse database only by using JDBC Statement.
- ClickHouse Writer allows you to specify the columns to which you want to write data. You can specify the columns in an order different from the order specified by the schema of the destination table.
- If ClickHouse Writer writes data in INSERT mode, we recommend that you throttle the transactions per second (TPS) to 1,000 to prevent high workloads on ClickHouse.
- After ClickHouse Writer writes all required data, ClickHouse Writer performs a single-process POST Flush operation to update the data records in the ClickHouse database.
- You must make sure that the driver version is compatible with your ClickHouse database. ClickHouse Writer supports only the following version of the ClickHouse database driver:

```
<dependency>
    <groupId>ru.yandex.clickhouse</groupId>
    <artifactId>clickhouse-jdbc</artifactId>
    <version>0.2.4.ali2-SNAPSHOT</version>
</dependency>
```

Background information

ClickHouse Writer writes data to ClickHouse databases. ClickHouse Writer connects to a remote ClickHouse database by using JDBC and executes an `INSERT INTO` statement to write data to the ClickHouse database.

ClickHouse Writer is designed for extract, transform, load (ETL) developers to import data from data warehouses to ClickHouse databases. ClickHouse Writer can also be used as a data migration tool by users such as database administrators.

ClickHouse Writer obtains data from a reader, generates an INSERT INTO statement based on your configurations, and then executes the INSERT INTO statement to write data to ClickHouse databases.

Parameters

Parameter	Description	Required	Default value

Parameter	Description	Required	Default value
jdbcUrl	The JDBC URL of the ClickHouse database. The jdbcUrl parameter must be included in the connection parameter. • You can configure only one JDBC URL for a database. • The value format of the jdbcUrl parameter must be in accordance with the official specifications of ClickHouse. You can also specify additional JDBC connection properties in the value of this parameter. Example: `jdbc:clickhouse://127.0.0.1:3306/database`.	Yes	No default value
username	The username that you can use to connect to the database.	Yes	No default value
password	The password that you can use to connect to the database.	Yes	No default value
table	The name of the table to which you want to write data. Specify the name in a JSON array. **Note** The table parameter must be included in the connection parameter.	Yes	No default value
column	The names of the columns to which you want to write data in the destination table. Separate the names with commas (,), such as `"column": ["id", "name", "age"]`. **Note** The column parameter cannot be left empty.	Yes	No default value
preSql	The SQL statement that you want to execute before the synchronization node is run.	No	No default value
postSql	The SQL statement that you want to execute after the synchronization node is run.	No	No default value
batchSize	The number of data records to write at a time. Set this parameter to an appropriate value based on your business requirements. This greatly reduces the interactions between Data Integration and ClickHouse and increases throughput. If you set this parameter to an excessively large value, an out of memory (OOM) error may occur during data synchronization.	No	*1024*

Configure ClickHouse Writer by using the codeless UI

This method is not supported.

Configure ClickHouse Writer by using the code editor

For more information about how to configure a synchronization node by using the code editor, see Create a synchronization node by using the code editor.

> **Note** Delete the comments from the following code before you run the code.

In the following code, a synchronization node is configured to write data to a ClickHouse database:

```
{
    "type":"job",
    "version":"2.0", // The version number.
    "steps":[
        {
            "stepType":"stream",
            "parameter":{},
            "name":"Reader",
            "category":"reader"
        },
        {
            "stepType":"clickhouse", // The writer type.
            "parameter":{
                "username": "",
                "password": "",
                "column": [ // The names of the columns to which you want to write data.
                    "id",
                    "name"
                ],
                "connection": [
                    {
                        "table": [ // The name of the table to which you want to write data
.
                            "ClickHouse_table"
                        ],
                        "jdbcUrl": "jdbc:clickhouse://ip:port/database"
                    }
                ],
                "preSql": [ // The SQL statement that you want to execute before the synchr
onization node is run.
                    "delete from table where db_id = -1"
                ],
                "postSql": [ // The SQL statement that you want to execute after the synchr
onization node is run.
                    "update table set db_modify_time = now() where db_id = 1"
                ],
                "batchSize": "1024",
                "batchByteSize": "67108864",
                "writeMode": "insert"
            },
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
```

```
        "errorLimit":{
            "record":"0" // The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":true, // Specifies whether to enable bandwidth throttling. A value o
f false indicates that bandwidth throttling is disabled, and a value of true indicates that
bandwidth throttling is enabled. The mbps parameter takes effect only if the throttle param
eter is set to true.
            "concurrent":1, // The maximum number of parallel threads.
            "mbps":"12" // The maximum transmission rate.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

3.6.4.30. Configure TSDB Writer

TSDB Writer can write data points to Alibaba Cloud Time Series Database (TSDB) databases. This topic describes the data types and parameters that are supported by TSDB Writer and how to configure TSDB Writer by using the code editor.

Background information

TSDB is a high-performance, cost-effective, stable, and reliable online database service. TSDB features high read and write performance and provides a high compression ratio for data storage. TSDB also enables the interpolation and aggregation of time series data. TSDB can be used in various systems, such as IoT device monitoring systems, energy management systems (EMSs) for enterprises, security monitoring systems for production, and electricity consumption monitoring systems.

You can write millions of data points to TSDB within seconds. TSDB provides the following features: high compression ratio, cost-effective data storage, downsampling, interpolation, multi-dimensional aggregation, and visualized query results. These features help you resolve issues that are caused by a large number of data collection points on devices and frequent data collection. The issues include high storage costs and low write and query efficiency.

TSDB Writer connects to a TSDB instance by sending an HTTP request and writes data points by using the `/api/put` HTTP API operation.

Limits

- DataWorks supports only batch synchronization of TSDB data by using the code editor.
- TSDB Writer supports TSDB V2.4.X and later.

Data types

TSDB Writer can convert internal data types supported by Data Integration to data types supported by TSDB databases. The following table describes the conversion relationships.

Category	Data Integration data type	TSDB data type
String	STRING	String to which a data point in TSDB is serialized. The data point can be a timestamp, metric, tag, or value.

Parameters

The following table describes the parameters that you must set when you synchronize TSDB data by using the code editor.

Parameter type	Parameter	Description	Default value
Common parameters	sourceDbType	The type of the destination database. Valid values: TSDB and RDB. • A value of TSDB indicates that the destination database is an OpenTSDB, Prometheus, or Timescale database. • A value of RDB indicates that the destination database is a relational database, such as a MySQL, Oracle, PostgreSQL, or Distributed Relational Database Service (DRDS) database.	*TSDB*
	endpoint	The HTTP endpoint of the destination TSDB database. Specify the endpoint in the format of http://IP address:Port number.	No default value
	batchSize	The number of data records to write at a time. The value of this parameter is of the INT type and must be greater than 0.	*100*
	maxRetryTime	The maximum number of retries to write data to the table after the data write fails.	*3*

Parameters for TSDB	Parameter	Description	Default value
Parameter type			
	ignoreWriteError	The processing policy when the data write fails. Valid values: • *true*: ignores write errors and continues to write data. If the number of retries exceeds the specified maximum number, the data write is terminated. **Note** You can set the maxRetryTime parameter to specify the maximum number of retries. • *false*: terminates the data write.	*false*
	endpoint	The HTTP endpoint of the destination relational database. Specify the endpoint in the format of http://IP address:Port number.	No default value
	column	The names of the columns to which you want to write data. **Note** You must specify the columns in the same order as the columns specified for a reader. For more information, see TSDB Reader.	No default value
Parameters for			

RDB Parameter type	Parameter	Description	Default value
	columnType	The types of the columns in the relational database. The following types are supported: • timestamp: a timestamp column. • tag: a tag column. • metric_num: a metric column whose value is of a numeric data type. • metric_string: a metric column whose value is of a string data type. **Note** You must specify the columns in the same order as the columns specified for a reader. For more information, see TSDB Reader.	No default value
	batchSize	The number of data records to write at a time. The value of this parameter is of the INT type and must be greater than 0.	*100*

Configure TSDB Writer by using the code editor

For more information about how to configure a synchronization node by using the code editor, see Create a synchronization node by using the code editor. In the following code, a node is configured to write data to a TSDB database.

```
{
    "order": {
        "hops": [
            {
                "from": "Reader",
                "to": "Writer"
            }
        ]
    },
    "setting": {
        "errorLimit": {
            "record": "0"
        },
        "speed": {
            "throttle":true, // Specifies whether to enable bandwidth throttling. A value o
f false indicates that bandwidth throttling is disabled, and a value of true indicates that
```

```
bandwidth throttling is enabled. The mbps parameter takes effect only if the throttle param
eter is set to true.
            "concurrent":1, // The maximum number of parallel threads.
            "mbps":"12" // The maximum transmission rate.
        }
    },
    "steps": [
        {
            "category": "reader",
            "name": "Reader",
            "parameter": {},
            "stepType": ""
        },
        {
            "category": "writer",
            "name": "Writer",
            "parameter": {
                "endpoint": "http://localhost:8242",
                "sourceDbType": "RDB",
                "batchSize": 256,
                "column": [
                    "name",
                    "type",
                    "create_time",
                    "price"
                ],
                "columnType": [
                    "tag",
                    "tag",
                    "timestamp",
                    "metric_num"
                ]
            },
            "stepType": "tsdb"
        }
    ],
    "type": "job",
    "version": "2.0"
}
```

Performance test report

- Characteristics of test data
  - Metric: a metric, which is specified as m.

- tagkv: the *zone*, *cluster*, *group*, and *app* keys that form time series. The *ip* key contains up to 2,000,000 time series, which start from 1. For example, the number of time series is calculated by using the following formula: `10 × 20 × 100 × 100 = 2,000,000`.

tag_k	tag_v
zone	z1–z10
cluster	c1–c20
group	g1–g100
app	a1–a100
ip	ip1–ip2,000,000

  - value: a random value from 1 to 100.
  - interval: a collection interval of 10 seconds. The total duration of data collection is 3 hours, and a total number of 2,160,000,000 data points are collected. The number of data points is calculated by using the following formula: `3 × 60 × 60/10 × 2,000,000 = 2,160,000,000`.

- Performance test results

Number of channels	Data integration speed (record/s)	Data integration bandwidth (Mbit/s)
1	129,753	15.45
2	284,953	33.70
3	385,868	45.71

3.6.5. Optimize synchronization performance

This topic describes how to maximize the synchronization speed by adjusting the concurrency configuration, the difference between nodes that are configured with bandwidth throttling and those that are not, and precautions for custom resource groups.

Data Integration is a one-stop platform that supports real-time and offline data synchronization between any connections in any location and in any network environment. You can synchronize 10 TB of data between various types of cloud storage and local storage each day.

DataWorks provides excellent data transmission performance and supports data exchanges between more than 400 pairs of disparate connections. These features allow you to focus on the key issues on constructing big data solutions.

Factors affecting the speed of data synchronization

The factors that affect the speed of data synchronization are listed as follows:

- Source
  - Database performance: the performance of the CPU, memory module, SSD, network, and hard disk.
  - Concurrency: A high concurrency results in a heavy database workload.

  - Network: the bandwidth (throughput) and speed of the network. Generally, a database with better performance can support more concurrent nodes and a larger concurrency value can be set for sync nodes.

- Sync node
  - Synchronization speed: whether an upper limit is set for the synchronization speed.
  - Concurrency: a maximum number of concurrent threads to read data from the source and write data to destination data storage within a single sync node.
  - Nodes that are waiting for resources.
  - Bandwidth throttling: The bandwidth of a single thread is 1,048,576 bit/s. Timeout occurs when the business is sensitive to the network speed. We recommend that you set a smaller value.
  - Whether to create an index for query statements.

- Destination
  - Performance: the performance of the CPU, memory module, SSD, network, and hard disk.
  - Load: Excessive load in the destination database affects the write efficiency within the sync nodes.
  - Network: the bandwidth (throughput) and speed of the network.

You need to monitor and optimize the performance, load, and network of the source and destination databases. The following describes the optimal settings of a sync node.

Concurrency

You can configure the concurrency for a node on the codeless user interface (UI). The following is an example of how to configure the concurrency in the code editor:

```
"setting": {
      "speed": {
        "concurrent": 10
      }
    }   }
```

Bandwidth throttling

By default, bandwidth throttling is disabled. In a sync node, data is synchronized at the maximum speed given the concurrency configured for the node. Considering that excessively fast synchronization may overstress the database and thus affect the production, Data Integration allows you to limit the synchronization speed and optimize the configuration as required. If bandwidth throttling is enabled, we recommend that you limit the maximum speed to 30 Mbit/s. The following is an example for configuring an upper limit for synchronization speed in the code editor, in which the transmission bandwidth is 1 Mbit/s:

```
"setting": {
      "speed": {
         "throttle": true // The bandwidth throttling is enabled.
        "mbps": 1, // The synchronization speed.
      }
    }
```

Note

- When the throttle parameter is set to false, throttling is disabled, and you do not need to configure the mbps parameter.
- The bandwidth value is a Data Integration metric and does not represent the actual network interface card (NIC) traffic. Generally, the NIC traffic is two to three times of the channel traffic, which depends on the serialization of the data storage system.
- A semi-structured file does not have shard keys. If multiple files exist, you can set the maximum job speed to increase the synchronization speed. However, the maximum job speed is limited by the number of files. For example, the maximum job speed limit is set to n Mbit/s for n files. If you set the limit to n+1 Mbit/s, the synchronization speed remains at n Mbit/s. If you set the limit to n-1 Mbit/s, the synchronization is performed at n-1 Mbit/s.
- A table can be partitioned according to the preset maximum job speed only when a maximum job speed and a shard key are configured for a relational database. Relational databases only support numeric shard keys, while Oracle databases support both numeric and string shard keys.

Scenarios of slow data synchronization

- Scenario 1: Resolve the issue that sync nodes to be run on the default resource group remain waiting for resources.
  - Example

    When you test a sync node in DataWorks, the node remains waiting for resources and an internal system error occurs.

    For example, a sync node is configured to synchronize data from RDS to MaxCompute. The node has waited for about 800 seconds before it is run successfully. However, the log shows that the node runs for only 18 seconds and then stops. The sync node uses the default resource group. When you run other sync nodes, they also remain in the waiting state.

    The log is displayed as follows:

    ```
    2017-01-03 07:16:54 : State: 2(WAIT) | Total: 0R 0B | Speed: 0R/s 0B/s | Error: 0R 0B |
    Stage: 0.0%
    ```

  - Solution

    The default resource group is not exclusively used by a single user. It is used by many projects concurrently, not just two or three nodes for a single user. If such resources are insufficient after you start to run a node, the node needs to wait for resources. In this case, the node is completed 800 seconds after you start running the node, but it only takes 10 seconds for the node to be executed.

    To improve the synchronization speed and reduce the waiting time, we recommend that you run sync nodes during off-peak hours. Typically, most sync nodes are run between 00:00 and 03:00.

- Scenario 2:

  Accelerate nodes that synchronize data from multiple source tables to the same destination table.

  - Example

    To synchronize data from tables of multiple data stores to a table, you configure multiple sync nodes to run in sequence. However, the synchronization takes a long time.

- Solution

  To launch multiple concurrent nodes that write data to the same destination database, pay attention to the following points:

  - Ensure that the destination database can support the execution of all the concurrent nodes.
  - You can configure a sync node that synchronizes multiple source tables to the same destination table. Alternatively, you can configure multiple nodes to run concurrently in the same workflow.
  - If resources are insufficient, you can configure sync nodes to run during off-peak hours.

- Scenario 3:

  If no index is added when the WHERE clause is used, a full table scan slows down the data synchronization.

  - Example

    SQL statement:

    ```
    select bid,inviter,uid,createTime from `relatives` where createTime>='2016-10-23 00:00:
    00'and reateTime<'2016-10-24 00:00:00';
    ```

    Assume that the sync node started to run the preceding statement at 2016-10-25 11:01:24 and started to return results from 2016-10-25 11:11:05. It took a long time to finish the sync node.

  - Cause

    When the WHERE clause is used for a query, the createTime column is not indexed, resulting in a full table scan.

  - Solution

    We recommend that you use an indexed column or add an index to the column that you want to scan if you use the WHERE clause.

3.7. Real-time synchronization

3.7.1. Overview

This topic describes the information about the real-time synchronization feature, including the benefits and supported data sources. This topic also describes how to use the real-time synchronization feature.

How it works

The real-time synchronization feature allows you to synchronize data changes of a table or all tables in a source to a destination in real time. For example, after you insert, modify, or delete the data in a source database, the real-time synchronization feature synchronizes these changes to a destination database in real time. If you perform both real-time synchronization and full synchronization for historical data, all the data in your source database is synchronized to your destination database. This way, data in the destination is consistent with data in the source in real time.

Benefits

- Diverse data sources
  - Multiple types of data sources are supported. You can synchronize data from different types of data sources to different destinations.

  - The feature will soon allow you to synchronize data from a single source to multiple destinations at the same time.

- Synchronization solutions
  - You can configure a synchronization solution to synchronize the full data or incremental data from a database to MaxCompute or Hologres.
  - The feature synchronizes full data first and then continuously synchronizes incremental data to the destination database based on the synchronization solution that you configure.
- Diverse synchronization methods

  You can synchronize data from table shards, a single table in a source, or multiple tables in a source, and configure different processing rules for messages about different DDL operations of tables.
- Data processing

  You can perform data filtering and string replacement on the data from a source and synchronize the processed data to a destination.
- Monitoring and alerting
  - The feature monitors service latency, failovers, dirty data, heartbeats, and failures during synchronization.
  - The system can send you alert notifications by email, phone call, or DingTalk message.
- Small impact on the source

  The feature is optimized to have a small impact on the source.
- Graphical development
  - You can perform drag-and-drop operations instead of writing code to develop real-time synchronization nodes.
  - The feature is easy to use for beginners.

Supported data sources

- Source: MySQL Binlog, Kafka, DataHub, LogHub, and PolarDB
- Destination: MaxCompute, Hologres, Kafka, and DataHub
- Data processing: data filtering and string replacement

How to use the real-time synchronization feature

You can create real-time synchronization nodes or synchronization solutions.

- For more information about how to create a real-time synchronization node, see Create a real-time synchronization node.
- For more information about how to create a synchronization solution, see Go to the Sync Solutions page.

3.7.2. Plug-ins for data sources that support real-time synchronization

You can use the reader, writer, and conversion plug-ins for various data sources to synchronize data in real time. This topic describes the plug-ins for data sources that support real-time synchronization.

Plug-in type	Plug-in name	References
Reader	MySQL Binlog Reader	MySQL binlogs
	DataHub Reader	DataHub
	LogHub Reader	LogHub
	Kafka Reader	Kafka
	PolarDB Reader	Configure PolarDB Reader
Writer	Hologres Writer	Configure Hologres Writer
	DataHub Writer	DataHub
	Kafka Writer	Kafka
	MaxCompute Writer	Configure MaxCompute Writer
Conversion	Data Filtering	Data filter
	String Replacement	String replacement

Note You cannot run a real-time synchronization node on the node configuration tab. Instead, you must run a real-time synchronization node in the production environment after you save and commit the node.

Basic configuration

After you configure the reader, writer, and conversion plug-ins, click the **Basic Configuration** tab in the right-side navigation pane to configure the real-time synchronization node.

Parameter	Description
Description	The description of the real-time synchronization node.
JVM parameters	The Java virtual machine (JVM) memory allocated for the real-time synchronization node. If this parameter is not specified, Data Integration automatically allocates JVM memory based on your node configurations.
Dirty Data Records Allowed	The maximum number of dirty data records allowed. If you set this parameter to 0, no dirty data records are allowed. If you do not specify this parameter, dirty data records are allowed by default.

3.7.3. Create, configure, commit, and manage real-time synchronization nodes

DataWorks allows you to synchronize data in real time. This topic describes how to create, configure, commit, and manage real-time synchronization nodes.

Create a real-time synchronization node

1. Log on to the DataWorks console.
2. On the DataStudio page, move the pointer over the Create icon and choose **Data Integration > Real-time synchronization**.

   Alternatively, you can find the required workflow, right-click the workflow name, and then choose **Create > Data Integration > Real-time synchronization**.
3. In the **Create Node** dialog box, configure the parameters.

Parameter	Description
Node Type	The type of the node. Default value: **Real-time synchronization**.
Sync Method	The method used to synchronize data. Valid values: ○ **End-to-end ETL**: synchronizes data in one table to one or more tables. Data type conversion is supported during the synchronization. ○ **Migration to Hologres**: synchronizes all or some tables in a database to Hologres. Destination tables can be automatically created in Hologres. ○ **Migration to MaxCompute**: synchronizes all or some tables in a database to MaxCompute. ○ **Migration to DataHub**: synchronizes all or some tables in a database to DataHub.
Node Name	The name of the node. The name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
Location	The directory in which the real-time synchronization node is stored.

4. Click **Commit**.

Configure the real-time synchronization node

The operations that you can perform on the configuration tab of the real-time synchronization node vary based on the synchronization method that you selected.

- To configure the real-time synchronization node for which **Sync Method** is set to **End-to-end ETL**, perform the following steps:
  i. Click **Basic configuration** in the right-side navigation pane. In the Basic configuration panel, select the required resource group from the **Resource Group** drop-down list.

No.	Description
1	The left-side navigation tree, which consists of the **Input**, **Output**, and **Conversion** sections.
2	The configuration canvas of the real-time synchronization node. You can drag a component from the navigation tree to the canvas and configure the component.

No.	Description
3	The property configuration panel of the real-time synchronization node. This panel appears after you click **Basic configuration** in the right-side navigation pane.

ii. Drag components from the navigation tree to the canvas, and draw lines to connect the components. This way, the components synchronize data based on the connections.

iii. Click each component on the canvas. In the panel that appears, configure the parameters.

iv. Click the icon in the top toolbar.

- To configure the real-time synchronization node for which **Sync Method** is set to **Migration to Hologres**, perform the following steps:

  i. Click **Basic configuration** in the right-side navigation pane. In the Basic configuration panel, select the required resource group from the **Resource Group** drop-down list.

  > **Notice** You must select a **resource group** before you commit the node. Otherwise, the system returns an error when you commit the node.

  ii. In the **Data source** section, specify **Type** and **Data source**.

  iii. In the **Select the source table for synchronization** section, select the tables that you want to synchronize in the **SOURCE Table** list and click the > icon to move the tables to the **Selected Source table** list.

  The SOURCE Table list displays all the tables in the source. You can select all or some tables to synchronize them at a time.

  > **Notice** If a selected table does not have a primary key, the table cannot be synchronized in real time.

  iv. (Optional)In the **Set synchronization rules** section, click **Add rule** and select an option to configure naming rules for destination tables.

  Supported options include **Table name conversion rules** and **Target table name rule**.

  - **Table name conversion rules**: the rule for converting the names of source tables to those of destination tables.
  - **Target table name rule**: the rule for adding a prefix and suffix to the converted names of destination tables.

  v. Click **Next Step**.

  vi. In the **Set target table** step, specify **Target Hologres data source** and **Schema**.

  vii. Click **Reload source table and Hologres Table mapping** to configure the mappings between the source tables and destination Hologres tables.

  viii. View the mapping progress, source tables, and mapped destination tables, and click **Next Step**.

  You can view the mapping progress between the source tables and destination tables. The mapping may take a long period of time if you want to synchronize a large number of tables.

An error message appears if the selected source table does not have a primary key. The synchronization can be performed if one of the selected source tables has a primary key. Source tables without primary keys are ignored during the synchronization.

You can set Table creation method to **Create tables automatically** or **Use existing Table**. The name of the destination table that appears in the **Hologres Table name** column varies based on the setting of Table creation method.

- If you set **Table creation method** to **Create tables automatically**, the name of the destination table that is automatically created appears. You can click the table name to view and modify the table creation statements.
- If you set Table creation method to **Use existing Table**, you must select a table from the drop-down list in the **Hologres Table name** column.

ix. In the **Run resource settings** step, specify **Maximum number of connections supported by source read** and **Number of concurrent writes on the target side**. Then, click the icon in the top toolbar.

- To configure the real-time synchronization node for which **Sync Method** is set to **Migration to MaxCompute**, perform the following steps:

i. Click **Basic configuration** in the right-side navigation pane. In the Basic configuration panel, select the required resource group from the **Resource Group** drop-down list.

ii. In the **Data source** section, specify **Type** and **Data source**.

iii. In the **Select the source table for synchronization** section, select the tables that you want to synchronize in the **SOURCE Table** list and click the icon to move the tables to the **Selected Source table** list.

The SOURCE Table list displays all the tables in the source. You can select all or some tables to synchronize them at a time.

> **Notice** If a selected table does not have a primary key, the table cannot be synchronized in real time.

iv. In the **Set synchronization rules** section, click **Add rule** and select an option to configure naming rules for destination tables.

Supported options include **Table name conversion rules** and **Target table name rule**.

- **Table name conversion rules**: the rule for converting the names of source tables to those of destination tables.
- **Target table name rule**: the rule for adding a prefix and suffix to the converted names of destination tables.

v. Click **Next Step**.

vi. In the **Set target table** step, select a data source from the **Target MaxCompute data source** drop-down list and click the icon next to **MaxCompute time automatic partition settings**. In the **Edit** dialog box, set the partition interval of tables in MaxCompute to day or hour.

vii. Click **Reload source table and MaxCompute Table mapping** to configure the mappings between the source tables and destination MaxCompute tables.

viii. View the mapping progress, source tables, and destination tables. Then, click **Next Step**.

You can view the mapping progress between the source tables and destination tables. The mapping may take a long period of time if you want to synchronize a large number of tables.

An error message appears if the selected source table does not have a primary key. The synchronization can be performed if one of the selected source tables has a primary key. Source tables without primary keys are ignored during the synchronization.

You can set Table creation method to **Create tables automatically** or **Use existing Table**. The name of the destination table that appears in the **MaxCompute Table name** column varies based on the setting of Table creation method.

- If you set **Table creation method** to **Create tables automatically**, the name of the destination table that is automatically created appears. You can click the table name to view and modify the table creation statements.
- If you set **Table creation method** to **Use existing Table**, you must select a table name from the drop-down list in the MaxCompute Table name column.

ix. In the **Run resource settings** step, specify **Maximum number of connections supported by source read** and **Number of concurrent writes on the target side**. Then, click the icon in the top toolbar.

- To configure the real-time synchronization node for which **Sync Method** is set to **Migration to DataHub**, perform the following steps:

i. On the node configuration tab that appears, click **Basic configuration** in the right-side navigation pane. In the Basic configuration panel, select the required resource group from the **Resource Group** drop-down list.

ii. In the **Data source** section, specify **Type** and **Data source**.

iii. In the **Select the source table for synchronization** section, select the tables that you want to synchronize in the **SOURCE Table** list and click the icon to move the tables to the **Selected Source table** list.

The SOURCE Table list displays all the tables in the source. You can select all or some tables to synchronize them at a time.

> **Notice** If a selected table does not have a primary key, the table cannot be synchronized in real time.

iv. In the **Set synchronization rules** section, click **Add rule** and then select an option to configure naming rules for destination DataHub topics.

Supported options include **SOURCE table name and Topic conversion rules** and **Target Topic rules**.

v. Click **Next Step**.

vi. In the **Set target table** step, select a data source from the **Target DataHub data source** drop-down list and then click **Reload source table and DataHub Topic mapping** to configure the mappings between the source tables and destination DataHub topics.

vii. View the mapping progress, source tables, and destination topics. Then, click **Next Step**.

You can view the mapping progress between the source tables and destination topics. The mapping may take a long period of time if you want to synchronize a large number of tables.

You can set Topic creation method to **Create tables automatically** or **Use existing Topic** . The message that appears in the **DataHub Topic** column varies based on the setting of Topic creation method.

- If you set Topic creation method to **Create tables automatically**, the **Create tables automatically** dialog box appears after you click **Next Step**. Click **Start table building** in the dialog box, and then click **Close** after the topic is created.
- If you set Topic creation method to **Use existing Topic**, you must select a topic from the drop-down list in the **DataHub Topic** column.

viii. In the **Run resource settings** step, specify **Maximum number of connections supported by source read** and **Number of concurrent writes on the target side**. Then, click the icon in the top toolbar.

Commit the real-time synchronization node

1. On the configuration tab of the real-time synchronization node, click the icon in the top toolbar.
2. In the **Commit Node** dialog box, enter your comments in the **Change description** field.
3. Click **OK**.

   If the workspace that you use is in standard mode, you must click **Deploy** in the upper-right corner after you commit the real-time synchronization node.

Manage the real-time synchronization node

1. After you commit or deploy the real-time synchronization node, click **Operation Center** in the upper-right corner of the DataStudio page to manage the node on the **Real Time DI** page.
2. On the **Real Time DI** page, find the real-time synchronization node, click the node name, and then view the O&M details about the node.

   On the Real Time DI page, you can start, stop, undeploy, or configure alert settings for the real-time synchronization node.

   - To start a node that is not running, perform the following steps:
     a. Find the node and click **Start** in the Operation column.

b. In the **Start** dialog box, configure the parameters.

Parameter	Description
Whether to reset the site	Specifies whether to set the time point for the next startup. If you select Reset site, the **Start time point** and **Time zone** parameters are required.
Start time point	The date and time for starting the real-time synchronization node.
Time zone	The time zone where the source resides. Select a time zone from the **Time zone** drop-down list.
Failover	■ The condition for automatically terminating the real-time synchronization node. You can specify the maximum number of dirty data records allowed. If you set this parameter to 0, no dirty data records are allowed. If this parameter is not specified, the node continues to run no matter whether dirty data records exist. ■ You can also specify the maximum number of failover times. If you do not specify the times, the node is automatically terminated if the node fails 100 times within 5 minutes. This prevents resource occupation caused by frequent startups.

c. Click **OK**.

- To stop a running node, perform the following steps:
  a. Find the node and click **Stop** in the Operation column.
  b. In the message that appears, click **Stop**.
- To undeploy a node that is not running, perform the following steps:
  a. Find the node and click **Offline** in the Operation column.
  b. In the message that appears, click **Offline**.
- Find the node and click **Alarm settings** in the Operation column. Then, you can view alert event information and alert rules on the **Alert event** and **Alarm rules** tabs.
- To configure alert settings for a node, perform the following steps:
  a. Select the node and click **New Alarm** in the lower part of the page.

b. In the **New rule** dialog box, configure the parameters.

Parameter	Description
Name	Required. The name of the rule that you want to create.
Description	The description of the rule.
Indicators	The metrics in the rule that you want to create. Valid values: **Task Status**, **Business latency**, **Failover**, **Dirty Data**, and **DDL error**.
Threshold	The threshold for reporting an alert. The default value is 5 minutes for both **WARNING** and **CRITICAL** alerts.
Alarm interval	The interval at which an alert is reported. The default value is 5 minutes.
WARNING	The method used to send alert notifications. The value of this parameter can be only **Mail**.
CRITICAL	
Recipient	The alert recipient. Select a recipient from the **Receiver** drop-down list.

c. Click **OK**.

- To modify alert settings for a node, perform the following steps:
  a. Select the node whose alarm settings you want to modify and click **Operation alarm** in the lower part of the page.
  b. In the **Operation alarm** dialog box, specify **Operation type** and **Alarm indicators**.

     DataWorks automatically modifies all the rules for the selected alert types at a time.
  c. Click **OK**.

3.7.4. Reader

3.7.4.1. MySQL binlogs

The MySQL binlog reader reads data from the tables of MySQL databases in real time.

The MySQL binlog reader reads data from the MySQL database in real time by using Canal, which parses incremental MySQL binlogs and subscribes to data changes.

Parameters

Parameter	Description	Required	Default value
dbHost	The endpoint of the database. You can set this parameter to the domain name or the IP address.	Yes	None

Parameter	Description	Required	Default value
dbName	The name of the database.	Yes	None
port	The port number of the database.	Yes	None
table	The name of the table to be synchronized.	Yes	None
username	The username used to access the database.	Yes	None
password	The password used to access the database.	Yes	None
startTimestampMills	The start time of data synchronization. The value of this parameter is a 13-bit timestamp. **Note** If the start time you specify is later than the last data change time of MySQL binlogs, data synchronization starts from the last data change time of MySQL binlogs. If the start time you specify is earlier than the earliest data change time of MySQL binlogs, no data is synchronized.	No	None

Example

In the following code, a node is configured to read data from a MySQL database and write data to DataHub in real time:

```
{
  "order": {
    "hops": [
      {
        "from": "mysqlbinlog_01",
        "to": "datahub_01"
```

```
      }
    ]
  },
  "setting": {
    "errorLimit": {
      "record": 0
    },
  },
  "steps": [
    {
      "category": "reader",
      "name": "mysqlbinlog_01",
      "parameter": {
        "password": "xxx",
        "column": [
          "_log_file_name_offset_",
          "_operation_type_",
          "_execute_time_",
          "_before_image_",
          "_after_image_",
          "id",
          "pipeline_name",
          "execute_name",
          "context"
        ],
        "dbHost": "127.0.0.1",
        "dbName": "xxx",
        "port": 3306,
        "startTimestampMills": 1555689600000,
        "table": "xxx",
        "username": "xxx"
      },
      "stepType": "mysqlbinlog"
    },
    {
      "category": "writer",
      "name": "datahub_01",
      "parameter": {
        "accessKey": "xxx",
        "accessId": "xxx",
        "batchSize": 1000,
        "column": [
          "_log_file_name_offset_",
          "_execute_time_",
          "_before_image_",
          "_after_image_",
          "id",
          "pipeline_name",
          "execute_name",
          "context"
        ],
        "columnMapping": [
          {
            "dstColName": "_log_file_name_offset_",
```

```
            "sourceColName": "_log_file_name_offset_"
          },
          {
            "dstColName": "_execute_time_",
            "sourceColName": "_execute_time_"
          },
          {
            "dstColName": "_before_image_",
            "sourceColName": "_before_image_"
          },
          {
            "dstColName": "_after_image_",
            "sourceColName": "_after_image_"
          },
          {
            "dstColName": "id",
            "sourceColName": "id"
          },
          {
            "dstColName": "pipeline_name",
            "sourceColName": "pipeline_name"
          },
          {
            "dstColName": "execute_name",
            "sourceColName": "execute_name"
          },
          {
            "dstColName": "context",
            "sourceColName": "context"
          }
        ],
        "endpoint": "xxx",
        "project": "xxx",
        "topic": "xxx"
      },
      "stepType": "datahub"
    }
  ]
}
```

3.7.4.2. Oracle CDC

StreamX uses Change Data Capture (CDC) to synchronize data of Oracle databases in real time. This synchronization mode uses triggers on the source database to capture change data. When the CDC synchronization mode is enabled, the performance of the database may be deteriorated.

Before synchronizing data from an Oracle database, you need to perform a series of operations in the database. Take data synchronization between two Oracle tables as an example.

Synchronize data from the source table named synctest.trade to the destination table named cdcuser.trade_target.

The source and destination tables are created by using the CREATE TABLE statement. StreamX obtains changes of the incremental data and synchronizes the changed data to the destination table. The changed incremental data that StreamX reads includes the specific information about the action types. The UPDATE statements are divided into the before image and after image. The before image contains the data before the update. The after image contains the updated data.

Procedure

1. Create a table. You can also select an existing table.

```
create tablespace ts_cdcpub
logging datafile '/u01/app/oracle/oradata/mydb/cdcpub01.dbf'
size 5000M autoextend off;
```

2. Create a user named cdcuser to store incremental data and grant permissions to the user.

```
CREATE USER cdcuser IDENTIFIED BY cdcuser DEFAULT TABLESPACE ts_cdcpub QUOTA UNLIMITED
ON ts_cdcpub;
GRANT CREATE SESSION TO cdcuser;
GRANT CREATE TABLE TO cdcuser;
GRANT SELECT_CATALOG_ROLE TO cdcuser;
GRANT EXECUTE_CATALOG_ROLE TO cdcuser;
GRANT CONNECT, RESOURCE TO cdcuser;
```

3. Grant the SELECT permission on the table to be synchronized to the cdcuser user.

```
grant select on syntest.trade to cdcuser;
```

4. Create a change set and a change table.
   - Create a change set

```
DBMS_CDC_PUBLISH.CREATE_CHANGE_SET(
change_set_name => 'tradetest',
description => 'Change set for synctest.trade info',
change_source_name => 'SYNC_SOURCE');
```

   - Create a change table

```
DBMS_CDC_PUBLISH.CREATE_CHANGE_TABLE(
owner => 'cdcuser',
change_table_name => 'trade_ct',
change_set_name => 'tradetest',
source_schema => 'synctest',
source_table => 'trade',
column_type_list => 'id number,money number,op_user varchar(100)',
capture_values => 'both',
rs_id => 'y',
row_id => 'n',
user_id => 'n',
timestamp => 'n',
object_id => 'n',
source_colmap => 'y',
target_colmap => 'y',
DDL_MARKERS => 'N',
options_string => 'TABLESPACE ts_cdcpub');
```

5. After the change set and change table are created, add an Oracle CDC connection and an Oracle connection.
6. Go to the **Data Integration** page. In the left-side navigation pane, choose **Nodes > Real-Time Sync**.
   - Configure the reader. The reader currently supports four connection types including MySQL Binlog, Oracle CDC, DataHub, and LogHub. Configure the corresponding connection before configuring the reader.

Parameter	Description
Connection	The name of the connection.
Table	By default, four data records in the selected table are available for preview.
Start Offset	The time when the data synchronization node runs.
Time Zone	The time zone of the data synchronization node. The time zone can be determined based on the time zone of the connection.

   - Configure the writer. The writer currently supports three connection types including MySQL, Oracle, and DataHub. Configure the corresponding connection before configuring the writer.

     Specify and map the fields in the source and destination.

3.7.4.3. DataHub

The DataHub stream reader reads data from DataHub in real time by using the DataHub SDK.

The reader keeps running after it is started and reads data from DataHub when DataHub stores new data. The DataHub stream reader has the following two features:

- Reads data in real time.
- Reads data concurrently based on the number of shards in DataHub.

Parameters

Parameter	Description	Required
endpoint	The endpoint used to access DataHub.	Yes
accessId	The AccessKey ID used to access DataHub.	Yes
project	The destination project of DataHub. A project is the resource management unit in DataHub for resource isolation and control.	Yes

Parameter	Description	Required
topic	The destination topic of DataHub.	Yes
batchSize	The size of data read at a time.	No. Default value: 1024.
startTimestampMills	The start time of data consumption. The time is an integer accurate to milliseconds. Example: 1554739200000.	No. You must specify either the startTimestampMills parameter or the position parameter.
position	The start time of data consumption for each shard. You can specify this parameter for each shard. Example: `{"allShardTimestampMillis":{"0":1549098048000,"1":1549098048000,"2":1549098048000,"3":1549098048000,"4":1549098048000}}`.	No. You must specify either the startTimestampMills parameter or the position parameter.

Example

In the following code, a node is configured to read data from DataHub and write data to a MySQL database in real time:

```
{
  "connections": [],
  "name": "yj_datahub2datahub4",
  "order": {
    "hops": [
      {
        "from": "datahubstreamreader",
        "to": "mysqloutput001"
      }
    ]
  },
  "params": [],
  "setting": {
    "performance": {}
  },
  "steps": [
    {
      "name": "datahubstreamreader",
      "category": "reader",
      "parameter": {
        "endpoint": "http://dh-cn-hangzhou.aliyuncs.com",
        "accessId": "********",
        "accessKey": "********",
        "project": "streamx_datahub",
        "topic": "mysqlbinlog_to_datahub_topic2",
        "batchSize": 512,
```

```
        "startTimestampMills": 1554739200000,
        "position": "{\"allShardTimestampMillis\": {\"0\":1549098048000,\"1\":1549098048000
,\"2\":1549098048000,\"3\":1549098048000,\"4\":1549098048000}}",
        "column": [
          "cdc_record_id",
          "cdc_operation",
          "cdc_timestamp",
          "cdc_before_image",
          "cdc_after_image",
          "id",
          "name",
          "age"
        ]
      },
      "stepType": "datahubstream"
    },
    {
      "name": "mysqloutput001",
      "category": "writer",
      "parameter": {
        "batchSize": 100,
        "writeMode": "insert",
        "username": "********",
        "password": "********",
        "column": [
          "cdc_record_id",
          "cdc_operation",
          "cdc_timestamp",
          "cdc_before_image",
          "cdc_after_image",
          "id",
          "name",
          "age"
        ],
        "connection": [
          {
            "jdbcUrl": "jdbc:mysql://10.101.83.3:3306/test",
            "table": [
              "streamx_test_real_time_inc_datahub"
            ]
          }
        ]
      },
      "stepType": "mysql"
    }
  ],
  "type": "job",
  "version": "2.0"
}
```

3.7.4.4. LogHub

The LogHub stream reader reads data from LogHub in real time by using the LogHub SDK.

The reader reads data from LogHub topics you specified in real time and supports shard merge and split.

> **Note** When shard merge or split is enabled, duplicate data records may exist. However, data missing is avoided.

Parameters

Parameter	Description	Required	Default value
endpoint	The endpoint used to access LogHub.	Yes	None
project	The name of the LogHub project.	Yes	None
logstore	The Logstore of LogHub.	Yes	None
table	The name of the table to be synchronized.	Yes	None
accessId	The AccessKey ID used to access LogHub.	Yes	None
accessKey	The AccessKey secret used to access LogHub.	Yes	None

Parameter	Description	Required	Default value
startTimestampMills	The start time of data synchronization. The value of this parameter is a 13-bit timestamp. **Note** If the start time you specify is later than the last data change time of MySQL binlogs, data synchronization starts from the last data change time of MySQL binlogs. If the start time you specify is earlier than the earliest data change time of MySQL binlogs, no data is synchronized.	No	None

Example

In the following code, a node is configured to read data from LogHub and write data to DataHub in real time:

```
{
  "order": {
    "hops": [
      {
        "from": "loghubstream_01",
        "to": "datahub_01"
      }
    ]
  },
  "steps": [
```

```
    {
      "category": "reader",
      "name": "loghubstream_01",
      "parameter": {
        "accessKey": "<yourAccessKey>",
        "accessId": "<yourAccessId>",
        "column": [
          "col1",
          "col2",
          "col3"
        ],
        "endpoint": "<yourEndpoint>",
        "logstore": "<yourLogstore>",
        "project": "<yourProject>",
        "startTimestampMills": 1555050358000,
      },
      "stepType": "loghubstream"
    },
    {
      "category": "writer",
      "name": "datahub_01",
      "parameter": {
        "accessKey": "<yourAccessKey>",
        "batchSize": 1000,
        "column": [
          "col1",
          "col2",
          "col3"
        ],
        "columnMapping": [
          {
            "dstColName": "col1",
            "sourceColName": "col1"
          },
          {
            "dstColName": "col2",
            "sourceColName": "col2"
          },
          {
            "dstColName": "col3",
            "sourceColName": "col3"
          }
        ],
        "endpoint": "<yourEndpoint>",
        "project": "<yourProject>",
        "topic": "<yourTopic>"
      },
      "stepType": "datahub"
    }
  ]
}
```

3.7.4.5. Kafka

Configure an input node of the Kafka type

Drag **Kafka** under **Reader** to the canvas. Click the node and then configure the node in the **Node Settings** dialog box that appears.

Parameter	Description
server	The broker server address of Kafka in the format of `ip:port`.
topic	The name of the Kafka topic.
keyType	The type of the Kafka key.
valueType	The type of the Kafka value.
Consumer Offset	The consumer offset from which Kafka starts to consume data.
Parameters	The JSON-formatted parameters for specifying additional Kafka configurations. Example: group_id.
Start Offset	The date and time of the start offset.
Time Zone	The time zone.
Output Fields	The output fields of the node. You can customize the output fields.

Take the following Kafka configuration script as an example.

```
"parameter": {
       "server": "100.81.127.26:9092",
       "topic": "xin_001",
       "keyType": "long",
       "valueType": "string",
       "startupMode": "earliestOffset", -- Consume data from the earliest consumer offset.
       "discoveryIntervalMillis": 10000, -- The interval between partitions. The value is a
utomatically discovered by the system.
       "kafkaConfig": {
         "group.id":"xin_group_001"
       }
     }
```

3.7.4.6. Configure PolarDB Reader

PolarDB Reader can read data only from PolarDB for MySQL databases. It cannot read data from PolarDB for PostgreSQL databases.

Procedure

1. Log on to the DataWorks console.

2. Move the pointer over the +Create icon and choose **Data Integration > Real-time synchronization**.

   Alternatively, you can click the required workflow, right-click **Data Integration**, and then choose **Create > Real-time synchronization**.

3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Notice** The node name must be 1 to 128 characters in length. It can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.
5. On the configuration tab of the real-time synchronization node, drag **PolarDB** in the **Input** section to the canvas on the right.
6. Click the **PolarDB** node. In the panel that appears, configure the parameters.

Parameter	Description
Data source	The PolarDB for MySQL data source that you have added. You can select only a PolarDB for MySQL data source. If no data source is available, click **New data source** to add one on the **Data Source** page.
Table	The name of the table from which you want to read data. You can click **Data preview** on the right to preview the selected table.
Output field	The fields from which you want to read data.

7. Click the icon in the toolbar.

3.7.5. Writer

3.7.5.1. Configure MaxCompute Writer

MaxCompute offers a comprehensive data import solution to support fast computing for large amounts of data.

Prerequisites

A reader or conversion node is configured.

Procedure

1. Log on to the DataWorks console.
2. Move the pointer over the +Create icon and choose **Data Integration > Real-time synchronization**.

   Alternatively, you can click the required workflow, right-click **Data Integration**, and then choose **Create > Real-time synchronization**.

3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

> **Notice** The node name must be 1 to 128 characters in length. It can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.
5. On the configuration tab of the real-time synchronization node, drag **MaxCompute** in the **Output** section to the canvas on the right. Connect the MaxCompute node to the configured reader or conversion node.
6. Click the **MaxCompute** node. In the panel that appears, configure the parameters.

Parameter	Description
Data source	The MaxCompute data source that you have configured. You can select only a MaxCompute data source. If no data source is available, click **New data source** on the right to add one on the **Data Source** page.
Table	The name of the MaxCompute table to which you want to write data. You can click **One-Click table creation** on the right to create a table, or click **Data preview** to preview the selected table. **Notice** Before you create a table, connect the MaxCompute node to a reader node, and make sure that the output field parameters are specified for the reader node.
Partition message	The information about the partitioned MaxCompute table.
Field Mapping	The field mappings between the source and destination. Click **Field Mapping** and configure field mappings. The synchronization node synchronizes data based on the field mappings.

If you want to create a table, click **One-Click table creation** next to Table. In the **New data table** dialog box, configure the parameters.

Parameter	Description
Table name	The name of the MaxCompute table.
Life cycle	The lifecycle of the MaxCompute table.
Data field structure	The field structure of the MaxCompute table. To add a field, click **Add**.
Partition settings	The partitions of the MaxCompute table. **Notice** You must configure at least two levels of partitions, which are yearly and monthly partitions. You can configure a maximum of five levels of partitions, which are yearly, monthly, daily, hourly, and minutely partitions.

7. Click the icon in the toolbar.

3.7.5.2. Configure Hologres Writer

You can build a real-time data warehouse by using the real-time write capability of Hologres.

Prerequisites

A reader or conversion node is configured.

Procedure

1. Log on to the DataWorks console.
2. Move the pointer over the +Create icon and choose **Data Integration > Real-time synchronization**.

   Alternatively, you can click the required workflow, right-click **Data Integration**, and then choose **Create > Real-time synchronization**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Notice** The node name must be 1 to 128 characters in length. It can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.
5. On the configuration tab of the real-time synchronization node, drag **Hologres** in the **Output** section to the canvas on the right. Then, draw a line to connect it to the configured reader or conversion node.
6. Click the **Hologres** node. In the panel that appears, configure the parameters.

Parameter	Description
Data source	The Hologres data source that you configured. You can select only a Hologres data source. If no data source is available, click **New data source** to add one on the **Data Source** page.
Table	The name of the Hologres table to which you want to write data. You can click **One-Click table creation** on the right to create a table, or click **Data preview** to preview the selected table.

Parameter	Description
Dynamic Time Partition	If the Hologres table is a partitioned table, you must specify a dynamic time-based partition. The dynamic time-based partition parses the value of a source field in the yyyymmdddhhmmss format. After the value is parsed, you can use the dynamic partition whose name is a string of variables in the destination table. The destination partition varies based on the value of the source field. For example, the value of the source field is 20200816, and the name of the destination partition is in the {yyyy}-{mm}-{dd} format. In this case, the value is written to the 2020-08-16 partition.
Job type	The type of the data write operation. Valid values: **Replay (replay operation log to restore data)** and **Insert (direct archive save)**. ○ **Replay (replay operation log to restore data)**: indicates that Hologres Writer performs the same operation on the Hologres destination as that performed on the source. For example, if the `INSERT` statement is executed to add a record to the source, Hologres Writer executes the `INSERT` statement to add the same record to the Hologres destination. If the `UPDATE` or `DELETE` statement is executed in the source, Hologres Writer executes the `UPDATE` or `DELETE` statement in the Hologres destination. ○ **Insert (direct archive save)**: indicates that Hologres Writer uses the Hologres destination as streaming data storage. Data is synchronized from the source to the Hologres destination by using the `INSERT` statement.
Writer conflict policy	The solution to data write conflicts. Valid values: ○ **Cover (Overwrite)**: indicates that Hologres Writer uses the new data synchronized from the source to overwrite the existing data in the Hologres destination. ○ **Ignore (Ignore)**: indicates that Hologres Writer ignores the new data synchronized from the source and retains the existing data in the Hologres destination.
Field Mapping	The field mappings between the source and destination. Click **Field Mapping** and configure field mappings between the source and destination. The synchronization node synchronizes data based on the field mappings.

7. Click the icon in the toolbar.

3.7.5.3. DataHub

Currently, the writer supports the DataHub connection. You need to configure the corresponding connection before configuring the writer.

DataHub is a platform designed to process streaming data. You can publish and subscribe applications to streaming data in DataHub and distribute the data to other platforms. This allows you to easily analyze streaming data and build applications based on the streaming data.

DataHub Writer writes data to DataHub by using the DataHub SDK for Java. The SDK version is as follows:

```
<dependency>
    <groupId>com.aliyun.datahub</groupId>
    <artifactId>aliyun-sdk-datahub</artifactId>
    <version>2.5.1</version>
</dependency>
```

Parameters

Parameter	Description	Required
endpoint	The endpoint used to access DataHub.	Yes
accessId	The AccessKey ID used to access DataHub.	Yes
accessKey	The AccessKey secret used to access DataHub.	Yes
project	The destination project of DataHub. A project is the resource management unit in DataHub for resource isolation and control.	Yes
topic	The destination topic of DataHub.	Yes
maxCommitSize	The size of the data written into DataHub at one time. Unit: Bytes.	No. Default value: 1,048,576 Bytes (1 MB).
maxRetryCount	The maximum number of operation retries in DataHub.	No. Default value: 500.
column	The output columns. Currently, you need to specify all columns in the destination DataHub data store as the output columns.	Yes
columnMapping	The mapping between the columns in the source and destination.	No

Example

```
{
  "connections": [],
```

```
  "name": "yj_datah***hub4",
  "order": {
    "hops": [
      {
        "from": "datahub***r86m8vT24",
        "to": "datahub_cM***aNAHUly"
      }
    ]
  },
  "params": [],
  "setting": {
    "performance": {}
  },
  "steps": [
    {
      "category": "reader",
      "name": "datahu***6m8vT24",
      "parameter": {
        "accessKey": "<yourAccessKey>",
        "accessId": "<yourAccessId",
        "column": [
          "string_col",
          "bigint_col",
          "double_col",
          "timestamp_col",
          "bool_col"
        ],
        "endpoint": "http://dh-cn-hangzhou.aliyuncs.com",
        "name": "<yourDatahubName>",
        "project": "streamx_datahub",
        "startTimestampMills": 1554739200000,
        "topic": "datahub_test_topic1"
      },
      "stepType": "datahubstream"
    },
    {
      "category": "writer",
      "name": "datahub_cM5XnifsIaNAHUly",
      "parameter": {
        "accessKey": "********",
        "accessId": "********",
        "batchSize": 512,
        "column": [
          "string_col",
          "bigint_col",
          "double_col",
          "timestamp_col",
          "bool_col"
        ],
        "columnMapping": [
          {
            "dstColName": "string_col",
            "sourceColName": "string_col"
          },
          {
```

```
            {
              "dstColName": "bigint_col",
              "sourceColName": "bigint_col"
            },
            {
              "dstColName": "double_col",
              "sourceColName": "double_col"
            },
            {
              "dstColName": "timestamp_col",
              "sourceColName": "timestamp_col"
            },
            {
              "dstColName": "bool_col",
              "sourceColName": "bool_col"
            }
          ],
          "endpoint": "http://dh-cn-hangzhou.aliyuncs.com",
          "name": "yj_datahub_datasource",
          "project": "streamx_datahub",
          "topic": "datahub_test_topic2"
        },
        "stepType": "datahub"
      }
    ]
}
```

3.7.5.4. Kafka

Create a real-time sync node

1. Log on to the DataWorks console.
2. Move the pointer over the DataWorks icon in the upper-left corner and select **Data Integration**.
3. In the left-side navigation pane, choose **Nodes > Real-Time Sync**. Click **Create Task** in the upper-right corner.
4. In the **Create Node** dialog box that appears, set **Node Name** and **Description**.
5. Click **OK**.

Configure an output node of the Kafka type

Drag **Kafka** under **Writer** to the canvas. Click the node and then configure the node in the **Node Settings** dialog box that appears.

Parameter	Description
server	The broker server address of Kafka in the format of `ip:port`.
topic	The name of the Kafka topic.
keyColumn	The column for storing the key.

Parameter	Description
keyType	The type of the Kafka key.
valueColumn	The column for storing the value. If this parameter is not specified, all columns are concatenated by using the delimiter specified by fieldDelimiter to form the value.
valueType	The type of the Kafka value.
batchSize	The amount of data written at a time. Default value: 1024.
Parameters	The extended parameters specified when KafkaConsumer is created, such as bootstrap.servers, auto.commit.interval.ms, and session.timeout.ms. By setting parameters in kafkaConfig, you can control the data consumption behaviors of KafkaConsumer.

Take the following Kafka configuration script as an example.

```
"parameter": {
       "server": "100.81.127.26:9092",
       "keyIndex": 0,
       "valueIndex": 1,
       "keyType": "long",
       "valueType": "string",
       "topic": "xin_002",
       "batchSize": 1,
     }
```

3.7.6. Transform

3.7.6.1. Data filter

The data filter plug-in is used to filter data.

Parameters

Parameter	Description	Required
category	The category of the plug-in. Set this parameter to filter for the data filter plug-in.	Yes
stepType	The type of the step. Set this parameter to filterrows for the data filter plug-in.	Yes
name	The name of the step.	Yes

Parameter	Description	Required
condition	The filter condition in the expression format. The expression can contain the columns of the reader. In this example, the id and age columns in the filter condition are two columns specified in the reader. An expression that does not contain any column names can also be valid, such as 1==1.	Yes

Example

```
{
  "connections": [],
  "name": "yj_mysql_2_mysql",
  "order": {
    "hops": [
      {
        "from": "J3Bji***NPLn",
        "to": "rYCN***VxmlK0"
      },
      {
        "from": "rYCNF***mlK0",
        "to": "nD***JWPh0YJl0X"
      }
    ]
  },
  "params": [],
  "setting": {
    "keyVersion": "201412091312",
    "metric": {
      "transMetric": {
        ...
      }
    },
  },
  "steps": [
    {
      "category": "reader",
      "name": "J3BjiG***QNPLn",
      "parameter": {
        "password": "<yourPassWord>",
        "access": "Native",
        "column": [
          "id",
          "name",
          "age"
        ],
        "dbHost": "10.101.83.3",
        "dbName": "test",
```

```
        "port": 3306,
        "startTimestampMills": 1551801600000,
        "table": "streamx_test_real_time",
        "type": "MYSQLBINLOG",
        "username": "<yourUserName>"
      },
      "stepType": "mysqlbinlog"
    },
    {
      "category": "filter",
      "name": "rYCNFINr38VxmlK0",
      "parameter": {
        "condition": " (id > 10) &&(age > 30)",
      },
      "stepType": "filterrows"
    },
    ... // The writer configuration is omitted here.
  ]
}
```

3.7.6.2. String replacement

The string replacement plug-in is used to replace the field values of the String type.

Parameter	Description
Field	The input fields read in the previous step.
Regular Expression Match	Specifies whether to perform regular expression match on the fields. For more information about regular expressions, see the relevant Java documentation.
Original String	The string to be replaced.
New String	The string used to replace the original string.
Case Sensitive	Specifies whether the query of the original string is case-sensitive.

3.8. Data synchronization solutions

3.8.1. Go to the Sync Solutions page

The Data Integration service of DataWorks allows you to create and configure a synchronization solution to synchronize data from a source to a destination in real time. You can use a synchronization solution to synchronize multiple tables at a time or synchronize both full and incremental data. If you want to synchronize both full and incremental data, you can synchronize the incremental data after the full data is synchronized.

Procedure

1. Log on to the DataWorks console.
2. On the DataStudio page, click the ☰ icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
3. In the left-side navigation pane, click **Sync Solutions**. The **Solution Task list** page appears.

   You can create synchronization solutions and view the details and statuses of the created synchronization solutions on this page. A synchronization solution has the following states:

   - **Not running**: The synchronization solution is not run. You can click **Start execution** in the Operation column that corresponds to the synchronization solution to run the synchronization solution.

     > **Note** You can click **Task configuration** in the Operation column that corresponds to a synchronization solution in the **Not running** state to edit the synchronization solution. If you click Task configuration in the Operation column that corresponds to a synchronization solution in another state, you can only view the information about the synchronization solution.

   - **Running**: The synchronization solution is running and cannot be terminated. You must wait until the synchronization solution is completed.
   - **Exception**: An error occurred during the running of the synchronization solution. You can click **Execution details** in the Operation column that corresponds to the synchronization solution to troubleshoot the error.
   - **Success**: The synchronization solution is completed. You can click **Execution details** in the Operation column that corresponds to the synchronization solution to view the running results of the synchronization solution.

3.8.2. Synchronize data to Hologres in real time

You can create and configure a data synchronization solution to synchronize data in a specified data source to Hologres in real time. This topic describes how to synchronize data to Hologres in real time.

Procedure

1. Go to the **Solution Task list** page.
   i. Log on to the DataWorks console.
   ii. Click the ☰ icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. In the left-side navigation pane, click **Sync Solutions**.
2. On the **Solution Task list** page, click **New Task** in the upper-right corner.
3. In the **New resolution task** dialog box, click **One-click real-time synchronization to Hologres**.
4. In the **Set synchronization sources and rules** step, configure the parameters.

i. In the **Basic configuration** section, configure the relevant parameters.

Parameter	Description
Scheme name	The name of the synchronization solution. The name can be a maximum of 50 characters in length.
Description	The description of the synchronization solution. The description can be a maximum of 50 characters in length.
Destination task storage location	If you select Automatically establish workflow, DataWorks automatically creates a workflow named in the format of clone_database_Source name+to+Destination name in the **Data Integration** directory. All synchronization nodes generated by the synchronization solution are placed in the directory of this workflow. If you do not select **Automatically establish workflow**, you must select a directory from the **Select Location** drop-down list. All synchronization nodes generated by the synchronization solution are placed in the specified directory.

ii. In the **Data source** section, specify **Type** and **Data source**.

iii. In the **Select the source table for synchronization** section, select the tables that you want to synchronize in the **SOURCE Table** list and click **>** to move the tables to the **Selected Source table** list.

The SOURCE Table list displays all the tables in the source. You can select all or some tables to synchronize them at a time.

> **Notice** If a selected table does not have a primary key, the table cannot be synchronized in real time.

iv. In the **Set synchronization rules** section, click **Add rule** and select an option to configure naming rules for destination tables.

Supported options include **Table name conversion rules** and **Target table name rule**.

- **Table name conversion rules**: the rule for converting the names of source tables to those of destination tables.
- **Target table name rule**: the rule for adding a prefix and suffix to the converted names of destination tables.

v. Click **Next Step**.

5. In the **Set target table** step, configure the parameters.

i. Specify **Target Hologres data source** and **Schema**. **Write Hologres policy** is set to **Replay (replay operation log to restore data)** by default and cannot be changed.

ii. Click **Reload source table and Hologres Table mapping** to configure the mappings between the source tables and destination Hologres tables.

iii. View the mapping progress, source tables, and mapped destination tables.

You can view the mapping progress between the source tables and destination tables. The mapping may take a long period of time if you want to synchronize a large number of tables.

An error message appears if the selected source table does not have a primary key. The synchronization can be performed if one of the selected source tables has a primary key. Source tables without primary keys are ignored during the synchronization.

You can set Table creation method to **Create tables automatically** or **Use existing Table**. The name of the destination table that appears in the **Hologres Table name** column varies based on the setting of Table creation method.

- If you set **Table creation method** to **Create tables automatically**, the name of the destination table that is automatically created appears. You can click the table name to view and modify the table creation statements.
- If you set **Table creation method** to **Use existing Table**, you must select a table from the drop-down list in the Hologres Table name column.

iv. Click **Next Step**.

6. In the **Run resource settings** step, configure the parameters.

Parameter	Description
Realtime task resource group	The resource group used for running the batch synchronization node and real-time synchronization nodes generated by the synchronization solution.
Offline task resource group	
Select scheduling Resource Group	The resource group for scheduling used for running the nodes generated by the synchronization solution.
Maximum number of connections supported by source read	The maximum number of Java Database Connectivity (JDBC) connections that are allowed for the source. Specify an appropriate number based on the resources of the source.
Offline task name rules	The name of the batch synchronization node that is used to synchronize the full data of the source. After a synchronization solution is created, a batch synchronization node is generated first to synchronize full data, and then real-time synchronization nodes are generated to synchronize incremental data.

7. Click **Complete configuration**.
8. On the **Solution Task list** page, find the newly created synchronization solution and click **Start execution** in the Operation column.

   After the running of the synchronization solution is successful, you can perform the following operations on the synchronization solution:

   - Click **Task configuration** in the Operation column to view the information about or edit the synchronization solution.

> **Note** You can click **Task configuration** in the Operation column that corresponds to the data synchronization solution in the **Not running** state to edit the data synchronization solution. If you click Task configuration in the Operation column of a synchronization solution in another state, you can only view information about the synchronization solution.

- Click **Execution details** in the Operation column to view the points in time at which the synchronization solution was started and ended and the status of each node.
- Click **Delete** in the Operation column to delete the synchronization solution. In the **Delete** message, click **Confirm**.

> **Note** After you click Confirm, only the configuration record of the data synchronization solution is deleted. The generated synchronization nodes and tables are not affected.

3.8.3. Synchronize data to MaxCompute in real time

You can create a synchronization solution to synchronize data from a specified data source to MaxCompute in real time.

Procedure

1. Go to the **Solution Task list** page.
   i. Log on to the DataWorks console.
   ii. Click the ☰ icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. In the left-side navigation pane, click **Sync Solutions**.
2. On the **Solution Task list** page, click **New Task** in the upper-right corner.
3. In the **New resolution task** dialog box, click **One-click real-time synchronization to MaxCompute**.
4. In the **Set synchronization sources and rules** step, configure the parameters.

i. In the **Basic configuration** section, configure the relevant parameters.

Parameter	Description
Scheme name	The name of the synchronization solution. The name can be a maximum of 50 characters in length.
Description	The description of the synchronization solution. The description can be a maximum of 50 characters in length.
Destination task storage location	If you select Automatically establish workflow, DataWorks automatically creates a workflow named in the format of clone_database_Source name+to+Destination name in the **Data Integration** directory. All synchronization nodes generated by the synchronization solution are placed in the directory of this workflow. If you do not select **Automatically establish workflow**, you must select a directory from the **Select Location** drop-down list. All synchronization nodes generated by the synchronization solution are placed in the specified directory.

ii. In the **Data source** section, specify **Type** and **Data source**.

iii. In the **Select the source table for synchronization** section, select the tables that you want to synchronize in the **SOURCE Table** list and click **>** to move the tables to the **Selected Source table** list.

The SOURCE Table list displays all the tables in the source. You can select all or some tables to synchronize them at a time.

> **Notice** If a selected table does not have a primary key, the table cannot be synchronized in real time.

iv. In the **Set synchronization rules** section, click **Add rule** and select an option to configure naming rules for destination tables.

Supported options include **Table name conversion rules** and **Target table name rule**.

- **Table name conversion rules**: the rule for converting the names of source tables to those of destination tables.
- **Target table name rule**: the rule for adding a prefix and suffix to the converted names of destination tables.

v. Click **Next Step**.

5. In the **Set target table** step, configure the parameters.

i. Select a data source from the **Target MaxCompute data source** drop-down list and specify **Write mode**.

ii. Click next to **MaxCompute time automatic partition settings**. In the **Edit** dialog box, modify the partition settings for the destination tables. You can configure daily and hourly partitions.

iii. Click **Reload source table and MaxCompute Table mapping** to configure the mappings between the source tables and destination MaxCompute tables.

iv. View the mapping progress, source tables, and mapped destination tables.

You can view the mapping progress between the source tables and destination tables. The mapping may take a long period of time if you want to synchronize a large number of tables.

An error message appears if the selected source table does not have a primary key. The synchronization can be performed if one of the selected source tables has a primary key. Source tables without primary keys are ignored during the synchronization.

You can set Table creation method to **Create tables automatically** or **Use existing Table**. The name of the destination table that appears in the MaxCompute Table name column varies based on the setting of Table creation method.

- If you set **Table creation method** to **Create tables automatically**, the name of the destination table that is automatically created appears. You can click the table name to view and modify the table creation statements.
- If you set **Table creation method** to **Use existing Table**, you must select a table name from the drop-down list in the MaxCompute Table name column.

v. Click **Next Step**.

6. In the **Run resource settings** step, configure the parameters.

Parameter	Description
Synchronization engine	Default value: **Default embedded engine**.
Realtime task resource group	The resource group that is used to run the nodes generated by the real-time synchronization solution.
Real-time synchronization task name	The name of the real-time synchronization solution.
Select scheduling Resource Group	The resource group that is used to run the real-time synchronization nodes and batch synchronization node generated by the synchronization solution. Synchronization solutions can run on shared resource groups and custom resource groups for Data Integration.
Offline task resource group	
Maximum number of connections supported by source read	The maximum number of Java Database Connectivity (JDBC) connections that are allowed for the source database. Specify an appropriate number based on the resources of the source database.
Offline task name rules	The name of the batch synchronization node that is used to synchronize the full data of the source database. After a data synchronization solution is created, DataWorks first generates a batch synchronization node to synchronize full data, and then generates real-time synchronization nodes to synchronize incremental data.

7. Click **Complete configuration**.
8. On the **Solution Task list** page, find the newly created synchronization solution and click **Start execution** in the Operation column.

After the synchronization solution is run, you can perform the following operations on the synchronization solution:

- Click **Task configuration** in the Operation column to view information about or configure the synchronization solution.

  > **Note** You can configure a synchronization solution only when it is in the **Not running** state. If you click Task configuration in the Operation column of a synchronization solution that is in another state, you can only view information about the synchronization solution.

- Click **Execution details** in the Operation column to view the time at which the synchronization solution was started and ended and the status of each node.
- Click **Delete** in the Operation column to delete the synchronization solution. In the **Delete** message, click **Confirm**.

  > **Note** After you click Confirm, only the configuration record of the data synchronization solution is deleted. The generated synchronization nodes and tables are not affected.

3.9. Resource groups

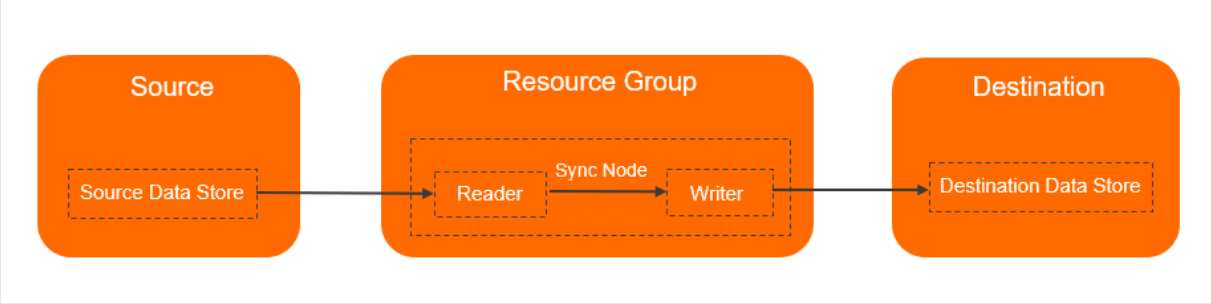
3.9.1. Overview

This topic introduces the definition of resource groups. It also describes how the connectivity and performance of resource groups affect data synchronization.

Definition

A resource group is a collection of computing resources on which synchronization nodes of Data Integration are run. In most cases, a resource group is one or more servers that consist of CPU, memory, and network resources.

In the process of running a synchronization node, the resource group pulls data from the source and pushes the data to the destination.



Connectivity and performance

When you use resource groups, you must pay attention to their connectivity and performance.

- Connectivity

  To ensure that data can be synchronized, a resource group must be connected to the source and destination. Connectivity is the most important factor that affects data synchronization.

Data Integration cannot build networks. Before you use Data Integration to synchronize data, you must make sure that the resource group is connected to the data sources. If the resource group is disconnected from the data sources, synchronization nodes cannot be run.

- Performance

  Synchronization nodes consume the CPU, memory, and network resources on the servers where the nodes are run. Insufficient resources may lead to various issues. For example, the nodes fail to be started, wait for resources for a long period of time after startup, transmit data at a low rate, or fail to generate data. To ensure the smooth running of synchronization nodes, you must allocate adequate resources for them.

3.9.2. Shared resource groups

Data Integration of DataWorks provides shared resource groups for you to create and run synchronization nodes.

Shared resource groups for Data Integration are created and maintained by Data Integration. Shared resource groups compose a public resource pool. Nodes that use resources in the public resource pool may not be run as scheduled due to insufficient resources. We recommend that you prepare sufficient resources to ensure the efficient running of synchronization nodes.

> Note
>
> - You can run a maximum of 25 parallel nodes on a shared resource group for Data Integration when the shared resource group is not in use.
> - You cannot change the memory size of a shared resource group. Instead, you can change the number of parallel nodes that can be run on the shared resource group.
>
>   The following formula is used to calculate the memory size: Memory size = `Number of parallel nodes × 512 MB`.

3.9.3. Create a custom resource group for Data Integration

This topic describes how to create a custom resource group for Data Integration and select a resource group for Data Integration to run a batch synchronization node.

Prerequisites

An Elastic Compute Service (ECS) instance is available.

Context

If the shared resource groups of DataWorks do not support your data sources or you want to speed up data transmission, you can create custom resource groups to run your synchronization nodes.

A workspace administrator can create or modify custom resource groups on the **Custom Resource Groups** page of Data Integration.

> Note

- The admin permission is required to access some files on the ECS instance that hosts a custom resource group. For example, the admin permission is required to call shell or Structured Query Language (SQL) files on the ECS instance when you write a shell script for a node.
- Resource groups for scheduling are used to run nodes. These resource groups have limited resources and are not suitable for computing nodes. Therefore, we recommend that you do not create custom resource groups on the ECS instances of a resource group for scheduling. MaxCompute can process large amounts of data. We recommend that you use MaxCompute for big data computing.

Custom resource groups for Data Integration are subject to the following limits:

- The difference between the time of the ECS instance where a custom resource group for Data Integration resides and the current Internet time must be within 2 minutes. Otherwise, service requests may time out, and nodes may fail to be run on the custom resource group for Data Integration.
- You can add only one custom resource group for Data Integration on an ECS instance. You can select only one network type for each custom resource group for Data Integration.
- Custom resource groups added on the **Custom Resource Groups** page of Data Integration can run synchronization nodes created only in the current workspace.

  Custom resource groups for Data Integration that you added on the Custom Resource Groups page cannot run synchronization nodes in a manually triggered workflow.

If the timeout error message `response code is not 200` exists in the log file of alisatasknode, the custom resource group for Data Integration was not accessible within the specific period in time. The ECS instance that hosts the custom resource group for Data Integration can continue to work if the exception persists for no more than 10 minutes. To find the exception details, view the heartbeat.log file in the `/home/admin/alisatasknode/logs` directory.

Create a custom resource group for Data Integration

1. Go to the **Data Integration** page.
   i. Log on to the DataWorks console.
   ii. Click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
2. In the left-side navigation pane, click **Custom Resource Group**.
3. On the **Custom Resource Groups** page, click **Add Resource Group** in the upper-right corner.

   > **Notice** By default, the Custom Resource Groups page displays only your custom resource groups and does not display your shared resource groups.

4. In the **Add Resource Group** wizard, perform the following steps:
   i. In the **Create Resource Group** step, set the **Resource Group Name** parameter.

   > **Note** The name can contain letters, digits, and underscores (_) and must start with a letter.

   ii. Click **Next**.

iii. In the **Add Server** step, set the parameters.

Parameter	Description
Network Type	The network type. Valid values: **Classic Network** and **VPC**.
Server Name or **ECS UUID**	The hostname or the universally unique identifier (UUID) of the ECS instance that hosts the custom resource group. ■ If you set Network Type to **Classic Network**, you must set the **Server Name** parameter. To obtain the hostname, log on to the ECS instance and run the `hostname` command. ■ If you set Network Type to **VPC**, you must set the **ECS UUID** parameter. To obtain the UUID, log on to the ECS instance and run the `dmidecode | grep UUID` command.
Server IP Address	The private IP address of the ECS instance.
Server CPU (Cores)	The number of CPU cores on the ECS instance. We recommend that you configure at least four CPU cores for an ECS instance that hosts a custom resource group.
Server RAM (GB)	The memory of the ECS instance. We recommend that you configure at least 8 GB RAM and 80 GB disk space for an ECS instance that hosts a custom resource group.

iv. Click **Next**.

v. Perform the steps that are listed in the **Install Agent** step.

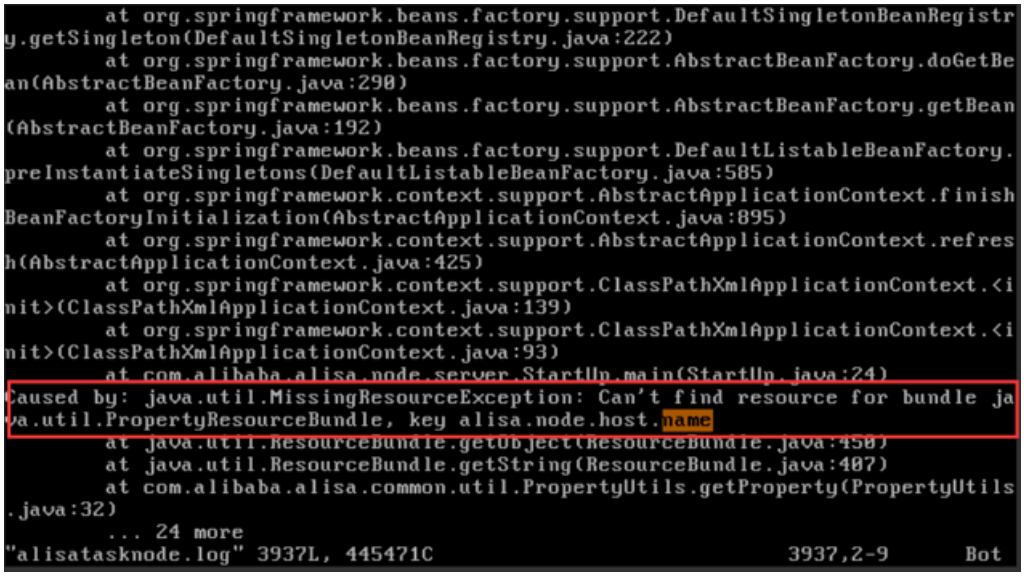
> **Note** If an error occurs when you run the `install.sh` script or you need to run it again, run the `rm -rf install.sh` command in the same directory as the `install.sh` script to delete the generated file. Then, run the `install.sh` script again.
>
> The commands to run during the installation and initialization process differ for each user. Run relevant commands based on the instructions on the initialization interface.

vi. Click **Next**.

vii. In the **Test Connection** step, click **Refresh** and check the status of the instance.

viii. Click **Complete**.

If the instance status remains **Stopped** after the preceding steps, the hostname may not be bound to an IP address, as shown in the following figure.



1. Log on to the ECS instance by using the admin user.
2. Run the `hostname -i` command to view the hostname binding information.
3. Run the `vim/etc/hosts` command to add the binding of the IP address and hostname.
4. Refresh the instance status and check whether the ECS instance is registered.

   If the ECS instance is still in the Stopped state after you refresh the page, perform the following steps to restart alisatasknode:

   i. Log on to the ECS instance by using the admin user.

   ii. Run the following command:

   ```
   /home/admin/alisatasknode/target/alisatasknode/bin/serverctl restart
   ```

   > **Note** You must enter your AccessKey pair when you run this command. Keep your AccessKey secret strictly confidential.

Configure the resource group for Data Integration

1. Click the icon in the upper-left corner of the Data Integration page and choose **All Products > Data Development > DataStudio**.
2. In the upper-left corner of the page that appears, select the workspace where your resource group for Data Integration resides.
3. On the **Data Analytics** tab, expand the workflow where the batch synchronization node you want to configure resides, find the batch synchronization node in the Data Integration folder, and then double-click it.
4. On the configuration tab of the node, click the **Resource Group configuration** tab in the right-side navigation pane.
5. On the **Resource Group configuration** tab, set **Programme** and select a resource group based on your business requirements.

6. On the configuration tab of the node, click the icon in the top toolbar.

3.10. Full-database migration

3.10.1. Overview

This section describes the full-database migration feature in terms of its functions and limits.

Full-database migration is an easy-to-use tool that helps you to improve cost-efficiency. It can quickly upload all the tables in a MySQL database to MaxCompute at a time, saving time that is spent on creating batch tasks for initial data migration to the cloud.

For example, if a database contains 100 tables, you must configure 100 data synchronization tasks in a traditional way. With the full-database migration, you can upload all the tables at a time. However, an upload failure might occur due to the issues that involve the principles of designing database tables.

Task generation rules

After the configuration is completed, MaxCompute tables are created and data synchronization tasks are generated based on the selected tables to be synchronized.

The table names, field names, and field types of the MaxCompute tables are generated according to the advanced settings. If no advanced settings are configured, the structure of MaxCompute tables is identical to that of MySQL tables. The partition of these tables is pt, and its format is yyyymmdd.

The generated data synchronization tasks are daily scheduled tasks and run automatically on the early morning of the next day. The typical transmission rate is 1 Mbit/s, but it varies depending on the synchronization method and concurrency configurations. **To customize a data synchronization task, locate the task by choosing clone_database > Data Source Name > mysql2odps_table name,** and then specify its settings.

> **Note** We recommend that you perform smoke testing on a data synchronization task on the day when it is generated.**To perform smoke testing, choose Administration Center > Task Management > project_etl_start > Upload Database > Data Source Name**, find the synchronization task, right-click the task, and then test the task.

Limits

Full-database migration has the following limits due to the issues that involve the principles of designing database tables.

- Currently, only the full-database migration from a MySQL data source to MaxCompute is supported. We are working on support for full-database migration from a Hadoop or Hive data source to Oracle.
- Only the daily incremental and daily full upload modes are available.

  If you want to synchronize historical data at a time, this feature cannot meet your needs. We recommend that:

  - You configure daily tasks instead of synchronizing historical data at a time. You trace the historical data with the provided retrospective data import feature. This eliminates the need to run temporary SQL tasks to split data after all the historical data is synchronized.
  - To synchronize historical data at a time, configure a task on the task development page and click **Run**. Then, data is converted by using SQL statements. They are both one-time operations.

If your daily incremental upload task uses a special business logic and cannot be identified by a date field, this feature cannot meet your needs. We provide the following suggestions:

- The incremental data upload can be achieved by using two methods: binlog provided by the DTS product and the date field for data changes provided by databases.

  Currently, Data Integration supports the second method. Therefore, your database must contain the date field for data changes. The system determines whether your data is changed on the same day as the business date by using this field. If yes, all the changed data is synchronized.

- To facilitate the incremental data uploading, we recommend that you include the gmt_create and gmt_modify fields when creating any database tables. Additionally, you can set the id field as the primary key to improve efficiency.

- Full-database migration supports batch upload and full upload modes.

  Batch upload is configured with time intervals. Currently, the connection pool protection feature for data sources is not supported, but will be available later.

  - To prevent overloads on the database, the full-database migration feature provides the batch upload mode. This mode enables you to upload tables in batches at a specified time interval and prevents compromised service functionality. We provide the following suggestions:
    - If you have master and slave databases, we recommend that you synchronize the data of the slave database.
    - In a batch upload task, each table has a database connection with a maximum transmission rate of 1 Mbit/s. For example, if you run a synchronization task for 100 tables at a time, 100 database connections are established. We recommend that you specify proper concurrency settings based on your business needs.
  - If you have special requirements for transmission efficiency, this feature cannot meet your needs. The maximum transmission of each generated tasks is 1 Mbit/s.

- Only the mapping of all table names, field names, and field types are supported.

  During the full-database migration process, MaxCompute tables are created automatically, where the partition field is pt, the field type is string, and the format is yyyymmdd.

  > **Note** When you select tables for synchronization, all fields must be synchronized and none of these fields can be edited.

3.10.2. Migrate a MySQL database

This topic describes how to migrate a MySQL database to MaxCompute.

The database migration feature improves efficiency and reduces costs. It can quickly upload all tables in a MySQL database to MaxCompute. For more information, see Overview.

Procedure

1. Log on to the DataWorks console.
2. Click the DataWorks icon in the upper-left corner and choose **All Products > Data Integration** to go to the **Data Integration** page.
3. In the left-side navigation pane, choose **Sync Resources > Connections**. On the page that appears, click **Add Connection**.

4. In the **Add Connection** dialog box that appears, select **MySQL**.
5. Add a MySQL connection named clone_database for database migration.
6. Click **Test Connection** and verify that the database can be accessed. Click **Complete**.
7. The added MySQL connection named clone_database appears in the connection list. Find the added connection and click **Migrate Database** in the Actions column.

   The database migration settings page consists of three functional modules.

Functional module	Description
Tables to migrate	This module lists all the tables in the MySQL connection named clone_database. Selected tables will be migrated.
Advanced Settings	You can configure the rules for converting the table name, column names, and data types.
Basic settings	You can select whether to synchronize full or incremental data, whether to upload data in one or more batches, and the synchronization efficiency. You can also view the migration progress and results.

8. Click **Advanced Settings** and configure conversion rules based on your needs. For example, you can add an ods_ prefix to the name of each MaxCompute table.
9. Specify basic settings. Set Sync Method to Synchronize Incremental Data Daily, and configure the incremental data to be determined based on the gmt_modified column. Data Integration will generate WHERE clauses based on the specified column and DataWorks scheduling parameters such as ${bdp.system.bizdate}.

   Data Integration reads data from MySQL tables by connecting to a remote MySQL database over JDBC and running SELECT statements. Data Integration uses standard SQL statements, and therefore you can configure WHERE clauses to filter data. The WHERE clause used in this example is provided as follows:

   ```
   STR_TO_DATE('${bdp.system.bizdate}', '%Y%m%d') <= gmt_modified AND gmt_modified < DATE_
   ADD(STR_TO_DATE('${bdp.system.bizdate}', '%Y%m%d'), interval 1 day)
   ```

   Select data upload in batches to protect the MySQL database from being overloaded. Let Data Integration start data synchronization for three tables every one hour from 00:00 each day.

   Click **Commit**. Then, you can view the migration progress and results of each table.
10. Find table a1 and click View Node to view the migration results.

You have configured a node for migrating a MySQL connection named clone_database to MaxCompute. This node is run based on the specified schedule, daily by default. You can also create retroactive node instances to transmit historical data. The database migration feature of **Data Integration** significantly simplifies the initial configurations for migrating your data to the cloud and reduces data migration costs.

You can view the migration success logs of table a1.

3.10.3. Migrate Oracle databases

This topic describes how to migrate an Oracle database to MaxCompute.

The database migration feature improves efficiency and reduces costs. It can quickly upload all tables in an Oracle database to MaxCompute. For more information, see Overview.

Procedure

1. Log on to the DataWorks console.
2. Click the DataWorks icon in the upper-left corner and choose **All Products > Data Integration** to go to the **Data Integration** page.
3. In the left-side navigation pane, choose **Sync Resources > Connections**. On the page that appears, click **Add Connection** in the upper-right corner.
4. In the **Add Connection** dialog box that appears, select **Oracle**.
5. Add an Oracle connection named clone_databae for database migration.
6. Click **Test Connection** and verify that the database can be accessed. Click **Complete**.
7. The added Oracle connection named clone_databae appears in the connection list. Find the added connection and click **Migrate Database** in the Actions column.

   The database migration settings page consists of three functional modules.

Functional module	Description
Tables to migrate	This module lists all the tables in the Oracle connection named clone_databae. Selected tables will be migrated.
Advanced Settings	You can configure the rules for converting the table name, column names, and data types.
Basic settings	You can select whether to synchronize full or incremental data, whether to upload data in one or more batches, and the synchronization efficiency. You can also view the migration progress and results.

8. Click **Advanced Settings** and configure conversion rules based on your needs.
9. Set Sync Method to Synchronize All Data Daily.

   > **Note** If a date column exists in your table, you can select incremental migration and configure the incremental data to be determined based on the date column. Data Integration will generate WHERE clauses based on the specified column and DataWorks scheduling parameters such as ${bdp.system.bizdate}.

   Select data upload in batches to protect the Oracle database from being overloaded. Let Data Integration start data synchronization for three tables every one hour from 00:00 each day.

   Click **Commit**. Then, you can view the migration progress and results of each table.
10. Find a related table and click **View Node** to view the node details.

    You have configured a node for migrating an Oracle connection named clone_databae to MaxCompute. This node is run based on the specified schedule, daily by default. You can also create retroactive node instances to transmit historical data. The database migration feature of **Data Integration** significantly simplifies the initial configurations for migrating your data to the cloud and reduces data migration costs.

4.Data Analytics

4.1. Solution

The data analytics mode of DataWorks is upgraded so that you can group multiple workflows in a solution of a workspace.

Overview

DataWorks upgrades the data analytics mode to organize various types of nodes based on the business category. You can organize workflows to analyze data by business.

By using the data analytics mode that involves the **workspace**, **solution**, and **workflow**, DataWorks defines a new development process and improves user experience.

- A workspace is the basic organizational unit that manages the development and O&M permissions of users. The code of all nodes in a workspace can be collaboratively developed and managed by workspace members.
- A solution contains one or more workflows. It has the following advantages:
  - A solution can contain multiple workflows.
  - A workflow can be added to multiple solutions.
  - All solutions in a workspace can be collaboratively developed and managed by workspace members.
- A workflow is an abstract entity of business that enables you to develop data analytics code from a business perspective. A workflow can be added to multiple solutions. It has the following advantages:
  - Workflows facilitate business-oriented code development. Nodes in a workflow are organized by type. A hierarchical directory structure is supported. We recommend that you create a maximum of four levels of sub-directories. To create a sub-directory, right-click the target node type and select **Create Folder**.
  - You can view and optimize each workflow from a business perspective.
  - You can view each workflow on a dashboard to develop code with improved efficiency.
  - You can deploy and manage each workflow as a whole.

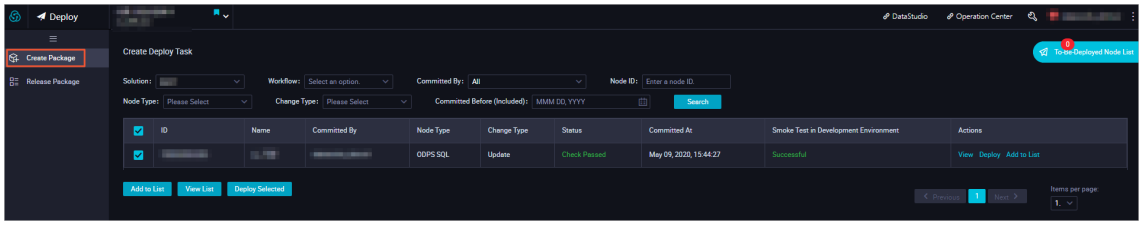
Develop a solution

If you double-click a solution in the left-side navigation pane, the left-side navigation pane only displays workflows in the solution. This prevents the development process from being affected by the code that is not related to the current solution in the workspace. To develop a solution, perform the following steps:

1. Log on to the DataWorks console.
2. On the **Data Analytics** tab, move the pointer over 

and select **Solution**.
3. In the **Create Solution** dialog box, set **Solution Name** and **Description**, select a workflow from the **Workflows** drop-down list, and then click **Create**.
4. In the solution list, right-click the created solution and select **Solution Kanban**. On the solution dashboard that appears, you can view the selected workflows or modify the solution.
5. Move the pointer over the solution name. The  and  icons appear.

- Click the icon. The **Deploy** page appears. You can view the nodes to be deployed in the current solution.



> **Note** This icon is available only when the workspace is in standard mode.

- Click the icon to go to the **Cycle Instance** page under **Cycle Task Maintenance** in **Operation Center**. You can view recurring instances of all nodes in the current solution.

In the left-side navigation pane, double-click the created solution. All the created workflows in the solution appear. You can click a workflow name to show the created nodes in it and perform operations on the nodes and the workflow.

A workflow can be added to multiple solutions. After you develop a solution and add a workflow to the solution, other users can edit the workflow you referenced in their solutions for collaborative development.

4.2. SQL coding guidelines and specifications

This topic describes the basic guidelines and detailed specifications of SQL coding.

SQL coding guidelines

The SQL coding guidelines are as follows:

- Make sure that the code is comprehensive.
- Make sure that code lines are clear, neat, well-organized, and structured.
- Consider the optimal execution speed during SQL coding.
- Provide comments whenever necessary to enhance the readability of your code.
- The guidelines impose non-mandatory constraints on the coding behavior of developers. In practice, understandable deviations are allowed when developers obey general rules.
- Use lowercase letters for all keywords and reserved words. Keywords and reserved words include select, from, where, and, or, union, insert, delete, group, having, and count.
- In addition to keywords and reserved words, other code such as field names and table alias must be in lowercase.
- A unit of indentation contains four spaces. All indentations must be the integral multiple of an indentation unit. The code is aligned according to its hierarchy.
- The `select *` operation is prohibited. The column name must be specified for all operations.
- Matching opening and closing parentheses must be placed in the same column.

SQL coding specifications

The SQL coding specifications are as follows:

- Code header

  The code header contains information such as the subject, description, author, and date. Reserve a line for change log and a title line so that later users can add change records. Each line can contain a maximum of 80 characters. The template is as follows:

```
-- MaxCompute(ODPS) SQL
--********************************************************************
-- ** Subject: Transaction
-- ** Description: Transaction refund analysis
-- ** Author: Youma
-- ** Created on: 20170616
-- ** Change log:
-- ** Modified on Modified by Content
-- yyyymmdd name comment
-- 20170831 Wuma Add a comment on the biz_type=1234 transaction
--********************************************************************
```

- Field arrangement
  - Use a line for each field that is selected for the SELECT statement.
  - Separate the first field from SELECT by one indentation unit.
  - Enter another field name in a separate line after two indentation units.
  - Place the comma (,) between two fields right before the second field.
  - Place the AS statement in the same line as the corresponding field. We recommend that you keep the AS statements of multiple fields in the same column.

```
select  channel_id              as channel_id
       ,trade_channel_desc      as trade_channel_desc
       ,trade_channel_edesc     as trade_channel_edesc
       ,inst_date               as inst_date
       ,trade_iswap             as trade_iswap
       ,channel_type            as channel_type
       ,channel_second_desc     as channel_second_desc
from   (
```

- Clause arrangement for an INSERT statement

  Arrange the clauses of an INSERT statement in the same line.

- Clause arrangement for a SELECT statement

  The clauses such as FROM, WHERE, GROUP BY, HAVING, ORDER BY, JOIN, and UNION in a SELECT statement must be arranged according to the following requirements:

  - Use a line for each clause.
  - Make sure that the clauses are left aligned with the SELECT statement.
  - Add two indentation units between the first word and the other code in a clause.
  - Keep the logical operators such as AND and OR in a WHERE clause left aligned with WHERE.

- If the length of a clause name exceeds two indentation units such as ORDER BY and GROUP BY, add a space between the clause name and its content.

```
select      trim(channel) channel
            ,min(id)      id
from        ods_trd_trade_base_dd
where       channel is not null
and         dt = ${tmp_uuuummdd}
and         trim(channel) <> ''
group by    trim(channel)
order by    trim(channel)
```

- Spacing before and after operators

  Keep one space before and one space after the arithmetic and logical operators and keep the operators in the same line, unless the clause contains more than 80 characters.

```
select      trim(channel) channel
            ,min(id)      id
from        ods_trd_trade_base_dd
where       channel is not null
and         dt = ${tmp_uuuummdd}
and         trim(channel) <> ''
group by    trim(channel)
order by    trim(channel)
```

- CASE statement

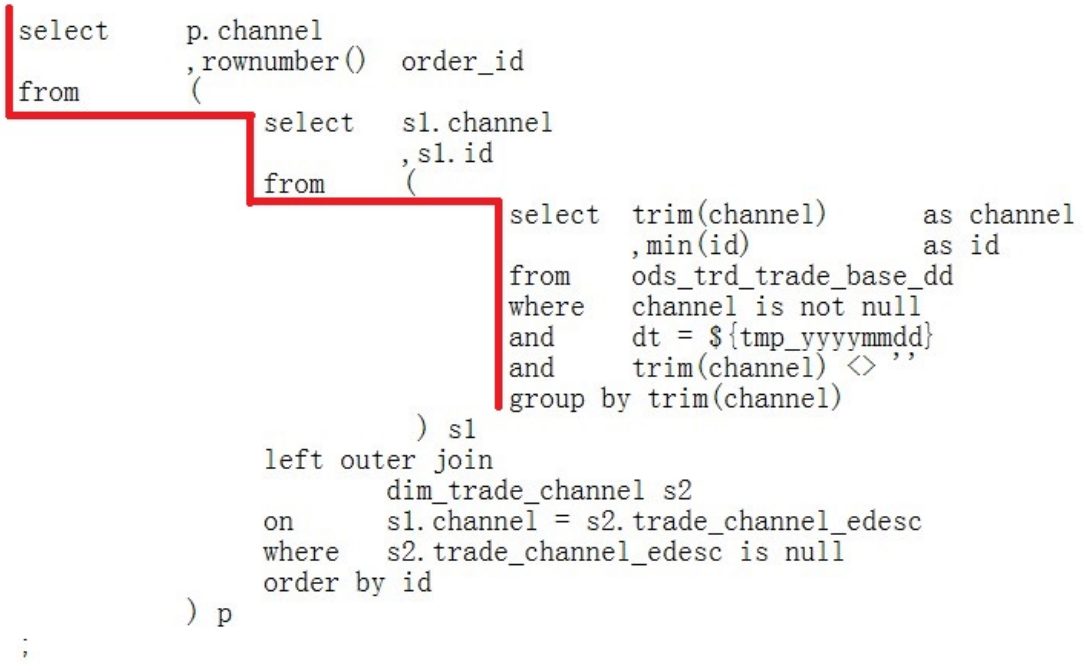
  The CASE statement can be used to determine the value of a field in a SELECT statement. Rules for writing CASE statements are as follows:

  - Place the WHEN clause in the same line as the CASE statement, with one indentation unit between them.
  - Keep a WHEN clause in one line whenever possible. If the statement is long, line breaks can be made.
  - A CASE statement must contain an ELSE clause. The ELSE clause must be aligned with the WHEN clause.

```
,case   when p1.trade_from = '3008' and p1.trade_email is null then 2
        when p1.trade_from = '4000' and p1.trade_email is null then 1
        when p9.trade_from_id is not null then p9.trade_from_id
 end                                as trade_from_id
,p1.trade_email                     as partner_id
```

- Nested query

Nested queries are often used in extract-transform-load (ETL) development of data warehouse systems. The following figure shows an example of a nested query.



```
select     p.channel
           ,rownumber()  order_id
from       (
               select   s1.channel
                        ,s1.id
               from     (
                            select  trim(channel)      as channel
                                    ,min(id)           as id
                            from    ods_trd_trade_base_dd
                            where   channel is not null
                            and     dt = ${tmp_yyyymmdd}
                            and     trim(channel) <> ''
                            group by trim(channel)
                        ) s1
               left outer join
                        dim_trade_channel s2
               on       s1.channel = s2.trade_channel_edesc
               where    s2.trade_channel_edesc is null
               order by id
           ) p
;
```

- Table alias
  - Once an alias is defined for a table in a SELECT statement, use the alias whenever you reference the table in the statement. Therefore, you must specify an alias for each table.
  - We recommend that you define the table aliases with simple characters, such as a, b, c, and d in sequence, and avoid using keywords.

- In the nested query, levels 1 to 4 of SQL statements are named part, segment, unit, and detail, which are abbreviated as P, S, U, and D. You can also use a, b, c, and d to represent levels 1 to 4.

  To differentiate multiple clauses at the same level, add numbers such as 1, 2, 3, and 4 next to the letters. Add comments to the table aliases as needed.

```
select     p.channel
           ,rownumber()  order_id
from       (
               select   s1.channel
                        ,s1.id
               from     (
                            select  trim(channel)      as channel
                                    ,min(id)           as id
                            from    ods_trd_trade_base_dd
                            where   channel is not null
                            and     dt = ${tmp_yyyymmdd}
                            and     trim(channel) <> ''
                            group by trim(channel)
                        ) s1
               left outer join
                       dim_trade_channel s2
               on      s1.channel = s2.trade_channel_edesc
               where   s2.trade_channel_edesc is null
               order by id
           ) p
;
```

- SQL comments
  - Add a comment for each SQL statement.
  - Use a separate line for the comment of each SQL statement and place the comment in front of the SQL statement.
  - Place the comment of a field right after the field.
  - Add comments for clauses that are difficult to understand.
  - Add comments for important code.
  - If a statement is long, we recommend that you add comments based on the purposes of each segment.
  - The description for a constant or variable is required. The comment on the valid value range is optional.

4.3. GUI elements

4.3.1. Overview

This topic describes the graphical user interface (GUI) elements on the DataStudio page and the configuration tab of an ODPS SQL node.

Log on to the DataWorks console. The **Data Analytics** page appears. You can double-click a created node to perform operations on the node configuration tab.

The following table describes the GUI elements.

No.	GUI element	Description
1	**Show My Nodes Only icon**	Click the icon to view your own nodes.
2	**Search Code icon**	Click the icon to search for a node or a code segment.
3	**Create icon**	Click the icon to create a solution, workflow, folder, node, table, resource, or function.
4	**Refresh icon**	Click the icon to refresh the directory tree in the left-side navigation pane.
5	**Locate icon**	Click the icon to find the current node in the left-side navigation pane.
6	**Import icon**	Click the icon to import local data to an online table. You must specify the encoding format. **Note** In a workspace of the standard mode, the local data is imported to a table in the development environment.
7	**Filter icon**	Click the icon to query nodes based on the specified filter conditions.
8	**Save icon**	Click the icon to save the code of the current node.
9	**Save as Ad-Hoc Query Node icon**	Click the icon to save the code of the current node in an ad-hoc query node. You can find the node on the Ad-Hoc Query tab.
10	**Commit icon**	Click the icon to commit the current node.
11	**Commit and Unlock icon**	Click the icon to commit and unlock the current node for editing.
12	**Steal Lock icon**	Click the icon to steal the lock of the current node and then edit it if you are not the owner of the node.
13	**Run icon**	Click the icon to run the code of the current node. You only need to assign values to variables in SQL statements once. The initial values are retained even if the node code changes.

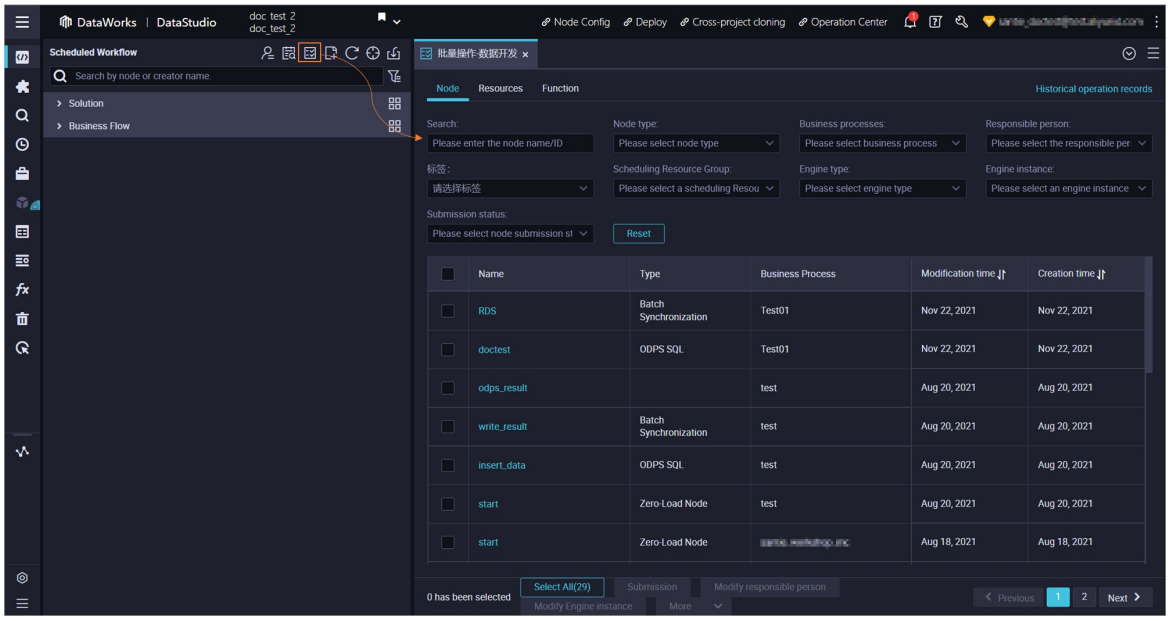
No.	GUI element	Description
14	**Run with Arguments icon**	Click the icon to run the code of the current node with the configured parameters. You must manually assign values to variables in SQL statements each time you click this icon. The initial values are passed to the **Run with Arguments** feature, which replaces the initial values with the assigned values. For example, if the run date of a node is set to April 2, the node always runs on April 2 when you click the Run icon. After you click Run with Arguments icon and change the run date to April 3, the run date is updated. When you click the Run icon again, the node is run on April 3.
15	**Stop icon**	Click the icon to stop running the code of the current node.
16	**Reload icon**	Click the icon to reload the code of the current node. The code will be restored to the version last saved. Unsaved changes will be lost.
17	**Run Smoke Test icon**	Click the icon to test the code of the current node. A smoke test allows you to replace the values of scheduling parameters in the specified data timestamp with your simulated ones. This feature tests the effect of value changes for scheduling parameters. **Note** Each time after you modify the values of scheduling parameters, you must save and commit the modification before running the smoke test. Otherwise, the new values of scheduling parameters do not take effect.
18	**View Smoke Test Log icon**	Click the icon to view the runtime logs of the current script template.
19	**Format Code icon**	Click the icon to format the code to avoid excessively long code in a single line.
20	**Operation Center button**	Click the icon to go to Operation Center.
21	**Properties tab**	Click the tab to configure the properties such as the scheduling properties, parameters, and resource group for the current node.
22	**Lineage tab**	Click the tab to view the relationships between the current node and other nodes.
23	**Versions tab**	Click the tab to view the committed and deployed versions of the current node.
24	**Code Structure tab**	Click the tab to view the code structure of the current node. If the code is excessively long, you can quickly find a code segment based on the key information in the structure.

4.3.2. Perform batch operations

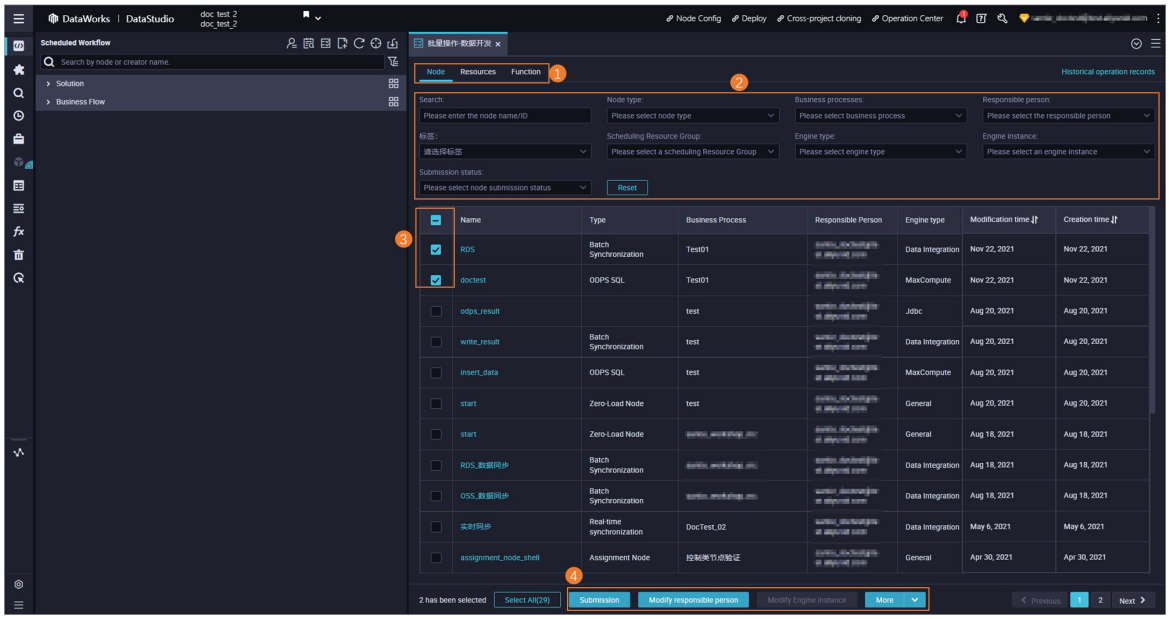
DataWorks allows you to perform operations on multiple nodes, resources, or functions at a time. For example, you can change owners. This topic describes how to perform batch operations.

Procedure

1. Log on to the DataWorks console. Go to the **DataStudio** page. In the left-side navigation pane, click the **Batch Operation** icon in the top toolbar.



2. Modify multiple DataWorks objects at a time.



   i. On the **Batch Operation-Data Development** tab, you can perform operations on multiple nodes, resources, or functions at a time on the **Node**, **Resources**, or **Function** tab.

   ii. You can filter data by conditions such as **Node Type** and **Workflow** in the upper part of the tab.

iii. If the search results are displayed on the tab, you can select multiple nodes, resources, or functions that you want to modify.

iv. After that, click a button in the lower part to perform the operation on the selected objects as needed.

4.3.3. Workflow Parameters

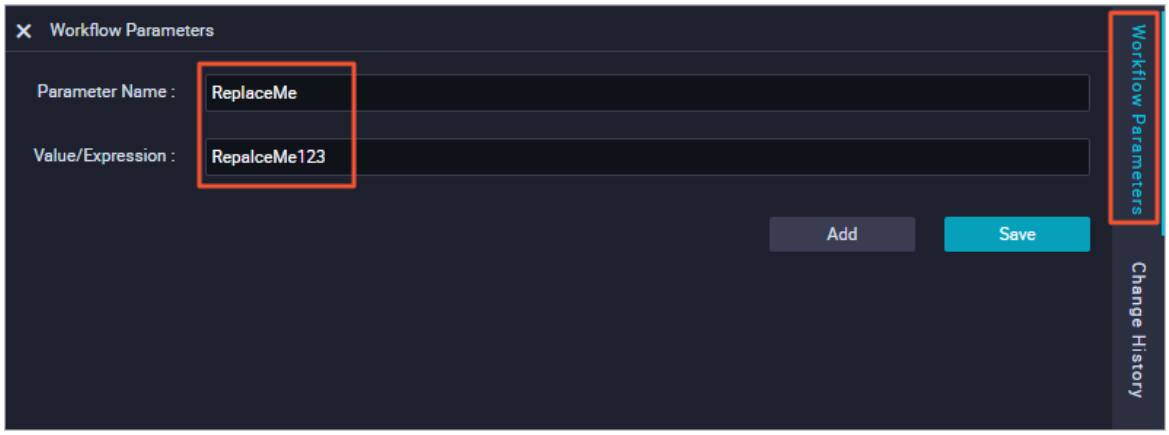
On the Workflow Parameters tab, you can assign a value to a variable or replace the value of a parameter for all nodes in the current workflow. This topic describes how to configure a workflow parameter by assuming that you want to replace the value of the ReplaceMe parameter with ReplaceMe123 in a manually triggered workflow.

Limits

- In manually triggered workflows, ODPS SQL nodes, Shell nodes, and sync nodes support global parameters. The format for specifying a global parameter varies based on the node type. For example, a global workflow parameter is specified as x=y1.
  - To configure the workflow parameter for an ODPS SQL node, double-click the target node and click the **General** tab in the right-side navigation pane. On the **General** tab, enter x=aaa in the Arguments field. When the node is run, x=aaa specified in the Arguments field is replaced with x=y1. You can use $x to reference the workflow parameter in the code.
  - To configure the workflow parameter for a Shell node, double-click the target node and click the **General** tab in the right-side navigation pane. On the **General** tab, enter $x in the Arguments field. When the node is run, x=aaa specified in the Arguments field is replaced with y1. You can use $1 to reference the workflow parameter in the code.
  - To configure the workflow parameter for a sync node, double-click the target node and click the **General** tab in the right-side navigation pane. On the **General** tab, enter -p"-Dx=aaa" in the Arguments field. When the node is run, x=aaa specified in the Arguments field is replaced with -p"-Dx=y1. You can use $x to reference the workflow parameter in the code.
- In auto triggered workflows, only ODPS SQL nodes support global parameters.
- Parameter names and values are case-sensitive.

Configure a workflow parameter

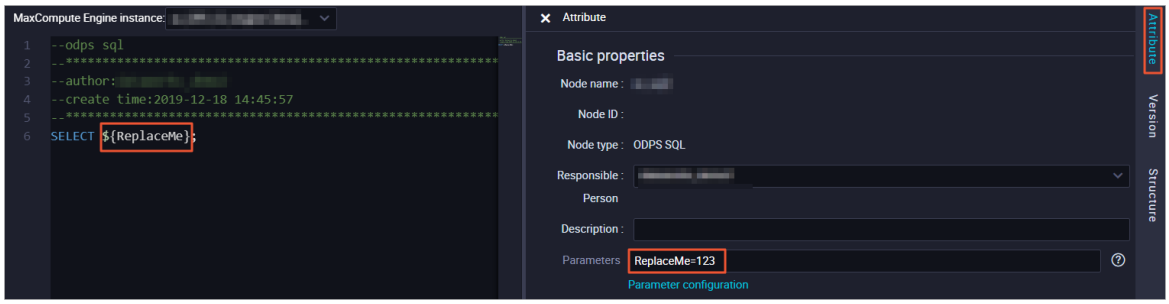
1. Log on to the DataWorks console.
2. In the left-side navigation pane, click **Manually Triggered Workflows**.
3. Double-click the target workflow to go to the workflow configuration tab.
4. In the right-side navigation pane, click the **Workflow Parameters** tab. In the Workflow Parameters pane, enter ReplaceMe in the **Parameter Name** field and ReplaceMe123 in the **Value/Expression** field.



5. Click in the toolbar.

Configure the workflow parameter for an ODPS SQL node

1. On the **DataStudio** page, click **Manually Triggered Workflows** in the left-side navigation pane.
2. Find the target workflow and choose **MaxCompute > Data Analytics** to show all the existing data analytics nodes. Double-click the target ODPS SQL node to go to the node configuration tab.
3. In the right-side navigation pane, click the **General** tab. In the General pane, enter ReplaceMe=123 in the **Arguments** field.

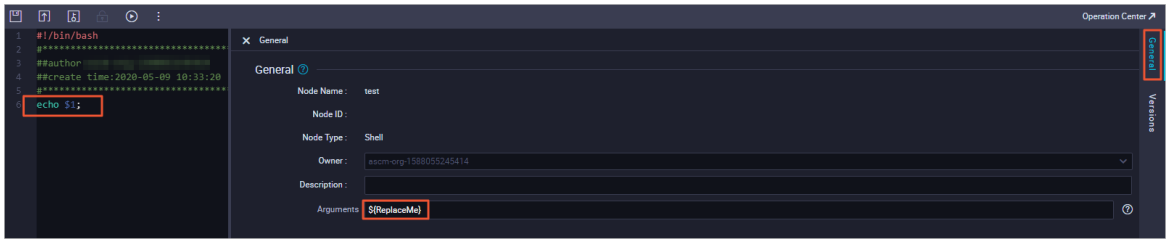


The workflow parameter is specified as ReplaceMe=ReplaceMe123. Therefore, the workflow parameter ReplaceMe is assigned the value ReplaceMe123 for this node when the workflow is run.

4. Click in the toolbar.

Configure the workflow parameter for a Shell node

1. On the **DataStudio** page, click **Manually Triggered Workflows** in the left-side navigation pane.
2. Find the target workflow and click **General** to show all the existing data analytics nodes. Double-click the target Shell node to go to the node configuration tab.
3. In the right-side navigation pane, click the **General** tab. In the General pane, enter ${ReplaceMe} in the **Arguments** field.



> **Note** Make sure that you enter the parameter in the correct format.

4. Click in the toolbar.

Configure the workflow parameter for a sync node

1. On the **DataStudio** page, click **Manually Triggered Workflows** in the left-side navigation pane.
2. Find the target workflow and click **Data Integration** to show all the existing data integration nodes. Double-click the target sync node to go to the node configuration tab.
3. In the right-side navigation pane, click the **General** tab. In the General pane, enter -p"ReplaceMe=abc" in the **Arguments** field.

> **Note** Make sure that you enter the parameter in the correct format, namely, -p"-DParameter name=Parameter value".

4. Click in the toolbar.

Run the workflow to view the result

On the configuration tab of the workflow, click in the toolbar. In the Warning dialog box, click

Settings. In the **Runtime Parameters** dialog box, set Arguments to ReplaceMe. The value of the workflow parameter is replaced when the workflow is run.

You can use the following methods to view the value assigned to the workflow parameter for different types of nodes:

- Right-click the ODPS SQL node and select **View Log**. Then, you can view the value assigned to the workflow parameter for the ODPS SQL node.
- Right-click the Shell node and select **View Log**. Then, you can view the value assigned to the workflow parameter for the Shell node.
- Right-click the sync node and select **View Log**. Then, you can view the value assigned to the workflow parameter for the sync node.

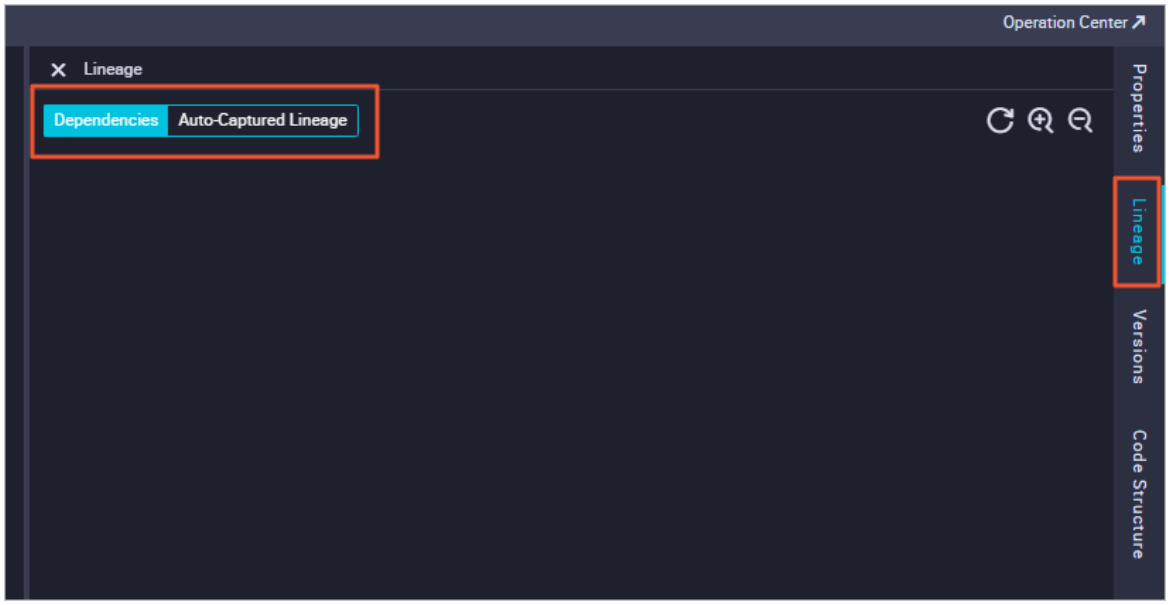
If you have not assigned a value to a workflow parameter on the **Workflow Parameters** tab for a manually triggered workflow, you must assign a value to the workflow parameter every time you run the workflow in the production environment.

4.3.4. Lineage

The Lineage tab displays the relationships between a node and other nodes. You can view the node dependencies and the lineage parsed from the code of the node.

Go to the Lineage tab

1. Log on to the DataWorks console.
2. Double-click the target node. For more information about how to create a node, see Create an ODPS SQL node.
3. On the node configuration tab that appears, click the **Lineage** tab in the right-side navigation pane.



On the **Lineage** tab, you can click **Dependencies** to view the dependencies or click **Auto-Captured Lineage** to view the lineage.

View the dependencies

You can check the node dependencies presented based on the current configuration. If the node dependencies fail to meet your expectations, you can reconfigure the node dependencies on the Properties tab.

View the auto-captured lineage

The lineage is parsed based on the code of the current node. For example, an ODPS SQL node contains the following SQL statements:

```
INSERT OVERWRITE TABLE dw_user_info_all_d PARTITION (dt='${bdp.system.bizdate}')
SELECT COALESCE(a.uid, b.uid) AS uid
  , b.gender
  , b.age_range
  , b.zodiac
  , a.region
  , a.device
  , a.identity
  , a.method
  , a.url
  , a.referer
  , a.time
FROM (
  SELECT *
  FROM ods_log_info_d
  WHERE dt = ${bdp.system.bizdate}
) a
LEFT OUTER JOIN (
  SELECT *
  FROM ods_user_info_d
  WHERE dt = ${bdp.system.bizdate}
) b
ON a.uid = b.uid;
```

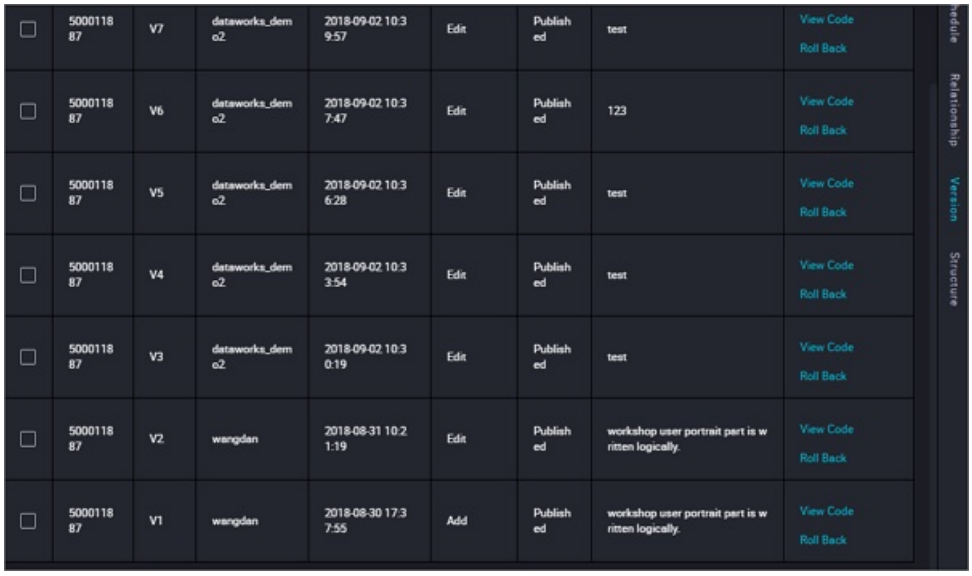
The following figure shows the lineage parsed from the preceding SQL statements. The results queried from the ods_log_info_d and ods_user_info_d tables are joined and then inserted into the dw_user_info_all_d table.

4.3.5. Versions

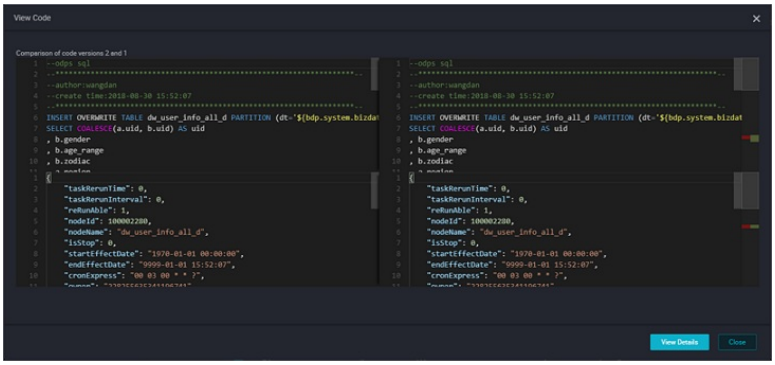
The Versions tab displays all committed and deployed versions of a node. You can view the historical versions and information about each version, including the user who committed the version, time when the version was committed, change type, status, and description.

> **Note** Only a committed node has the version information. Every time a node is committed, a version is generated and added to the **Versions** tab.

1. Log on to the DataWorks console.
2. On the **DataStudio** page, double-click the target node.
3. On the node configuration tab that appears, click the **Versions** tab in the right-side navigation pane. In the Versions pane, view the committed and deployed versions of the current node.



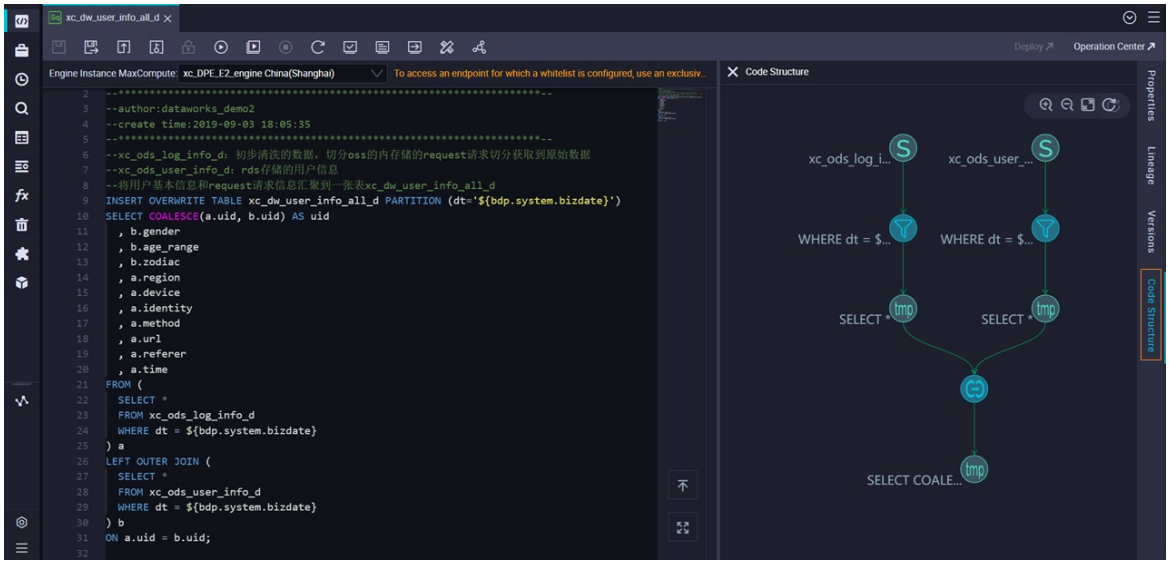
GUI element	Description
File ID	The ID of the node.
Versions	The version of the node. A version is generated each time the node is committed and deployed. V1 indicates version 1 and V2 indicates version 2. The version number is incremented by 1 each time.
Committed By	The user who committed the version.
Committed At	The time when the version was committed. If a version is committed and then deployed at a later time point, the value of this parameter is updated to the time when the version is deployed. By default, this column records the time when the version is last operated.
Change Type	The operation on the node. The value of this parameter is Create if the node is committed and deployed for the first time or Change if the node is modified, committed, and then deployed.
Status	The status of the version. Valid values: ○ **Yes**: The version is committed to the development environment but the related deployment task has not been created. The version has not been deployed in the production environment. ○ **Not Deployed**: The version is committed to the development environment and the deployment task is created. The version is pending for deployment. ○ **Deployed**: The version is committed to the development environment and deployed in the production environment.

GUI element	Description
Description	The change description of the version when it is committed. This description helps other users find the relevant version when they manage the node.
Actions	The actions that you can perform on the version. Two actions are available: **View Code** and **Roll Back**. ○ **View Code**: Click the button to view the code of the current version. ○ **Roll Back**: Click the button to roll back the node from the current version to the required version. After you roll back a node, you must commit and deploy it again.
Compare	Click the button to compare the code and properties between two selected versions.  Click **View Details**. On the details page that appears, you can view the changes in code and properties. **Note** You can only compare two versions and cannot compare one or more than two versions at a time.

4.3.6. Code Structure

The Code Structure tab displays the SQL code structure parsed from the code of a node. The code structure helps you view and modify the code.

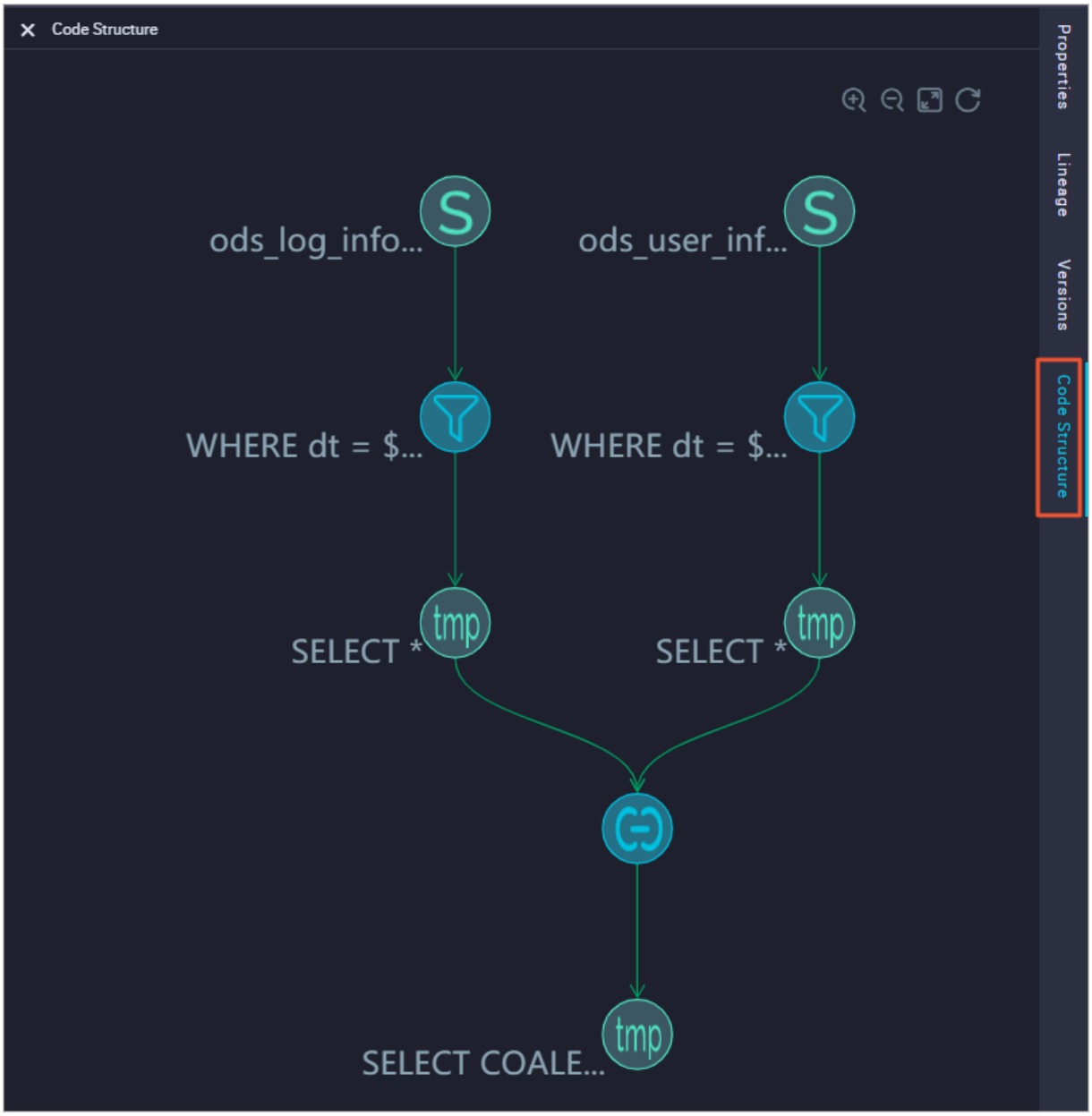
1. Log on to the DataWorks console.
2. Double-click the ODPS SQL node whose code structure you want to view. For more information about how to create a node, see Create an ODPS SQL node.
3. On the node configuration tab that appears, click the **Code Structure** tab in the right-side navigation pane.



In this example, the ODPS SQL node contains the following SQL statement:

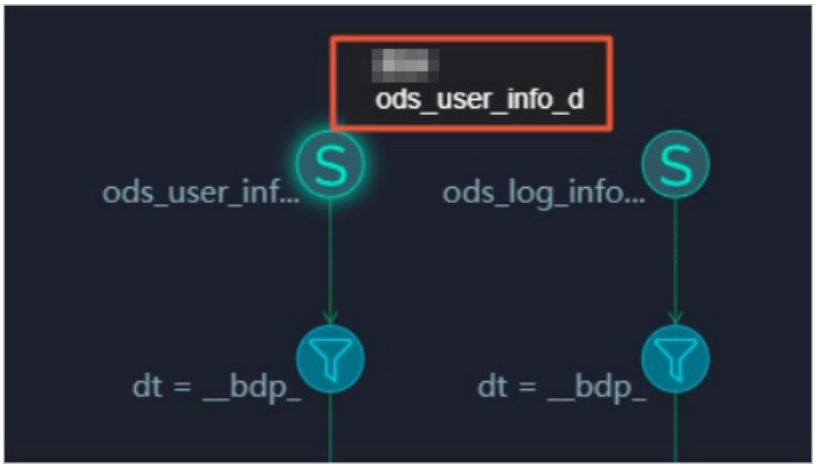
```
INSERT OVERWRITE TABLE dw_user_info_all_d PARTITION (dt='${bdp.system.bizdate}')
SELECT COALESCE(a.uid, b.uid) AS uid
  , b.gender
  , b.age_range
  , b.zodiac
  , a.region
  , a.device
  , a.identity
  , a.method
  , a.url
  , a.referer
  , a.time
FROM (
  SELECT *
  FROM ods_log_info_d
  WHERE dt = ${bdp.system.bizdate}
) a
LEFT OUTER JOIN (
  SELECT *
  FROM ods_user_info_d
  WHERE dt = ${bdp.system.bizdate}
) b
ON a.uid = b.uid;
```

The following figure shows the code structure parsed from the preceding SQL statement.

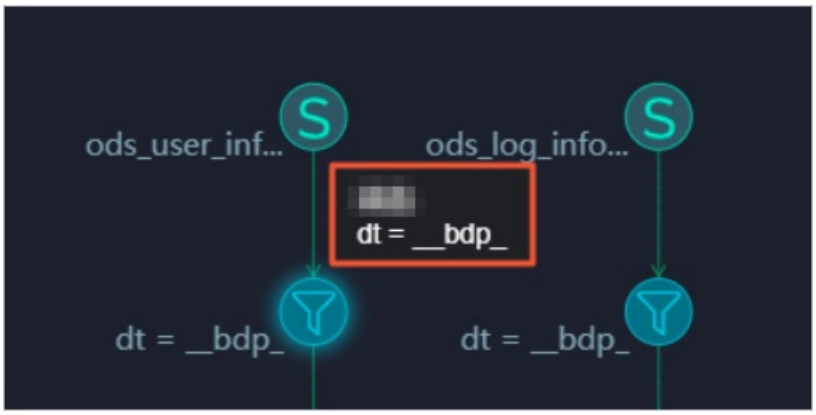


Move the pointer over a circle to view the description.

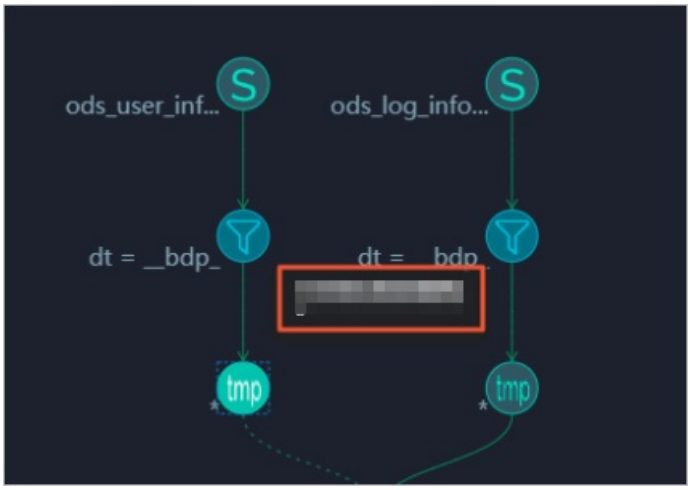
- **Source table**: the table to be queried.



- **Filter**: the condition for filtering the partitions in the table to be queried.

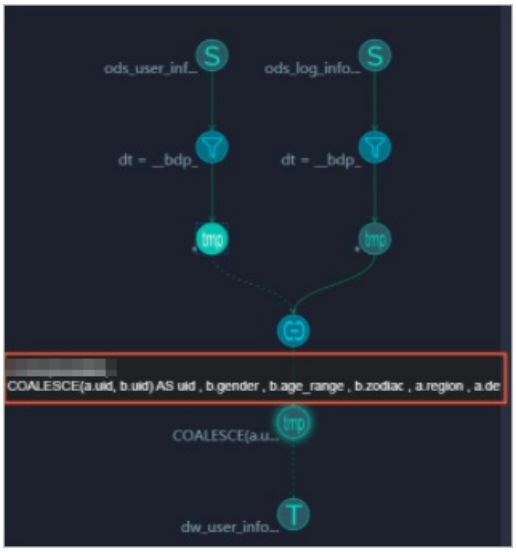


- **First intermediate table (view)**: the temporary table that stores the query results.

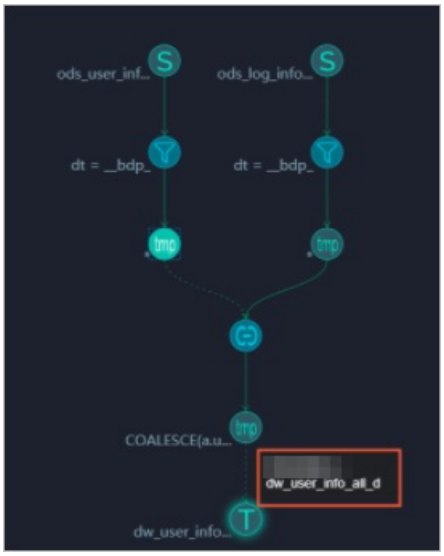


- **Join**: the operation for joining the query results.

- **Second intermediate table (view)**: the temporary table that stores the results of the JOIN operation. This temporary table can be stored for three days. After three days, this table is automatically deleted.



- **Destination table (insert)**: the destination table to which the query results are inserted by using an **INSERT OVERWRITE** statement.



4.4. Business flows

4.4.1. Overview

DataWorks organizes different types of nodes in a workflow by business category. This allows you to develop code by business.

DataWorks provides you with a dashboard for different types of nodes in each workflow. DataWorks also provides tools for you to optimize and manage nodes in each workflow. This promotes easy and intelligent development and management.

Workflow structure

A workspace supports multiple types of compute engines and multiple workflows. A workflow is a collection of various types of nodes that are closely associated with each other. DataWorks automatically generates a DAG so that you can view the workflow. A workflow supports the following types of nodes: data integration, data analytics, table, resource, function, and algorithm.

Each type of node has an independent folder. You can also create subfolders in each folder. To facilitate management, we recommend that you create a maximum of four levels of subfolders. If more than four subfolder levels are required, your workflow is too complex. We recommend that you split the workflow into two or more workflows and add the split workflows to one solution.

Create a workflow

1. Log on to the DataWorks console.
2. In the left-side navigation pane, click **Data Analytics**.
3. On the **Data Analytics** tab, right-click **Business Flow** and select **Create Workflow**.
4. In the **Create Workflow** dialog box, set **Workflow Name** and **Description**.

   > **Notice** The name of the workflow, which cannot exceed 128 characters in length.

5. Click **Create**.

Workflow nodes

A workflow consists of the following types of nodes:

- **Data integration**

  Click the target workflow and double-click **Data Integration** to view all data integration nodes of the workflow.

- **MaxCompute**

  The MaxCompute engine supports various data analytics nodes, such as ODPS SQL, SQL Snippet, ODPS Spark, PyODPS, ODPS Script, and ODPS MR nodes. You can also view and create tables, resources, and functions.

  - **Data analytics**

    Right-click Data Analytics under **MaxCompute** in the target workflow and select Create to create a data analytics node of a specific type.

  - **Table**

    Click the target workflow and choose **Create > Table** under **MaxCompute** to create a table. You can also view all the tables that are created in the current MaxCompute project.

  - **Resource**

    Click the target workflow, choose **Create > Resource** under **MaxCompute**, and then click a specific resource type to create a resource. You can also view all the resources that are created in the current MaxCompute project.

  - **Function**

    Click the target workflow and choose **Create > Function** under **MaxCompute** to create a function. You can also view all the functions that are created in the current MaxCompute project.

- **EMR**

  The E-MapReduce compute engine supports the following types of data analytics nodes: EMR Hive, EMR MR, EMR Spark, and EMR Spark SQL. You can also view and create E-MapReduce resources.

  > **Note** The **EMR** folder is available only after you create an E-MapReduce compute engine on the **Project Management** page.

  - **Data analytics**

    Click the target workflow, right-click **Data Analytics** under **EMR**, and then select Create to create a data analytics node of a specific type.

  - **Resource**

    Click the target workflow, right-click **Resource** under **EMR**, and then select Create to create a resource of a specific type. You can also view all the resources that are created in the current E-MapReduce compute engine.

- **Algorithm**

  Click the target workflow, right-click **Algorithm**, and then choose Create > PAI Experiment to create a PAI Experiment node. You can also view all the PAI Experiment nodes that are created in the current workflow.

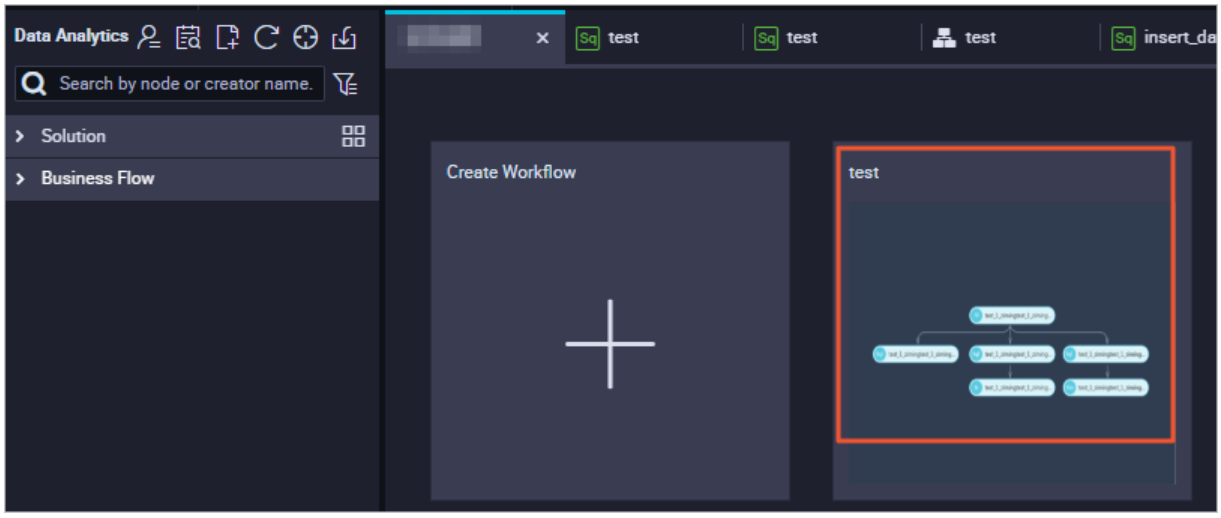
- **General**

Click the target workflow, right-click **General**, and then select Create to create a node of a specific type.

View all workflows

On the **Data Analytics** tab, right-click **Business Flow** and select **All Workflows** to view all workflows that are created in the current workspace.

Click a workflow. The dashboard of the workflow appears.



View the dashboard for each node type

DataWorks provides a dashboard for each type of nodes in a workflow. On the dashboard, each node is presented by a card that offers operation and optimization suggestions, so that you can intelligently manage nodes.

For example, the card of each data analytics node provides two indicators to show whether baseline-based monitoring and event notification are enabled for the node. This allows you to understand the status of each node.

You can double-click a folder in a workflow to view the dashboard of the selected node type.

Commit a workflow

1. Go to the dashboard of a workflow and click in the toolbar.
2. In the **Commit** dialog box, select the nodes to be committed, set **Description**, and then select **Ignore I/O Inconsistency Alerts**.
3. Click **Commit**.

> **Note** If a node has been committed but the node code is not modified, the node cannot be selected again. In this case, you can enter your comments on the node and click **Commit**. The property changes of the node are automatically committed.

4.4.2. Create and reference a node group

This topic describes how to create and reference a node group.

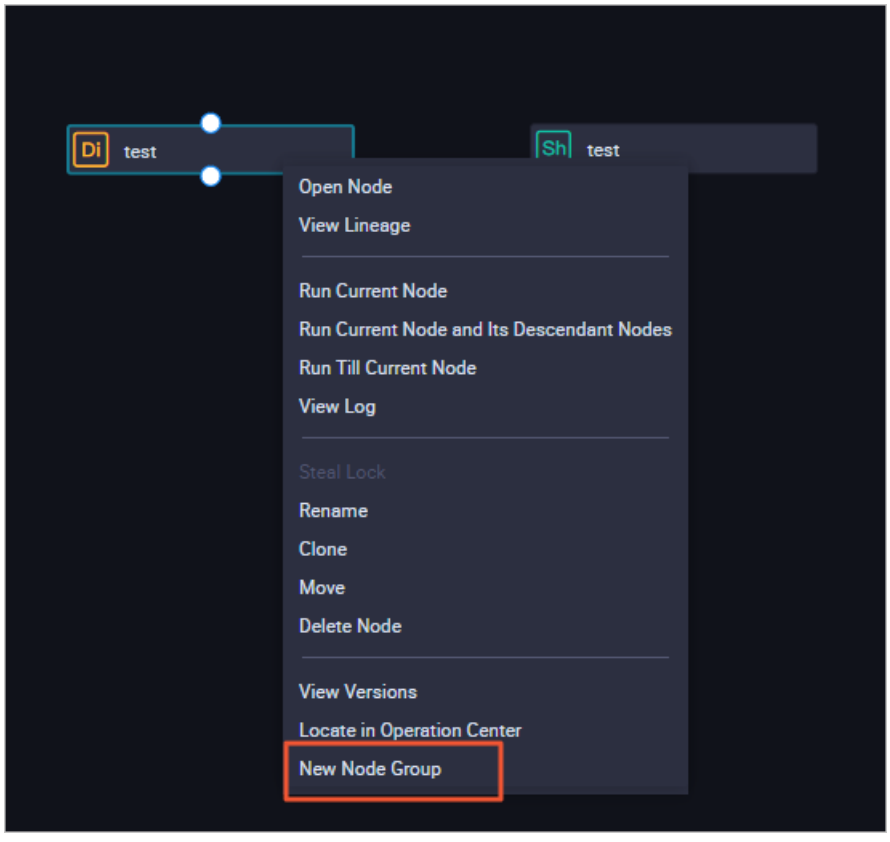
Context

You can group several nodes that are frequently reused together as a node group. The configuration of each node remains unchanged after the nodes are added to a node group. Later, you can directly reference the node group to reuse these nodes.

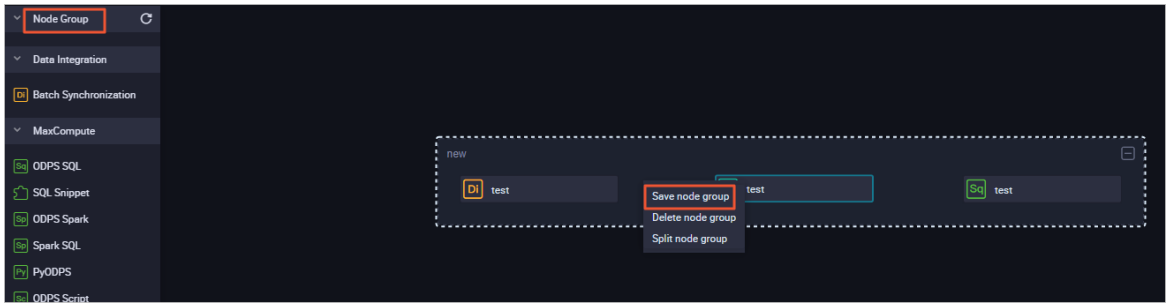
Create a node group

1. Log on to the DataWorks console.
2. On the **Data Analytics** tab, create a workflow. For more information, see Create a workflow.
3. Go to the dashboard of the created workflow. Click in the upper-right corner and drag a box to select the target nodes to be included in a node group.

4. Right-click any node among the selected nodes and select **New Node Group**.



5. In the **New Node Group** dialog box, enter a name in the **Name** field and click **OK**.
6. Right-click the node group and select **Save node group**. In the dialog box that appears, click OK. Then, you can view the created node group in the **Node Group** section.



Menu item	Description
Save node group	Save the node group. The node group that you have created appears in the Node Group section only after you click **Save node group**. A node group that is not saved cannot be referenced in other workflows.
Delete node group	Delete the node group. Click **Delete node group** to delete all nodes in the selected node group.
Split node group	Dismiss the node group. After the node group is dismissed, the selected nodes no longer form a node group in the workflow. However, the node group still exists in the Node Group section.

> **Note** If the created node group contains a PAI Experiment node, create a PAI experiment in another workflow to reference the node group. If the created node group contains a branch node, add digits to the value in the **Associated Node Output** parameter.

Reference a node group

You can directly drag a node group to another workflow to reference the node group in the workflow. The dependencies among the nodes in the node group remain unchanged.

You can run the workflow or commit and deploy the workflow. Then, go to **Operation Center** to view the running result.

4.5. Node types

4.5.1. Data Integration

4.5.1.1. Create a batch sync node

Batch sync nodes support various types of data stores, including MaxCompute, MySQL, DRDS, SQL Server, PostgreSQL, Oracle, MongoDB, Db2, Table Store, OSS, FTP, HBase, LogHub, HDFS, and Stream.

Context

When you enter a table name, a drop-down list appears, displaying all matched tables. Only exact match is supported. Therefore, you must enter a complete table name. Tables are labeled as unsupported if they are not supported by batch sync nodes.

If you move the pointer over a table in the list, the details of the table appear, including the database, IP address, and owner of the table. After you select a table, the column information is automatically entered. You can add, move, and delete columns.

Procedure

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over +Create and choose **Data Integration > Batch Synchronization**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **Data Integration**, and then choose **Create > Batch Synchronization**.
3. In the **Create Node** dialog box, set **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.
4. Click **Commit**.
5. Configure the batch sync node. For more information, see Overview.
6. On the configuration tab of the batch sync node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, set parameters in the Schedule section. For more information, see Basic properties.
7. Commit the node.

   > **Notice** You must set **Rerun** and **Parent Nodes** before you can commit the node.

   i. Click in the toolbar.

   ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.

   iii. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the batch sync node. For more information, see Publish nodes.
8. Test the batch sync node. For more information, see Manage auto triggered nodes.

4.5.2. MaxCompute

4.5.2.1. Create an ODPS SQL node

Using the SQL-like syntax, ODPS SQL nodes can process terabytes of data in distributed processing scenarios that do not require real-time processing.

Context

Generally, it takes a long time from preparing to committing a job. You can use ODPS SQL nodes to process thousands to tens of thousands of transactions. ODPS SQL nodes are online analytical processing (OLAP) applications designed to deal with large amounts of data.

Limits

- You cannot use SET statements, USE statements, or SQL alias statements independently in the code of an ODPS SQL node. They must be executed together with other SQL statements. For example, you can use a SET statement together with a CREATE TABLE statement.

```
set a=b;
create table name(id string);
```

- You cannot add comments to statements containing keywords, including SET statements, USE statements, and SQL alias statements, in the code of an ODPS SQL node. For example, the following comment is not allowed:

```
create table name(id string);
set a=b; // Comment.
create table name1(id string);
```

- The running of an ODPS SQL node during workflow development and the scheduled running of an ODPS SQL node have the following differences:
  - Running during workflow development: combines all the statements containing keywords, including SET statements, USE statements, and SQL alias statements, in the node code and executes them before executing other SQL statements.
  - Scheduled running: executes all SQL statements in sequence.

```
set a=b;
create table name1(id string);
set c=d;
create table name2(id string);
```

The following table shows the differences between the two running modes for the preceding SQL statements.

SQL statement	Running during workflow development	Scheduled running
First SQL statement	`set a=b;` `set c=d;` `create table name1(id string);`	`set a=b;` `create table name1(id string);`
Second SQL statement	`set a=b;` `set c=d;` `create table name2(id string);`	`set c=d;` `create table name2(id string);`

- You must specify a scheduling parameter in the format of key=value. Do not add any spaces before or after the equation mark (=). Examples:

```
time = {yyyymmdd hh:mm:ss} // Incorrect format.
a =b // Incorrect format.
```

- If you use keywords such as bizdate and date as scheduling parameters, you must specify the values

in the format of yyyymmdd. If you want to use other time formats, do not use the preceding keywords as scheduling parameters. Example:

```
bizdate=201908 // Incorrect format.
```

- You can only use statements starting with SELECT, READ, or WITH to query the result data for a node during the workflow development. Otherwise, no results are returned.
- Separate multiple SQL statements with semicolons (;) and place them in different lines.
  - Incorrect example

    ```
    create table1;create table2
    ```

  - Correct example

    ```
    create table1;
    create table2;
    ```

Create an ODPS SQL node

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over +Create and choose **MaxCompute > ODPS SQL**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **MaxCompute**, and then choose **Create > ODPS SQL**.

   > **Notice** The MaxCompute module is available on the DataStudio page only after you bind a MaxCompute compute engine to the current workspace on the **Project Management** page.

3. In the **Create Node** dialog box, specify **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.
5. Edit the code of the ODPS SQL node.

   Edit the code of the ODPS SQL node. The code must conform to the syntax. The following example creates a table, inserts data to the table, and queries data in the table:

   i. Create a table named test1.

   ```
   CREATE TABLE IF NOT EXISTS test1
   ( id BIGINT COMMENT '' ,
     name STRING COMMENT '' ,
     age BIGINT COMMENT '' ,
     sex STRING COMMENT '');
   ```

ii. Insert data to the table.

```
INSERT INTO test1 VALUES (1,'Zhang San',43,'Male');
INSERT INTO test1 VALUES (1,'Li Si',32,'Male');
INSERT INTO test1 VALUES (1,'Chen Xia',27,'Female');
INSERT INTO test1 VALUES (1,'Wang Wu',24,'Male');
INSERT INTO test1 VALUES (1,'Ma Jing',35,'Female');
INSERT INTO test1 VALUES (1,'Zhao Qian',22,'Female');
INSERT INTO test1 VALUES (1,'Zhou Zhuang',55,'Male');
```

iii. Query data in the table.

```
select * from test1;
```

iv. After you enter the preceding SQL statements in the code editor, click in the toolbar.

DataWorks executes your SQL statements from top to bottom and displays logs.

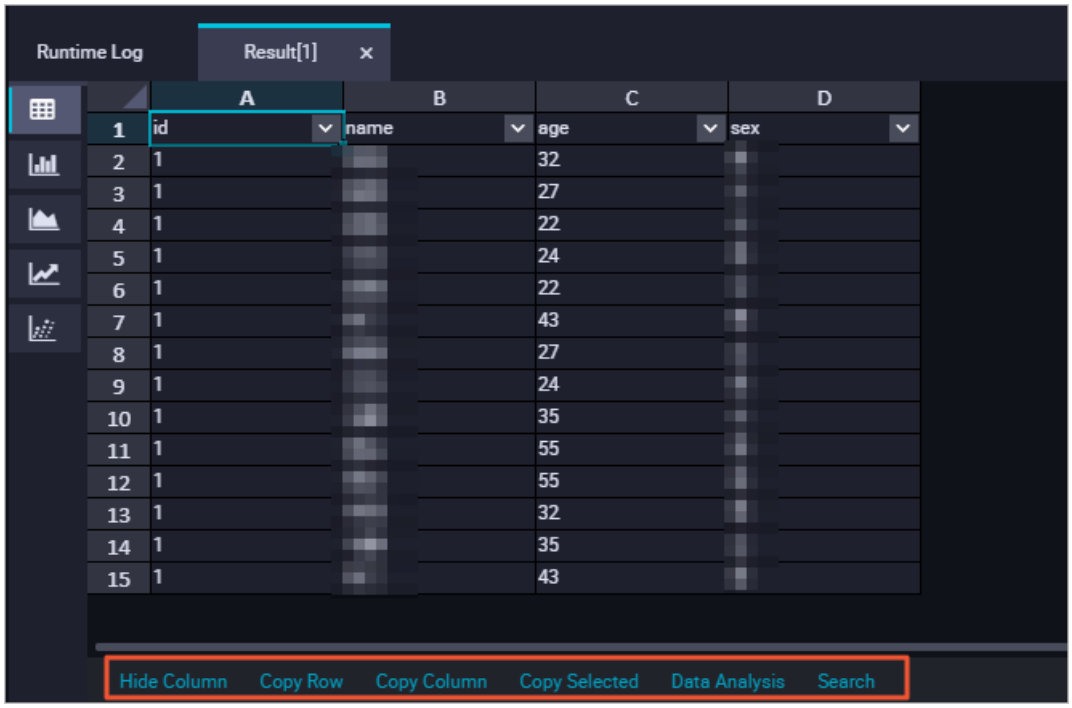
The `INSERT INTO` statement may result in unexpected data duplication. Although DataWorks does not re-execute the `INSERT INTO` statement, it may rerun corresponding nodes. We recommend that you avoid using the `INSERT INTO` statement. When DataWorks executes the INSERT INTO statement, the following information appears in logs:

```
The INSERT INTO statement in SQL may cause repeated data insertion. Although SQL-le
vel retries have been revoked for the INSERT INTO statement, task level retries may
still happen. We recommend that you avoid the use of the INSERT INTO statement.
If you continue to use INSERT INTO statements, we deem that you are aware of the as
sociated risks and are willing to take the consequences of potential data duplicati
on.
```

v. View the query result.

DataWorks displays the query result in a workbook.

You can view or manage the query result in the workbook, or copy the query result to a local Excel file.



Action	Description
Hide Column	Select one or more columns and click **Hide Column** at the bottom to hide the selected columns.
Copy Row	Select one or more rows and click **Copy Row** at the bottom to copy the selected rows.
Copy Column	Select one or more columns and click **Copy Column** at the bottom to copy the selected columns.
Copy Selected	Select one or more cells in the workbook and click **Copy Selected** at the bottom to copy the selected cells.
Data Analysis	Click **Data Analysis** at the bottom to go to the workbook editing page.
Search	Click **Search** at the bottom to search for data in the workbook. After you click the button, a search box appears in the upper-right corner of the Results tab.

6. On the configuration tab of the batch synchronization node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, configure parameters in the Schedule section. For more information, see Basic properties.
7. Commit the node.

> **Notice** You must set **Rerun** and **Parent Nodes** before you can commit the node.

i. Click in the toolbar.

ii. In the **Commit Node** dialog box, ignore the alert on mismatch between the input and output that you set with those detected in code lineage analysis, enter your comments in the **Description** field, and then select **I confirm to proceed with the commission**.

iii. Click **OK**.

In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the ODPS SQL node. For more information, see Publish nodes.

8. Test the node. For more information, see Manage auto triggered nodes.

4.5.2.2. Create an SQL Snippet node

SQL script templates are SQL templates that involve multiple input and output parameters. Each SQL script template involves one or more source tables. You can use an SQL script template to filter, join, or aggregate data in source tables.

Context

When a new version is released for a script template, you can decide whether to upgrade the version of the script template used in your nodes to the latest version.

The script template upgrade mechanism allows developers to continuously upgrade script template versions. This mechanism enhances the process execution efficiency and optimizes the business performance.

For example, User A uses V1.0 of a script template that belongs to User B. Then, User B releases V2.0 for the script template. User A receives a notification of the new version. After User A compares the code of the two versions, User A can decide whether to upgrade the script template to the latest version.

To upgrade an SQL script template, click **Update Code** and check whether the parameter configuration of the SQL script template is valid in the new version. Set parameters for the SQL script template of the new version based on the version description. Then, save the node and commit it for deployment.

Procedure

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over +Create and choose **MaxCompute > SQL Snippet**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **MaxCompute**, and then choose **Create > SQL Snippet**.

   > **Notice** The MaxCompute module is available on the DataStudio page only after you bind a MaxCompute compute engine to the current workspace on the **Project Management** page.

3. In the **Create Node** dialog box, specify **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.
5. On the node configuration tab, select a script template from the **Snippet** drop-down list.

   To improve development efficiency, you can create data analytics nodes by using the script templates provided by workspace members and tenants.

   - The script templates provided by members of the current workspace are available on the **Workspace-Specific** tab.
   - The script templates provided by tenants are available on the **Public** tab.
6. Click the **Parameters** tab in the right-side navigation pane and set parameters for the SQL script template.
7. On the configuration tab of the batch synchronization node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, configure parameters in the Schedule section. For more information, see Basic properties.
8. Commit the node.

   > **Notice** You can commit the node only after you specify the **Rerun** and **Parent Nodes** parameters.

   i. Click the icon in the toolbar.

   ii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

   iii. Click **OK**.

   In a workspace in standard mode, you must click **Publish** in the upper-right corner after you commit the node. For more information, see Publish nodes.
9. Test the node. For more information, see Manage auto triggered nodes.

4.5.2.3. Create an ODPS Spark node

DataWorks supports ODPS Spark nodes. This topic uses the JAR resource type as an example to describe how to create and configure an ODPS Spark node.

Create and upload a resource

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over +Create and choose **MaxCompute > Resource > JAR**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **MaxCompute**, and then choose **Create > Resource > JAR**.

   > **Notice** The MaxCompute module is available on the DataStudio page only after you bind a MaxCompute compute engine to the current workspace on the **Project Management** page.
3. In the **Create Resource** dialog box, set **Resource Name** and **Location**.
4. Click **Upload** and select the target file to upload.
5. Click **OK**.

Create an ODPS Spark node

1. On the **Data Analytics** tab, move the pointer over **+Create** and choose **MaxCompute > ODPS Spark**.

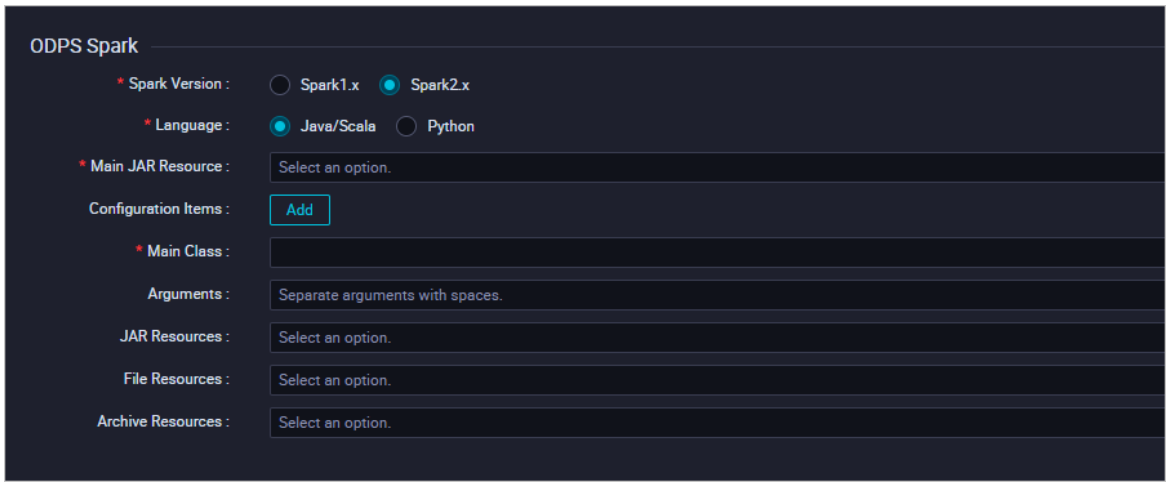
   Alternatively, you can click a workflow in the Business Flow section, right-click **MaxCompute**, and then choose **Create > ODPS Spark**.

2. In the **Create Node** dialog box, specify **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

3. Click **Commit**.

4. On the node configuration tab, set the parameters.



You can set **Spark Version** and **Language** as needed. The parameters vary with the value of the **Language** parameter. You can set the parameters as prompted.

The following table describes the parameters that appear after you set the Language parameter to **Java/Scala**.

Parameter	Description
Spark Version	The Spark version of the node. Valid values: **Spark1.x** and **Spark2.x**.
Language	The programming language of the node. Valid values: **Java/Scala** and **Python**. Select **Java/Scala**.
Main JAR Resource	The main JAR resource referenced by the node. Select a JAR resource that you uploaded from the drop-down list.
Configuration Items	The configuration items of the node. Click **Add** and set key and value to add a configuration item.
Main Class	The class name of the node.
Arguments	The parameter used to assign a value to a variable in the code during node scheduling. Separate multiple parameters with spaces.

Parameter	Description
JAR Resources	The JAR resource referenced by the node. Select a JAR resource that you uploaded from the drop-down list. The ODPS Spark node automatically finds the uploaded JAR resources based on the resource type.
File Resources	The file resource referenced by the node. Select a file resource that you uploaded from the drop-down list. The ODPS Spark node automatically finds the uploaded file resources based on the resource type.
Archive Resources	The archive resource referenced by the node. Select an archive resource that you uploaded from the drop-down list. The ODPS Spark node automatically finds the uploaded archive resources based on the resource type. Only compressed resources appear.

5. On the configuration tab of the batch synchronization node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, configure parameters in the Schedule section. For more information, see Basic properties.
6. Commit the node.

   > **Notice** You can commit the node only after you specify the **Rerun** and **Parent Nodes** parameters.

   i. Click the icon in the toolbar.

   ii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

   iii. Click **OK**.

   In a workspace in standard mode, you must click **Publish** in the upper-right corner after you commit the node. For more information, see Publish nodes.
7. Test the node. For more information, see Manage auto triggered nodes.

4.5.2.4. Create a PyODPS node

DataWorks supports PyODPS nodes, which are integrated with the Python SDK of MaxCompute. You can edit Python code in PyODPS nodes of DataWorks to process data in MaxCompute.

Context

You can also use the Python SDK of MaxCompute to process data in MaxCompute.

> **Note**
> - The Python version of PyODPS nodes is 2.7.
> - Each PyODPS node can process a maximum of 50 MB data and can occupy a maximum of 1 GB memory. Otherwise, DataWorks terminates the PyODPS node. Avoid writing too much data processing code for a PyODPS node.

PyODPS nodes are designed to use the Python SDK of MaxCompute. If you want to run pure Python code, you can create a Shell node to run the Python scripts uploaded to DataWorks.

Create a PyODPS node

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over +Create and choose **MaxCompute > PyODPS**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **MaxCompute**, and then choose **Create > PyODPS**.

   > **Notice** The MaxCompute module is available on the DataStudio page only after you bind a MaxCompute compute engine to the current workspace on the **Project Management** page.

3. In the **Create Node** dialog box, specify **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.
5. Edit the code of the PyODPS node on the node configuration tab.
   i. Use the MaxCompute entry.

   Each PyODPS node includes the global variable odps or o, which is the MaxCompute entry. Therefore, you do not need to manually specify the MaxCompute entry.

   ```
   print(odps.exist_table('PyODPS_iris'))
   ```

   ii. Run SQL statements.

   In PyODPS nodes, you can execute MaxCompute SQL statements to query data and obtain the query results. You can use the execute_sql or run_sql method to run MaxCompute job instances.

   To execute statements that are not directly compatible with the MaxCompute console, you can use some methods. For example, you cannot directly execute statements other than DDL and DML in the MaxCompute console.

   To execute a GRANT or REVOKE statement, use the run_security_query method. To run a PAI command, use the run_xflow or execute_xflow method.

   ```
   o.execute_sql('select * from dual')  # Execute the statement in synchronous mode. O
   ther nodes are blocked until the SQL statement is executed.
   instance = o.run_sql('select * from dual') # Execute the statement in asynchronous
   mode.
   print(instance.get_logview_address()) # Obtain the Logview URL of an instance.
   instance.wait_for_success() # Other nodes are blocked until the SQL statement is ex
   ecuted.
   ```

iii. Set runtime parameters.

You can use the hints parameter to set the runtime parameters. The type of the hints parameter is DICT.

```
o.execute_sql('select * from PyODPS_iris', hints={'odps.sql.mapper.split.size': 16}
)
```

If you set the sql.settings parameter for the global configuration, you must set the runtime parameters each time you run the code.

```
from odps import options
options.sql.settings = {'odps.sql.mapper.split.size': 16}
o.execute_sql('select * from PyODPS_iris')  # The hints parameter is automatically
set based on the global configuration.
```

iv. Obtain SQL query results.

You can use the open_reader method to obtain query results in the following scenarios:

- The SQL statement returns structured data.

```
with o.execute_sql('select * from dual').open_reader() as reader:
for record in reader:  # Process each record.
```

- SQL statements such as DESC are executed. In this case, you can use the reader.raw property to obtain raw query results.

```
with o.execute_sql('desc dual').open_reader() as reader:
print(reader.raw)
```

> **Note** If you use a custom time variable, you must fix the variable to a time. PyODPS nodes do not support relative time variables.

6. On the configuration tab of the batch synchronization node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, configure parameters in the Schedule section. For more information, see Basic properties.
7. Commit the node.

> **Notice** You can commit the node only after you specify the **Rerun** and **Parent Nodes** parameters.

i. Click the icon in the toolbar.

ii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

iii. Click **OK**.

In a workspace in standard mode, you must click **Publish** in the upper-right corner after you commit the node. For more information, see Publish nodes.

8. Test the node. For more information, see Manage auto triggered nodes.

Built-in modules for PyODPS nodes

A PyODPS node contains the following built-in modules:

- setuptools
- cython
- psutil
- pytz
- dateutil
- requests
- pyDes
- numpy
- pandas
- scipy
- scikit_learn
- greenlet
- six
- Other built-in modules in Python 2.7, such as smtplib

4.5.2.5. Create an ODPS Script node

You can create an ODPS Script node to develop an SQL script by using the SQL engine provided by MaxCompute V2.0.

Context

The ODPS Script node allows DataWorks to compile the SQL script as a whole, instead of compiling the SQL statements in the script one by one. In this way, the SQL script is committed and run as a whole. This guarantees that an execution plan is only queued and executed once, making full use of MaxCompute computing resources.

Create an ODPS Script node

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over +Create and choose **MaxCompute > ODPS Script**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **MaxCompute**, and then choose **Create > ODPS Script**.

   > **Notice** The MaxCompute module is available on the DataStudio page only after you bind a MaxCompute compute engine to the current workspace on the **Project Management** page.

3. In the **Create Node** dialog box, specify **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.
5. Edit the SQL script of the ODPS Script node as required.

6. On the configuration tab of the batch synchronization node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, configure parameters in the Schedule section. For more information, see Basic properties.
7. Commit the node.

   > **Notice** You can commit the node only after you specify the **Rerun** and **Parent Nodes** parameters.

   i. Click the icon in the toolbar.

   ii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

   iii. Click **OK**.

   In a workspace in standard mode, you must click **Publish** in the upper-right corner after you commit the node. For more information, see Publish nodes.
8. Test the node. For more information, see Manage auto triggered nodes.

SQL syntax and limits for ODPS Script nodes

Write SQL statements based on your business logic in a way similar to that of using a common programming language. You do not need to consider how to organize the SQL statements.

```
-- SET statements
set odps.sql.type.system.odps2=true;
[set odps.stage.reducer.num=***;]
[...]
-- DDL statements
create table table1 xxx;
[create table table2 xxx;]
[...]
-- DML statements
@var1 := SELECT [ALL | DISTINCT] select_expr, select_expr, ...
    FROM table3
    [WHERE where_condition];
@var2 := SELECT [ALL | DISTINCT] select_expr, select_expr, ...
    FROM table4
    [WHERE where_condition];
@var3 := SELECT [ALL | DISTINCT] var1.select_expr, var2.select_expr, ...
    FROM @var1 join @var2 on ... ;
INSERT OVERWRITE|INTO TABLE [PARTITION (partcol1=val1, partcol2=val2 ...)]
    SELECT [ALL | DISTINCT] select_expr, select_expr, ...
    FROM @var3;
[@var4 := SELECT [ALL | DISTINCT] var1.select_expr, var.select_expr, ... FROM @var1
    UNION ALL | UNION
    SELECT [ALL | DISTINCT] var1.select_expr, var.select_expr, ... FROM @var2;
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] table_name
    AS
    SELECT [ALL | DISTINCT] select_expr, select_expr, ...
    FROM var4;]
```

SQL syntax and limits for ODPS Script nodes

- ODPS Script nodes support SET statements, DML statements, and some DDL statements. The DDL

statements used to return data, such as DESC and SHOW statements, are not supported.

- A complete script consists of SET statements, DDL statements, and DML statements in sequence. You can write one or more statements of each type, or even skip a type without writing any statements of that type. However, you cannot mix different types of statements together. You must strictly follow the sequence of SET statements > DDL statements > DML statements.
- The at signs (@) residing before some statements indicate that these statements are connected by using variables.
- A script supports only one statement that returns data, such as an independent SELECT statement. If multiple such statements are provided, an error occurs. We recommend that you do not use SELECT statements in a script.
- A script supports only one `CREATE TABLE AS` statement, which must be the last statement. We recommend that you put CREATE TABLE statements and INSERT statements in different sections to separate them.
- If one statement in a script fails, the whole script fails.
- A job is generated to process data only after all the input data is prepared for a script.
- If a script writes data to a table and then reads the table, an error occurs. For example, an error occurs for the following statements:

  ```
  insert overwrite table src2 select * from src where key > 0;
  @a := select * from src2;
  select * from @a;
  ```

  To avoid the error, modify the statements to the following:

  ```
  @a := select * from src where key > 0;
  insert overwrite table src2 select * from @a;
  select * from @a;
  ```

Sample script:

```
create table if not exists dest(key string , value bigint) partitioned by (d string);
create table if not exists dest2(key string,value bigint ) partitioned by (d string);
@a := select * from src where value >0;
@b := select * from src2 where key is not null;
@c := select * from src3 where value is not null;
@d := select a.key,b.value from @a left outer join @b on a.key=b.key and b.value>0;
@e := select a.key,c.value from @a inner join @c on a.key=c.key;
@f := select * from @d union select * from @e union select * from @a;
insert overwrite table dest partition (d='20171111') select * from @f;
@g := select e.key,c.value  from @e join @c on e.key=c.key;
insert overwrite table dest2 partition (d='20171111') SELECT * from @g;
```

Scenarios of ODPS Script nodes

- You can use an ODPS Script node to rewrite a single statement with nested subqueries, or a script that must be split into multiple statements due to its complexity.
- Data from different data stores may be prepared at different time points, and the time difference may be large. For example, the data from one data store can be prepared at 01:00, whereas that from another data store can be prepared at 07:00. In this case, table variables are not suitable for connecting statements. You can use an ODPS Script node to combine the statements to a script.

4.5.2.6. Create an ODPS MR node

MaxCompute supports the MapReduce API. You can create and commit ODPS MR nodes that call the Java API operations of MapReduce to develop MapReduce programs for processing data in MaxCompute.

Context

Before you create ODPS MR nodes, you must upload, commit, and then deploy required resources.

Create a resource

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over Create and choose **MaxCompute > Resource > JAR**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **MaxCompute**, and then choose **Create > Resource > JAR**.

   > **Notice** The MaxCompute module is available on the DataStudio page only after you bind a MaxCompute compute engine to the current workspace on the **Project Management** page.

3. In the **Create Resource** dialog box, set **Resource Name** and **Location**.

   > **Note**
   > - The resource name can be different from the name of the uploaded file.
   > - Convention for naming resources: A resource name can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive and must be 1 to 128 characters in length. A JAR resource name must end with .jar. A Python resource name must end with .py.

4. Click **Upload** and select the target file to upload.
5. Click **OK**.
6. Click in the toolbar to commit the resource to the development environment.

Create an ODPS MR node

1. On the **Data Analytics** tab, find the target workflow, right-click **MaxCompute**, and then choose **Create > ODPS MR**.
2. In the **Create Node** dialog box, specify **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

3. Click **Commit**.
4. Edit the ODPS MR node.

```
-- Create an input table.
CREATE TABLE if not exists jingyan_wc_in (key STRING, value STRING);
-- Create an output table.
CREATE TABLE if not exists jingyan_wc_out (key STRING, cnt BIGINT);
    --- Create the dual table.
    drop table if exists dual;
    create table dual(id bigint); -- Create the dual table if no dual table exists in t
he current workspace and initialize the table.
    --- Initialize the dual table.
    insert overwrite table dual select count(*)from dual;
    --- Insert the sample data to the wc_in table.
    insert overwrite table jingyan_wc_in select * from (
    select 'project','val_pro' from dual
    union all
    select 'problem','val_pro' from dual
    union all
    select 'package','val_a' from dual
    union all
    select 'pad','val_a' from dual
      ) b;
-- Reference the uploaded JAR package. You can find the JAR package in the resource lis
t, right-click the JAR resource, and select Insert Resource Path.
--@resource_reference{"mapreduce-examples.jar"}
jar -resources mapreduce-examples.jar -classpath ./mapreduce-examples.jar com.aliyun.od
ps.mapred.open.example.WordCount jingyan_wc_in jingyan_wc_out
```

Pay attention to the following information when you write the code:

- `--@resource_reference` : references a resource. Find the target resource, right-click it, and then select **Insert Resource Path** to generate the reference statement.
- `-resources` : the name of the referenced JAR resource.
- `-classpath` : the path of the JAR resource. You can enter ./Resource name because the resource has been referenced.
- `com.aliyun.odps.mapred.open.example.WordCount` : the main class in the JAR resource to be called during node running. It must be the same as the main class name in the JAR resource.
- `jingyan_wc_in` : the name of the input table of the ODPS MR node. The input table is created in the preceding code.
- `jingyan_wc_out` : the name of the output table of the ODPS MR node. The output table is created in the preceding code.
- If you use multiple JAR resources in a single ODPS MR node, separate the resource paths with commas (,), for example, `-classpath ./xxxx1.jar,./xxxx2.jar` .

5. On the configuration tab of the batch synchronization node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, configure parameters in the Schedule section. For more information, see Basic properties.
6. Commit the node.

> **Notice** You can commit the node only after you specify the **Rerun** and **Parent Nodes** parameters.

i. Click the icon in the toolbar.

ii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

iii. Click **OK**.

In a workspace in standard mode, you must click **Publish** in the upper-right corner after you commit the node. For more information, see Publish nodes.

7. Test the node. For more information, see Manage auto triggered nodes.

4.5.2.7. Create a MaxCompute table

This topic describes how to create a MaxCompute table.

Procedure

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over +Create and choose **MaxCompute > Table**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **MaxCompute**, and then choose **Create > Table**.

   > **Notice** The MaxCompute module is available on the DataStudio page only after you bind a MaxCompute compute engine to the current workspace on the **Project Management** page.

3. In the **Create Table** dialog box, set **Table Name** and click **Commit**.

   > **Notice** A table name can be up to 64 characters in length. The table name must start with a letter and cannot contain Chinese or special characters.

4. On the table configuration tab that appears, set the parameters in the **General** section.

Parameter or button	Description
Display Name	The display name of the table.
Level 1 Folder	The name of the level-1 folder where the table resides. **Note** Level-1 and level-2 folders only show the table locations in DataWorks so that you can better manage tables.
Level 2 Folder	The name of the level-2 folder where the table resides.
Create Folder	Goes to the **Folder Management** tab. On this tab, you can create level-1 and level-2 folders for tables.

Parameter or button	Description
Description	The description of the table.

5. Create a table.

   Use one of the following methods to create a table:

   - Create a table by using DDL statements.

     Click **DDL Statement** in the top navigation bar. In the dialog box that appears, enter the statements for creating a table.

     After you finish editing the statements, click **Generate Table Schema**. Information is automatically entered in the General, Physical Model, and Schema sections.

   - Create a table on the graphical user interface (GUI).

     If DDL statements are inappropriate for you to create a table, try to use the GUI. The following table describes the relevant parameters for creating a table on the GUI.

Section	Parameter or button	Description
Physical Model	**Partitioning**	Specifies whether the table is partitioned. Valid values: **Partitioned Table** and **Non-Partitioned Table**.
	Time-to-Live	The time-to-live of data in MaxCompute. If you select this check box, you must enter a number in the **TTL** field. If the table or partition is stored for more than the specified number of days, data that has not been updated is cleared.
	Table Level	The level of the table. Generally, tables are divided into operation data store (ODS), common data model (CDM), and application data service (ADS) levels. You can specify a custom level name.
	Categories	The category of the table. Tables are categorized into basic services, advanced services, and other services. You can specify a custom category name. If you want to create a table category or level, click **Create Level** to go to the **Level Management** tab. **Note** Categories are designed only for your management convenience and do not involve underlying implementation.
	Table Type	The type of the table. Default value: **Internal Table**.
	Field Name	The name of the field. The name can contain letters, digits, and underscores (_).

Section	Parameter or button	Description
Schema	**Display Name**	The display name of the field.
	Data Type	The data type of the field.
	Definition or Maximum Value Length	The maximum value length of a field. You can set a maximum value length only for fields of the DECIMAL, VARCHAR, ARRAY, MAP, and STRUCT types.
	Description	The description of the field.
	Primary Key Field	Specifies whether the field serves as the primary key or part of a composite primary key.
	Create Field	Adds a field to the table.
	Delete Field	Deletes a field from the table. **Note** If you delete a field from an existing table, DataWorks requests you to delete the table and create another table with the same name. This operation is forbidden in the production environment.
	Move Up	Adjusts the field sequence of the table. If you adjust the sequence of fields in an existing table, DataWorks requests you to delete the table and create another table with the same name. This operation is forbidden in the production environment.
	Move Down	The description is the same as that of the **Move Up** operation.
	Add	Adds a partition to the table. If you add a partition to an existing table, DataWorks requests you to delete the table and create another table with the same name. This operation is forbidden in the production environment.
	Delete	Deletes a partition from the table. If you delete a partition from an existing table, DataWorks requests you to delete the table and create another table with the same name. This operation is forbidden in the production environment.
	Actions	Commits a partition or deletes a field.

Section	Parameter or button	Description
Partition Field Design Note This section is available only when Partitioning under Physical Model is set to Partitioned Table.	**Add**	Adds a partition field.
	Data Type	The data type of the partition field. We recommend that you use the STRING type for all partition fields.
	Length	The maximum length of the partition field. You can set the maximum length only for fields of the VARCHAR-type.
	Description	The description of the partition field.
	Partition Column Date Format	The format of the date partition. If the partition field is a date, although the data type may be STRING, select or enter a date format, such as *yyyymmmdd* or *yyyy-mm-dd*.
	Partition Column Date Granularity	The granularity of the date partition. The granularities can be second, minute, hour, day, month, quarter, and year. You can enter a partition granularity as required. If you want to specify multiple partition granularities, note that a greater granularity corresponds to a higher partition level. For example, three partitions whose granularities are day, hour, and month, respectively, are available. Multi-level partitions are in the hierarchical order of level-1 partition (month), level-2 partition (day), and level-3 partition (hour).

6. Click **Commit in Development Environment** and **Commit to Production Environment** in sequence.

   If you are using a workspace in basic mode, you only need to click **Commit to Production Environment**.

Button	Description
Load from Development Environment	If the table has been committed to the development environment, the button is clickable. After you click the button, the information about the table you create in the development environment overwrites the table information on the current page. Note This feature is supported only for MaxCompute tables.
Commit in Development Environment	Before you click the button, make sure that you have filled in all required parameters on the table configuration tab. Do not click the button if any parameters are not specified.

Button	Description
Load from Production Environment	After you click the button, the information about the table that is committed to the production environment overwrites the table information on the current page. **Note** This feature is supported only for MaxCompute tables.
Commit to Production Environment	After you click the button, the table is created in the workspace of the production environment.

What's next

After the table is created, you can query the table data and modify or delete the table. For more information, see Manage tables.

4.5.2.8. Create, reference, and download resources

This topic describes how to create, reference, and download JAR and Python resources.

Context

If your code or function requires resource files such as.jar files, you can upload resources to your workspace and reference them.

If the existing built-in functions do not meet your requirements, DataWorks allows you to create user-defined functions (UDFs) and customize processing logic. You can upload the required JAR packages to your workspace so that you can reference them when you create UDFs.

Note

- You can view built-in functions on the **Built-In Functions** tab. For more information, see View built-in functions.
- You can view the UDFs that you have committed or deployed on the **MaxCompute Functions** tab.

The resources that you can upload to MaxCompute include text files, MaxCompute tables, Python code, and compressed packages in the .zip, .tgz, .tar.gz, .tar, and .jar formats. You can read or use these resources when you run UDFs or MapReduce.

MaxCompute provides API operations for you to read and use resources. The following types of MaxCompute resources are available:

- **Python**: the Python code you have written. You can use Python code to register Python UDFs.
- **JAR**: the compiled Java JAR packages.
- **Archive**: the compressed files that can be identified by the file name extension. Supported file types include .zip, .tgz, .tar.gz, .tar, and .jar.
- **File**: files in the .zip, .so, or .jar format.

JAR resources and **file** resources have the following differences:

- To create a JAR resource, write Java code in the offline Java environment, compress the code to a JAR package, and upload the package as a JAR resource to DataWorks.

- To create a file resource that is smaller than or equal to 500 KB in size, you can create and edit it in the DataWorks console.
- To create a file resource that is larger than 500 KB in size, select **Large File (more than 500 KB)** and click Upload to upload the file.

> **Note** Each resource file to be uploaded in the DataWorks console cannot exceed 30 MB.

Create a JAR resource

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over +Create and choose **MaxCompute > Resource > JAR**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **MaxCompute**, and then choose **Create > Resource > Python**.

   > **Notice** The MaxCompute module is available on the DataStudio page only after you bind a MaxCompute compute engine to the current workspace on the **Project Management** page.

3. In the **Create Resource** dialog box, set **Resource Name** and **Location**.

   > **Note**
   > - The resource name can be different from the name of the uploaded file.
   > - A resource name can contain letters, digits, underscores (_), and periods (.), and is not case-sensitive. It must be 1 to 128 characters in length. A JAR resource name must end with .jar, and a Python resource name must end with .py.

4. Click **Upload** and select the target file to upload.
5. Click **OK**.
6. Click in the toolbar to commit the resource to the development environment.

Create a Python resource and register a UDF

1. Create a Python resource.
   i. On the **Data Analytics** tab, move the pointer over +Create and choose **MaxCompute > Resource > Python**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **MaxCompute**, and then choose **Create > Resource > Python**.

   ii. In the **Create Resource** dialog box, set **Resource Name** and **Location**.

   iii. Click **OK**.

iv. On the configuration tab that appears, edit the code of the created resource. Sample code:

```
from odps.udf import annotate
@annotate("string->bigint")
class ipint(object):
    def evaluate(self, ip):
        try:
            return reduce(lambda x, y: (x << 8) + y, map(int, ip.split('.')))
        except:
            return 0
```

v. Click  in the toolbar.

2. Register a UDF.

i. On the **Data Analytics** tab, move the pointer over  and choose **MaxCompute > Function**.

Alternatively, you can click a workflow in the Business Flow section, right-click **MaxCompute**, and then choose **Create > Function**.

ii. In the **Create Function** dialog box, set **Function Name** and **Location**.

iii. Click **Commit**.

iv. In the **Register Function** section of the configuration tab that appears, enter the class name and the name of the Python resource that has been created, and then click in the toolbar. In this example, the class name is `ipint.ipint`.

v. Check whether the ipint function is valid and meets your expectation. For example, you can create an ODPS SQL node to test the ipint function by running an SQL statement.

Reference and download resources

- For more information about how to reference resources for functions, see Register a UDF.
- For more information about how to reference resources for nodes, see Create an ODPS MR node.

To download a resource, double-click **Resource** under the target workflow. In the resource list that appears, move the pointer over the required resource and click **Download**.

4.5.2.9. Register a UDF

DataWorks allows you to develop UDFs in Python and Java. This topic describes how to register a UDF.

Prerequisites

Before you register a UDF, you must upload the related resource.

Procedure

1. Log on to the DataWorks console.
2. Create a workflow. For more information, see Create a workflow.
3. Write Java code in the offline Java environment, compress the code to a JAR package, and upload the package as a JAR resource to DataWorks. For more information, see Create a JAR resource.
4. Create a UDF.

i. Find the target workflow, right-click **MaxCompute**, and then choose **Create > Function**.

ii. In the **Create Function** dialog box, set **Function Name** and **Location** and click **OK**.

iii. In the **Register Function** section of the configuration tab that appears, set the parameters.

Parameter	Description
Function Type	The type of the function. Valid values: **Mathematical Function**, **Aggregate Function**, **String Function**, **Date Function**, **Analytic Function**, and **Other**.
Engine Instance MaxCompute	The MaxCompute engine instance bound to the current workspace. By default, you cannot change the engine instance.
Function Name	The name of the function, which is used to reference the function in SQL. The function name must be globally unique and cannot be modified after the function is registered.
Owner	The owner of the function. By default, this parameter is automatically set.
Class Name	Required. The name of the class for implementing the function. **Note** If the resource type is Python, enter the class name in the Python resource name.Class name format. Do not include the .py extension in the resource name.
Resources	Required. The list of resources. You can search for existing resources in the current workspace in fuzzy match mode.
Description	The description of the function.
Expression Syntax	The instructions on how to use the function, for example, `test`.
Parameter Description	The description of supported input and output parameter types.
Return Value	Optional. The value to return. Example: 1.
Example	Optional. An example of the function.

5. Click in the toolbar.

6. Commit the function.

i. Click in the toolbar.

ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.

iii. Click **OK**.

4.5.3. EMR

4.5.3.1. Modes for associating an EMR cluster with a DataWorks workspace

After you associate an EMR cluster with a DataWorks workspace, you can create nodes such as EMR Hive, EMR MR, EMR Presto, and EMR Spark SQL nodes based on an EMR compute engine and configure EMR workflows. You can also schedule the nodes and manage metadata. This improves your data output.

DataWorks provides two modes for you to associate an EMR cluster with a workspace: **Shortcut mode** and **Security mode**. The two modes can meet the security requirements of various enterprises. If you associate an EMR cluster with a workspace by using the **Shortcut mode**, you can create and run EMR nodes to generate data. If you associate an EMR cluster with a workspace by using the **Security mode**, you can create and run EMR nodes to generate data and manage permissions on the data to ensure higher security.

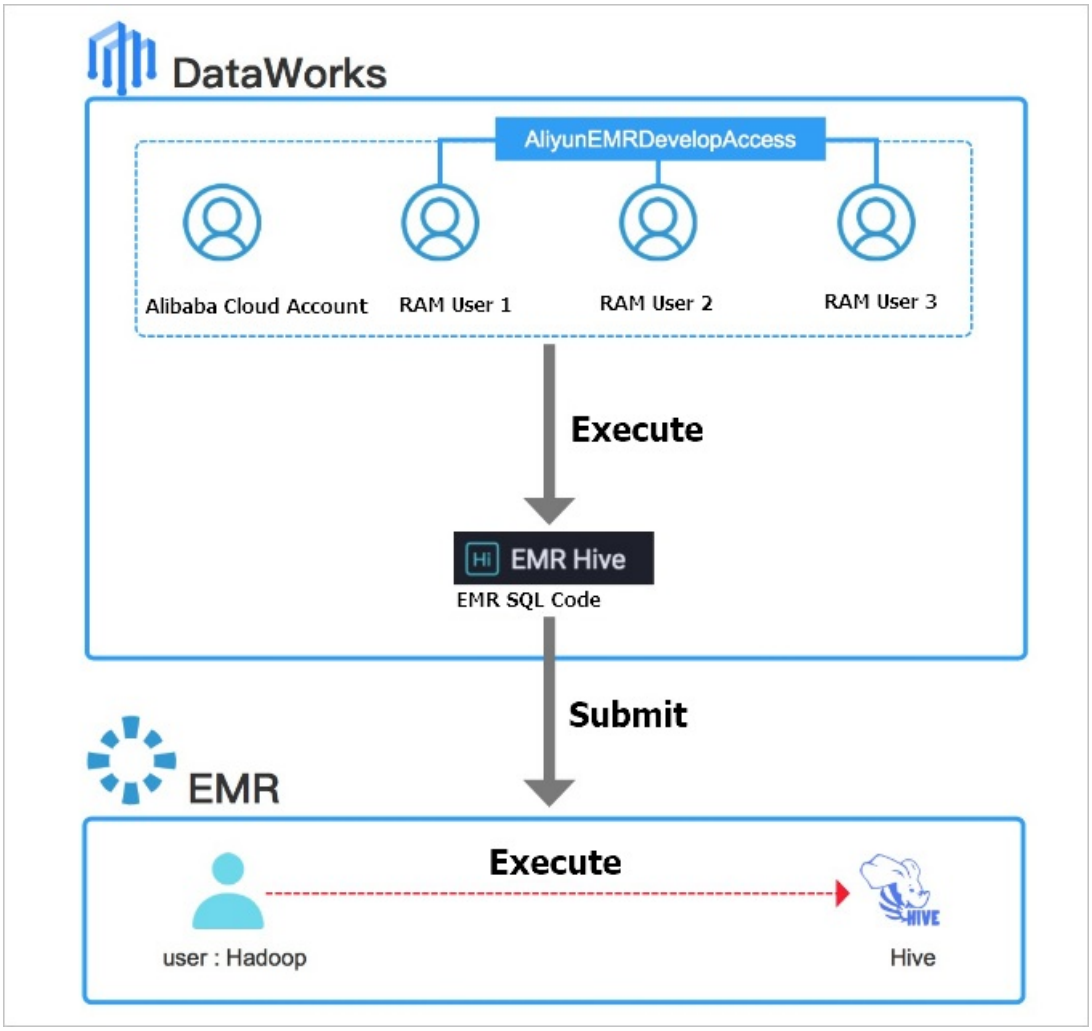
Shortcut mode

In **Shortcut mode**, if you run or schedule EMR nodes in DataWorks by using your Apsara Stack tenant account or as a RAM user, the code is committed to the EMR cluster and run by the Hadoop user of the EMR cluster.

> Notice

- The Hadoop user has all the permissions on the EMR cluster. Proceed with caution when you use the Shortcut mode to associate an EMR cluster with a workspace.
- Before you use the **Shortcut mode** to associate an EMR cluster with a workspace, you must attach the AliyunEMRDevelopAccess policy to workspace roles such as developers and administrators. This way, the roles can be used to create and run EMR nodes in DataStudio.
  - The AliyunEMRDevelopAccess policy is attached to Apsara Stack tenant accounts by default.
  - To run EMR nodes as a RAM user, you must attach the AliyunEMRDevelopAccess policy to the RAM user.

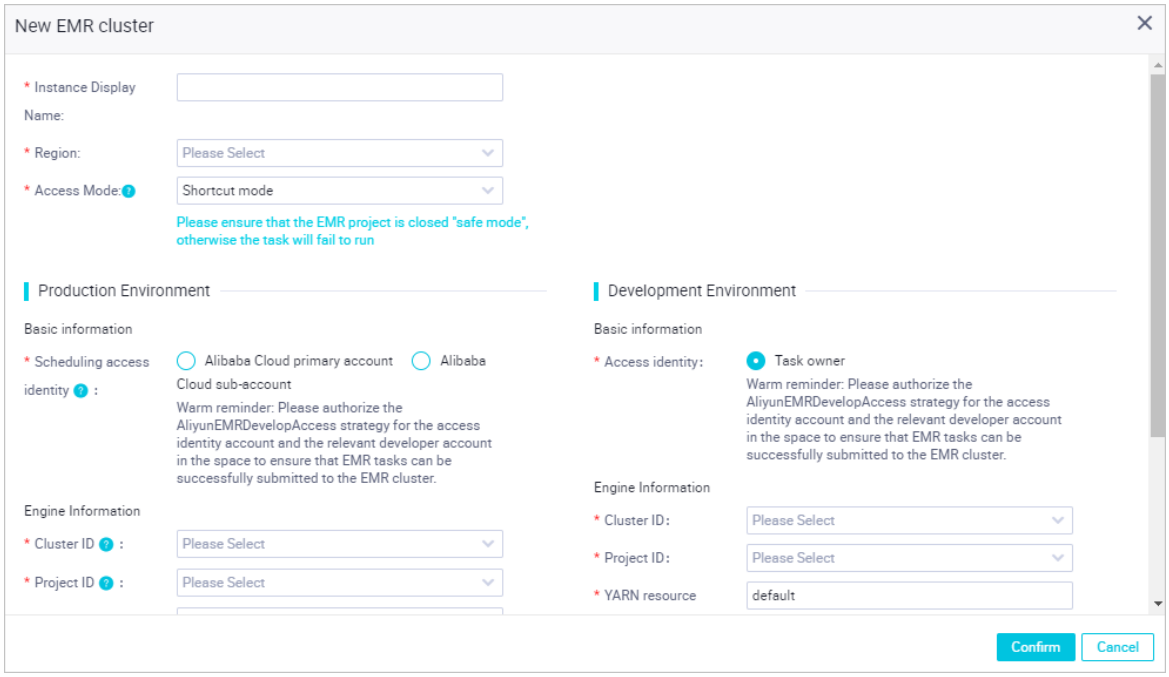
The **Shortcut mode** is suitable for workspaces that do not require strict permission management for users who run nodes.



To associate an EMR cluster with a workspace in **Shortcut mode**, perform the following steps:

1. Log on to the DataWorks console.
2. In the upper-right corner of the page, click the icon to go to the Workspace Management page.
3. In the **Compute Engine Information** section, click the **E-MapReduce** tab.
4. On the **E-MapReduce** tab, click **Add Instance**.
5. In the **New EMR cluster** dialog box, set the parameters.

Parameters in the New EMR cluster dialog box vary based on the mode in which your DataWorks workspace runs. The following table describes the parameters for a DataWorks workspace in standard mode. You must set the parameters for both the production environment and the development environment.



Parameter	Description
Instance Display Name	The display name of the EMR compute engine instance.
Region	The region of the current workspace.
Access Mode	The access mode of the EMR cluster. Select **Shortcut mode** from the drop-down list.
Scheduling access identity	The identity that is used to commit the code of an EMR node to the EMR cluster. The code is committed when the node is committed to the scheduling system of DataWorks in the production environment. Valid values: **Alibaba Cloud primary account** and **Alibaba Cloud sub-account**. **Note** ○ This parameter is available only for the production environment. ○ Before you use the **Shortcut mode** to associate an EMR cluster with a workspace, you must attach the AliyunEMRDevelopAccess policy to workspace roles such as developers and administrators. This way, the roles can be used to create and run EMR nodes in DataStudio.

Parameter	Description
Access identity	The identity that is used to commit the code of an EMR node in the development environment to the EMR cluster. Default value: **Task owner**. Note ◦ This parameter is available only for the development environment of a workspace in standard mode. ◦ **Task owner** can be an Apsara Stack tenant account or a RAM user. Before you use the **Shortcut mode** to associate an EMR cluster with a workspace, you must attach the AliyunEMRDevelopAccess policy to workspace roles such as developers and administrators. This way, the roles can be used to create and run EMR nodes in DataStudio.
Cluster ID	The ID of the EMR cluster that you want to associate with the workspace. Select an ID from the drop-down list. The EMR cluster with the selected ID is used as the runtime environment of EMR nodes.
Project ID	The ID of the EMR project that you want to associate with the workspace. Select an ID from the drop-down list. The selected EMR project with the selected ID is used as the runtime environment of EMR nodes. Note The IDs of the EMR projects in **Security mode** are not displayed and cannot be selected.
YARN resource queue	The name of the resource queue in the EMR cluster. Unless otherwise specified, set this parameter to *default*.
Endpoint	The endpoint of the EMR cluster. Unless otherwise specified, set this parameter to *default*.

6. Click **Confirm**.

Security mode

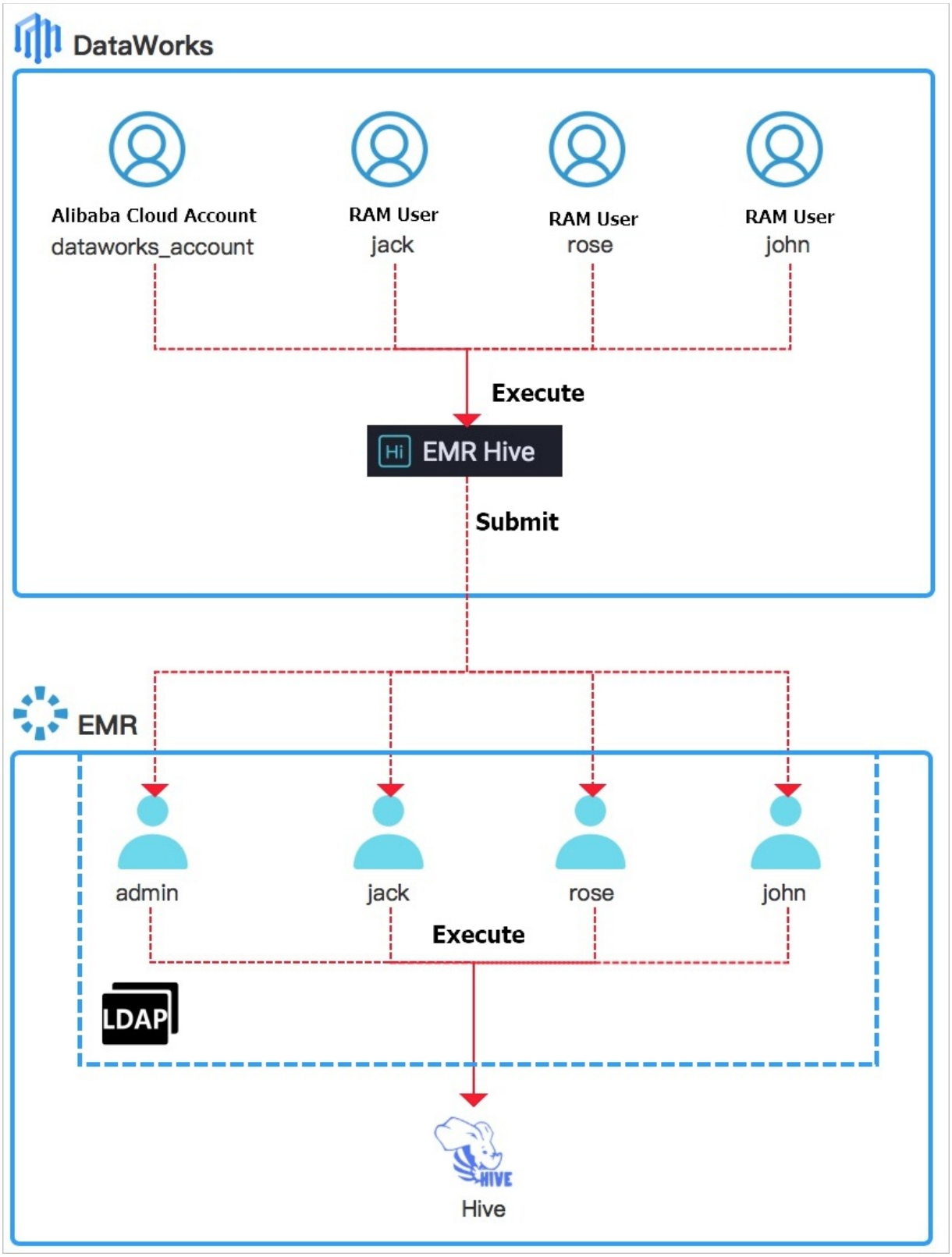
In **Security mode**, if you commit the code of EMR nodes by using an Apsara Stack tenant account or as a RAM user to an EMR cluster, the code is run by a user that has the same name as the Apsara Stack tenant account or RAM user. EMR Ranger can be used to manage the permissions of each Hadoop user in the EMR cluster. This ensures that different Apsara Stack tenant accounts, node owners, or RAM users have different data permissions when they run EMR nodes in DataWorks. This provides higher data security.

Note

Before you use the **Security mode** to associate an EMR cluster with a workspace, you must add the credentials of workspace roles such as developers and administrators to the Lightweight Directory Access Protocol (LDAP) directory of the EMR cluster. In addition, you must attach the AliyunEMRDevelopAccess or AliyunEMRFullAccess policy and grant relevant data permissions to the workspace roles. This way, the roles can be used to create and run EMR nodes in DataStudio.

- The credentials of Apsara Stack tenant accounts are in the LDAP directory of the EMR cluster by default. The AliyunEMRDevelopAccess and AliyunEMRFullAccess policies are also attached to Apsara Stack tenant accounts by default.
- To run EMR nodes as a RAM user, you must add the credential of the RAM user to the LDAP directory of the EMR cluster. For more information, see the *Add the credentials of specific RAM users to the LDAP directory of the EMR cluster* step. In addition, you must attach the AliyunEMRDevelopAccess or AliyunEMRFullAccess policy to the RAM user.

The **Security mode** is suitable for workspaces that require strict management and isolation of data permissions for users who run nodes.



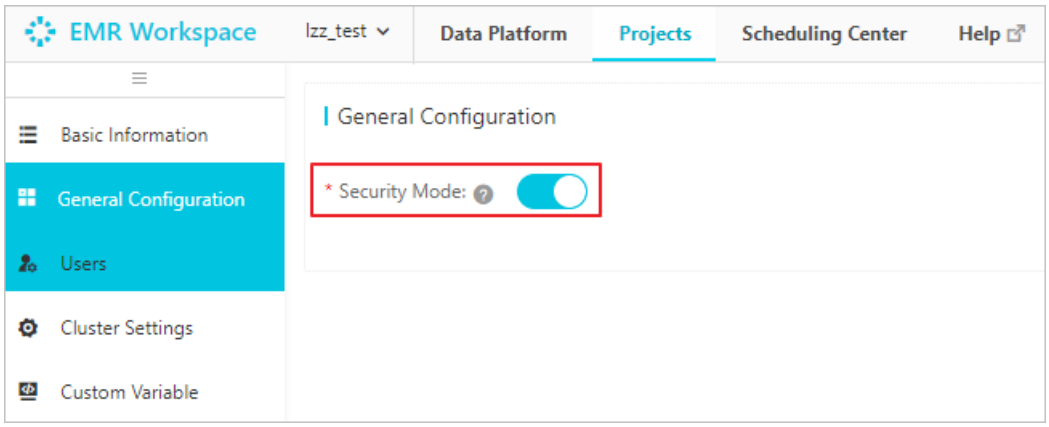
To use the **Security mode** to associate an EMR cluster with a workspace, perform the following steps:

1. Turn on Security Mode for the EMR project.
   i. Log on to the EMR console.
   ii. In the top navigation bar, click **Data Platform**.

iii. In the **Projects** section, find the project for which you want to enable the Security mode and click **Edit Job** in the Actions column.

iv. On the page that appears, click the **Projects** tab in the top navigation bar.

v. In the left-side navigation pane, click **General Configuration**. On the General Configuration page, turn on **Security Mode**.



2. Add the credentials of specific RAM users to the LDAP directory of the EMR cluster.

i. Go back to the homepage of the EMR console. In the top navigation bar, click **Cluster Management**.

ii. Find the cluster that you want to manage and click **Details** in the Actions column.

iii. In the left-side navigation pane, click **Users**.

iv. On the **Users** page, click **Add User**.

v. In the **Add User** dialog box, set the parameters.

We recommend that you add the credentials of the following RAM users to the LDAP directory of the EMR cluster:

- RAM users that create, test, and run EMR nodes in DataStudio
- RAM users that create, commit, and deploy EMR nodes in DataStudio

vi. Click **OK**.

3. Configure EMR Ranger and manage the permissions of the Hadoop users that correspond to your Apsara Stack tenant account and RAM users.

4. Associate the EMR cluster with the current DataWorks workspace.

i. Log on to the DataWorks console.

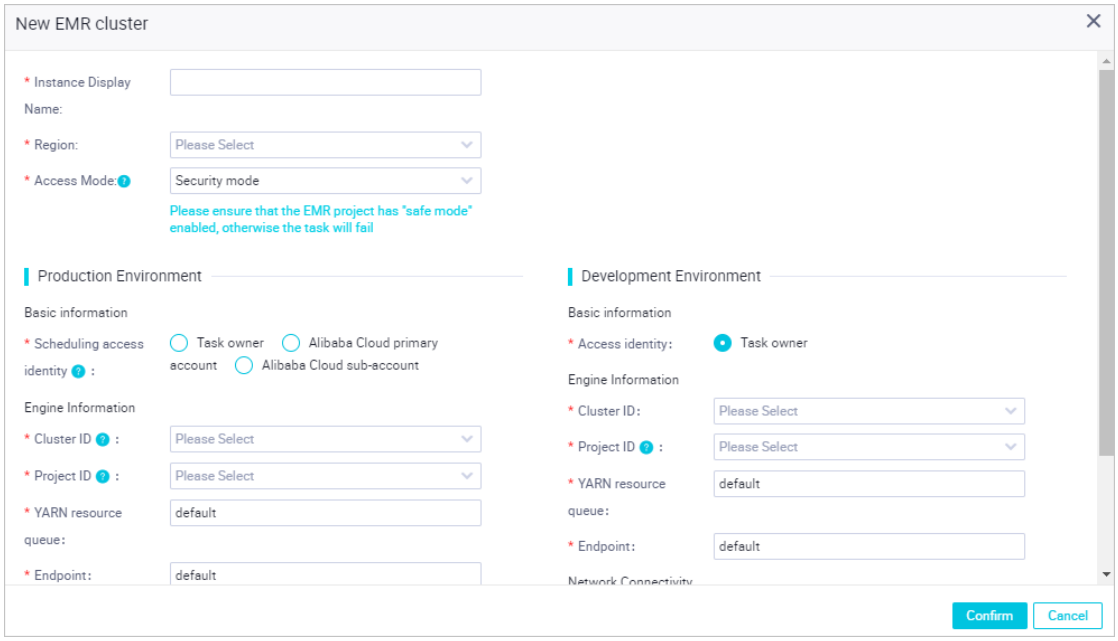
ii. In the upper-right corner of the page, click the icon to go to the Workspace Management page.

iii. In the **Compute Engine Information** section, click the **E-MapReduce** tab.

iv. On the **E-MapReduce** tab, click **Add Instance**.

v. In the **New EMR cluster** dialog box, set the parameters.

Parameters in the New EMR cluster dialog box vary based on the mode in which your DataWorks workspace runs. The following table describes the parameters for a DataWorks workspace in standard mode. You must set the parameters for both the production environment and the development environment.



Parameter	Description
Instance Display Name	The display name of the EMR compute engine instance.
Region	The region of the current workspace.
Access Mode	The access mode of the EMR cluster. Select **Security mode** from the drop-down list and click **Confirm** in the **Please note** message. **Note** You cannot use multiple modes to associate an EMR cluster with a DataWorks workspace at the same time. Proceed with caution when you change the access mode of the EMR cluster because a mode change leads to permission changes.

Parameter	Description
Scheduling access identity	The identity that is used to commit the code of an EMR node to the EMR cluster. The code is committed when the node is committed and deployed to the DataWorks scheduling system in the production environment. The Hadoop user that corresponds to this identity runs the code. Valid values: **Task owner**, **Alibaba Cloud primary account**, and **Alibaba Cloud sub-account**. ■ **Task owner**: commits and runs the code of an EMR node as the node owner. If you select this value, the data permissions of Hadoop users are isolated. **Task owner** can be an Apsara Stack tenant account or a RAM user. ■ **Alibaba Cloud primary account**: commits the code of an EMR node to the EMR cluster by using an Apsara Stack tenant account. ■ **Alibaba Cloud sub-account**: commits the code of an EMR node to the EMR cluster as a RAM user. Note ■ This parameter is available only for the production environment. ■ The credentials of Apsara Stack tenant accounts are in the LDAP directory of the EMR cluster by default. The AliyunEMRDevelopAccess and AliyunEMRFullAccess policies are also attached to Apsara Stack tenant accounts by default. ■ To run EMR nodes as a RAM user, you must add the credential of the RAM user to the LDAP directory of the EMR cluster. For more information, see the *Add the credentials of specific RAM users to the LDAP directory of the EMR cluster* step. In addition, you must attach the AliyunEMRDevelopAccess or AliyunEMRFullAccess policy to the RAM user.

Parameter	Description
Access identity	The identity that is used to commit the code of an EMR node in the development environment to the EMR cluster. Default value: **Task owner**. The Hadoop user that corresponds to the user who runs the node runs the code. **Note** ■ This parameter is available only for the development environment of a workspace in standard mode. ■ Make sure that the credential of the user who runs the node is added to the LDAP directory of the EMR cluster. In addition, make sure that the AliyunEMRDevelopAccess or AliyunEMRFulAccess policy is attached to the user and relevant data permissions are granted to the user. This way, the user can run EMR nodes in DataStudio. **Task owner** can be an Apsara Stack tenant account or a RAM user. ■ The credentials of Apsara Stack tenant accounts are in the LDAP directory of the EMR cluster by default. The AliyunEMRDevelopAccess and AliyunEMRFullAccess policies are also attached to Apsara Stack tenant accounts by default. ■ To run EMR nodes as a RAM user, you must add the credential of the RAM user to the LDAP directory of the EMR cluster. For more information, see the *Add the credentials of specific RAM users to the LDAP directory of the EMR cluster* step. In addition, you must attach the AliyunEMRDevelopAccess or AliyunEMRFullAccess policy to the RAM user.
Cluster ID	The ID of the EMR cluster that you want to associate with the workspace. Select an ID from the drop-down list. The EMR cluster with the selected ID is used as the runtime environment of EMR nodes.
Project ID	The ID of the EMR project that you want to associate with the workspace. Select the ID of an EMR project in Security mode from the drop-down list. **Note** The IDs of the EMR projects that are not in **Security mode** are not displayed and cannot be selected.
YARN resource queue	The name of the resource queue in the EMR cluster. Unless otherwise specified, set this parameter to *default*.

Parameter	Description
Endpoint	The endpoint of the EMR cluster. Unless otherwise specified, set this parameter to *default*.

vi. Click **Confirm**.

4.5.3.2. Create an EMR MR node

You can create an EMR MR node to compute a large-scale dataset by using multiple Map tasks in a parallel manner.

Prerequisites

The EMR folder is available on the DataStudio page only after you bind an E-MapReduce compute engine to the current workspace on the **Project Management** page.

Procedure

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over **Create** and choose **EMR > EMR MR**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **EMR**, and then choose **Create > EMR MR**.
3. In the **Create Node** dialog box, specify **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.
5. On the node configuration tab, select an E-MapReduce compute engine from the **Engine Instance EMR** drop-down list and edit the code of the node.
6. On the configuration tab of the batch synchronization node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, configure parameters in the Schedule section. For more information, see Basic properties.
7. Commit the node.

   i. Click in the toolbar.

   ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.

   iii. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Publish nodes.
8. Test the node. For more information, see Manage auto triggered nodes.

4.5.3.3. Create an EMR Spark SQL node

You can create an EMR Spark SQL node to use the distributed SQL query engine to process structured data, improving the task execution efficiency.

Prerequisites

The EMR folder is available on the DataStudio page only after you bind an E-MapReduce compute engine to the current workspace on the **Project Management** page.

Procedure

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over Create and choose **EMR > EMR Spark SQL**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **EMR**, and then choose **Create > EMR Spark SQL**.
3. In the **Create Node** dialog box, specify **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.
5. On the node configuration tab, select an E-MapReduce compute engine from the **Engine Instance EMR** drop-down list and edit the code of the node.
6. On the configuration tab of the batch synchronization node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, configure parameters in the Schedule section. For more information, see Basic properties.
7. Commit the node.
   i. Click in the toolbar.
   ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.
   iii. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Publish nodes.
8. Test the node. For more information, see Manage auto triggered nodes.

4.5.3.4. Create an EMR Spark node

You can create an EMR Spark node to perform complex memory analysis and build large and low-latency data analysis applications.

Prerequisites

The EMR folder is available on the DataStudio page only after you bind an E-MapReduce compute engine to the current workspace on the **Project Management** page.

Procedure

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over Create and choose **EMR > EMR Spark**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **EMR**, and then choose **Create > EMR Spark**.

3. In the **Create Node** dialog box, specify **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.
5. On the node configuration tab, select an E-MapReduce compute engine from the **Engine Instance EMR** drop-down list and edit the code of the node.
6. On the configuration tab of the batch synchronization node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, configure parameters in the Schedule section. For more information, see Basic properties.
7. Commit the node.
   i. Click in the toolbar.
   ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.
   iii. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Publish nodes.
8. Test the node. For more information, see Manage auto triggered nodes.

4.5.3.5. Create an EMR Hive node

This topic describes how to create an EMR Hive node. This type of node allows you to use SQL-like statements to read data from, write data to, and manage data warehouses with a large amount of data stored in a distributed storage system. By using this type of node, you can efficiently analyze a large amount of log data.

Prerequisites

The EMR folder is available on the DataStudio page only after you bind an E-MapReduce compute engine to the current workspace on the **Project Management** page.

Procedure

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over +Create and choose **EMR > EMR Hive**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **EMR**, and then choose **Create > EMR Hive**.
3. In the **Create Node** dialog box, specify **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.
5. On the node configuration tab, select an E-MapReduce compute engine from the **Engine Instance EMR** drop-down list and edit the code of the node.
6. On the configuration tab of the batch synchronization node, click the **Properties** tab in the right-

side navigation pane. On the Properties tab, configure parameters in the Schedule section. For more information, see Basic properties.

7. Commit the node.
   i. Click in the toolbar.
   ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.
   iii. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Publish nodes.
8. Test the node. For more information, see Manage auto triggered nodes.

4.5.3.6. Create and use an EMR Shell node

You can create an EMR Shell node and run the node by using the code editor.

Prerequisites

- An EMR cluster is created. An inbound rule that contains the following content is added to the security group to which the EMR cluster belongs:
  - Action: Allow
  - Protocol type: Custom TCP
  - Port range: 8898/8898
  - Authorization object: 100.104.0.0/16
- An EMR compute engine instance is associated with the current workspace. The EMR folder is displayed only after you associate an EMR compute engine instance with the workspace on the Workspace Management page. For more information, see Associate an EMR cluster with a workspace.
- If you integrate Hive with Ranger in EMR, you must modify whitelist configurations and restart Hive before you develop EMR nodes in DataWorks. Otherwise, the error message **Cannot modify spark.yarn.queue at runtime** or **Cannot modify SKYNET_BIZDATE at runtime** is returned when you run EMR nodes.
  i. You can modify the whitelist configurations by using custom parameters in EMR. You can append key-value pairs to the value of a custom parameter. In this example, the custom parameter for Hive components is used. The following code provides an example:

  ```
  hive.security.authorization.sqlstd.confwhitelist.append=tez.*|spark.*|mapred.*|mapred
  uce.*|ALISA.*|SKYNET.*
  ```

  > **Note** In the code, `ALISA.*` and `SKYNET.*` are configurations in DataWorks.

  ii. After the whitelist configurations are modified, restart the Hive service to make the configurations take effect.

Create an EMR Shell node and use the node to develop data

1. Log on to the DataWorks console.
2. Create an **EMR Shell** node.

i. On the DataStudio page, move the pointer over the **+Create** icon and choose **EMR > EMR Shell**.

You can also find the workflow in which you want to create the EMR Shell node, right-click the workflow name, and then choose **Create > EMR > EMR Shell**.

ii. In the **Create Node** dialog box, set the **Node Name**, **Node Type**, and **Location** parameters.

> **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

iii. Click **Commit**. Then, the configuration tab of the **EMR Shell** node appears.

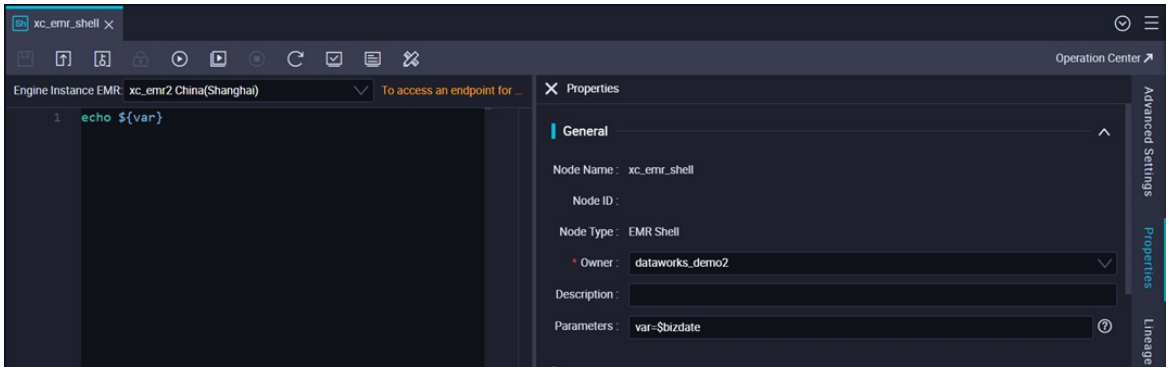
3. Use the **EMR Shell** node to develop data.

The following code provides an example:

```
DD=`date`;
echo "hello world, $DD"
## Scheduling parameters are supported.
echo ${var};
```

For more information about the scheduling parameters, see Scheduling parameters.

If you want to change the values that are assigned to the parameters in the code, click the Run with Parameters icon in the top toolbar.



4. Click the **Advanced Settings** tab in the right-side navigation pane. In the Advanced Settings panel, change the values of the parameters.
   - "USE_GATEWAY":true: If you set this parameter to true, the EMR Shell node is automatically committed to the master node of an EMR gateway cluster.
   - "SPARK_CONF": "--conf spark.driver.memory=2g --conf xxx=xxx": the parameters that are required to run Spark jobs. You can configure multiple parameters in the --conf xxx=xxx format.
   - "queue": the scheduling queue to which jobs are committed. Default value: default.
   - "vcores": the number of CPU cores. Default value:1.
   - "memory": the memory that is allocated to the launcher. Unit: MB. Default value: 2048.
   - "priority": the priority. Default value: 1.
   - "FLOW_SKIP_SQL_ANALYZE": specifies how SQL statements are executed. A value of false indicates that only one SQL statement is executed at a time. A value of true indicates that multiple SQL statements are executed at a time.
5. Configure scheduling properties for the EMR Shell node.

If you want the system to periodically run the EMR Shell node, you can click the **Properties** tab in the right-side navigation pane to configure scheduling properties for the node based on your business requirements.

6. Commit and deploy the EMR Shell node.
   i. Click the icon in the top toolbar to save the node.
   ii. Click the icon in the top toolbar to commit the node.
   iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.
   iv. Click **OK**.
7. View the EMR Shell node.
   i. On the configuration tab of the EMR Shell node, click **Operation Center** in the upper-right corner to go to Operation Center.
   ii. View the scheduled EMR Shell node. For more information, see Manage auto triggered nodes.

4.5.3.7. Create and use an EMR Spark Shell node

You can create an EMR Spark Shell node and run the node by using the code editor.

Prerequisites

- An EMR cluster is created. An inbound rule that contains the following content is added to the security group to which the EMR cluster belongs:
  - Action: Allow
  - Protocol type: Custom TCP
  - Port range: 8898/8898
  - Authorization object: 100.104.0.0/16
- An EMR compute engine instance is associated with the current workspace. The EMR folder is displayed only after you associate an EMR compute engine instance with the workspace on the Workspace Management page. For more information, see Associate an EMR cluster with a workspace.
- If you integrate Hive with Ranger in EMR, you must modify whitelist configurations and restart Hive before you develop EMR nodes in DataWorks. Otherwise, the error message **Cannot modify spark.yarn.queue at runtime** or **Cannot modify SKYNET_BIZDATE at runtime** is returned if you run EMR nodes.
  i. You can modify the whitelist configurations by using custom parameters in EMR. You can append key-value pairs to the value of a custom parameter. In this example, the custom parameter for Hive components is used. The following code provides an example:

  ```
  hive.security.authorization.sqlstd.confwhitelist.append=tez.*|spark.*|mapred.*|mapred
  uce.*|ALISA.*|SKYNET.*
  ```

  > **Note** In the code, `ALISA.*` and `SKYNET.*` are configurations in DataWorks.

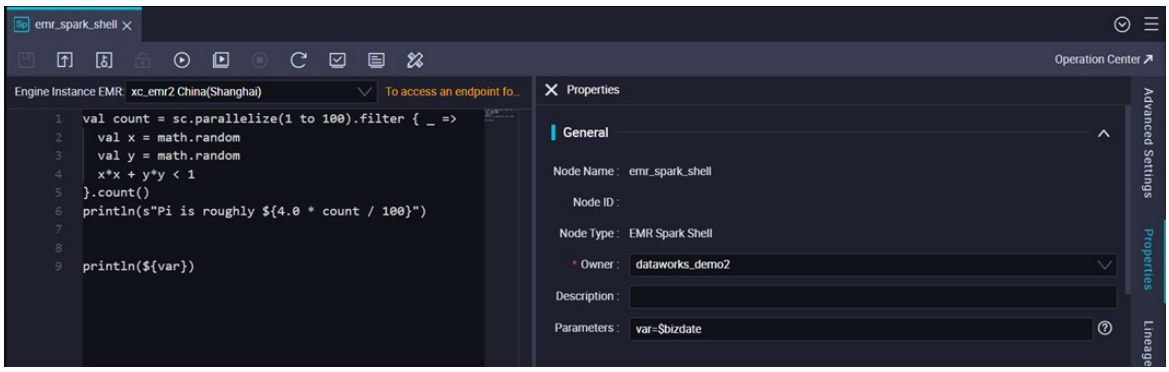
  ii. After the whitelist configurations are modified, restart the Hive service to make the configurations take effect.

Create an EMR Spark Shell node and use the node to develop data

1. Log on to the DataWorks console.
2. Create an **EMR Spark Shell** node.
   i. On the DataStudio page, move the pointer over the **Create** icon and choose **EMR > EMR Spark Shell**.
   You can also find the workflow in which you want to create the EMR Spark Shell node, right-click the workflow name, and then choose **Create > EMR > EMR Spark Shell**.
   ii. In the **Create Node** dialog box, set the **Node Name**, **Node Type**, and **Location** parameters.
   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
   iii. Click **Commit**. Then, the configuration tab of the **EMR Spark Shell** node appears.
3. Use the **EMR Spark Shell** node to develop data.
   The following code provides an example:

```
val count = sc.parallelize(1 to 100).filter { _ =>
  val x = math.random
  val y = math.random
  x*x + y*y < 1
}.count()
println(s"Pi is roughly ${4.0 * count / 100}")
println(${var})
```

   Scheduling parameters are supported. For more information, see Scheduling parameters
   If you want to change the values that are assigned to the parameters in the code, click the Run with Parameters icon in the top toolbar.



4. Click the **Advanced Settings** tab in the right-side navigation pane. In the Advanced Settings panel, change the values of the parameters.
   - "USE_GATEWAY":true: If you set this parameter to true, the EMR Spark Shell node is automatically committed to the master node of an EMR gateway cluster.
   - "SPARK_CONF": "--conf spark.driver.memory=2g --conf xxx=xxx": the parameters that are required to run Spark jobs. You can configure multiple parameters in the --conf xxx=xxx format.
   - "queue": the scheduling queue to which jobs are committed. Default value: default.
   - "vcores": the number of CPU cores. Default value:1.
   - "memory": the memory that is allocated to the launcher. Unit: MB. Default value: 2048.

- "priority": the priority. Default value: 1.
- "FLOW_SKIP_SQL_ANALYZE": specifies how SQL statements are executed. A value of false indicates that only one SQL statement is executed at a time. A value of true indicates that multiple SQL statements are executed at a time.

5. Configure scheduling properties for the EMR Spark Shell node.

   If you want the system to periodically run the EMR Spark Shell node, you can click the **Properties** tab in the right-side navigation pane to configure scheduling properties for the node based on your business requirements.

6. Commit and deploy the EMR Spark Shell node.

   i. Click the icon in the top toolbar to save the node.

   ii. Click the icon in the top toolbar to commit the node.

   iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

   iv. Click **OK**.

7. View the EMR Spark Shell node.

   i. On the configuration tab of the EMR Spark Shell node, click **Operation Center** in the upper-right corner to go to Operation Center.

   ii. View the scheduled EMR Spark Shell node. For more information, see Manage auto triggered nodes.

4.5.3.8. Create an EMR Impala node

This topic describes how to create an EMR Impala node. EMR Impala nodes allow you to perform interactive analysis and queries by executing SQL statements on petabytes of data.

Prerequisites

- An EMR cluster is created. An inbound rule that contains the following content is added to the security group to which the EMR cluster belongs:
  - Action: Allow
  - Protocol type: Custom TCP
  - Port range: 8898/8898
  - Authorization object: 100.104.0.0/16
- An EMR compute engine instance is associated with the current workspace. The EMR folder is displayed only after you associate an EMR compute engine instance with the workspace on the Workspace Management page. For more information, see Associate an EMR cluster with a workspace.
- If you integrate Hive with Ranger in EMR, you must modify whitelist configurations and restart Hive before you develop EMR nodes in DataWorks. Otherwise, the error message **Cannot modify spark.yarn.queue at runtime** or **Cannot modify SKYNET_BIZDATE at runtime** is returned if you run EMR nodes.

  i. You can modify the whitelist configurations by using custom parameters in EMR. You can append key-value pairs to the value of a custom parameter. In this example, the custom parameter for Hive components is used. The following code provides an example:

```
hive.security.authorization.sqlstd.confwhitelist.append=tez.*|spark.*|mapred.*|mapred
uce.*|ALISA.*|SKYNET.*
```

> **Note** In the code, `ALISA.*` and `SKYNET.*` are configurations in DataWorks.

ii. After the whitelist configurations are modified, restart the Hive service to make the configurations take effect.

Procedure

1. Log on to the DataWorks console.
2. On the DataStudio page, move the pointer over the +Create icon and choose **EMR > EMR Impala**.

   You can also find the required workflow, right-click the workflow name, and then choose **Create > EMR > EMR Impala**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**. Then, the configuration tab of the **EMR Impala** node appears.
5. On the node configuration tab, enter the code for the node.

   Sample code:

   ```
   -- SQL statement example
   -- The size of SQL statements cannot exceed 130 KB.
   show tables;
   -- Scheduling parameters are supported.
   CREATE TABLE IF NOT EXISTS userinfo (
   ip STRING COMMENT'IP address',
   uid STRING COMMENT'User ID'
   )PARTITIONED BY(
   dt STRING
   );
   ALTER TABLE userinfo ADD IF NOT EXISTS PARTITION(dt=${bizdate});
   -- The system automatically adds limit 10000 to the SELECT statement.
   select * from userinfo ;
   ```

   For more information about the scheduling parameters, see Scheduling parameters.

   If you want to change the values that are assigned to the parameters in the code, click the Run with Parameters icon in the top toolbar.

   > **Note** If multiple EMR compute engine instances are associated with the current workspace, you must select one EMR compute engine instance. If only one EMR compute engine instance is associated with the current workspace, you do not need to make a choice.
6. Click the **Advanced Settings** tab in the right-side navigation pane. In the Advanced Settings panel, change the values of the parameters.
   - "USE_GATEWAY":true: If you set this parameter to true, the EMR Impala node is automatically

committed to the master node of an EMR gateway cluster.

- "SPARK_CONF": "--conf spark.driver.memory=2g --conf xxx=xxx": the parameters that are required to run Spark jobs. You can configure multiple parameters in the --conf xxx=xxx format.
- "queue": the scheduling queue to which jobs are committed. Default value: default.
- "vcores": the number of CPU cores. Default value:1.
- "memory": the memory that is allocated to the launcher. Unit: MB. Default value: 2048.
- "priority": the priority. Default value: 1.
- "FLOW_SKIP_SQL_ANALYZE": specifies how SQL statements are executed. A value of false indicates that only one SQL statement is executed at a time. A value of true indicates that multiple SQL statements are executed at a time.

7. Configure scheduling properties for the EMR Impala node.

   If you want the system to periodically run the EMR Impala node, you can click the **Properties** tab in the right-side navigation pane to configure scheduling properties for the node based on your business requirements.

8. Commit and deploy the EMR Impala node.

   i. Click the icon in the top toolbar to save the node.

   ii. Click the icon in the top toolbar to commit the node.

   iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

   iv. Click **OK**.

9. View the EMR Impala node.

   i. On the configuration tab of the EMR Impala node, click **Operation Center** in the upper-right corner to go to Operation Center.

   ii. View the scheduled EMR Impala node. For more information, see Manage auto triggered nodes.

4.5.3.9. Create and use an EMR Presto node

This topic describes how to create an EMR Presto node in DataWorks. EMR Presto nodes allow you to perform interactive analysis and queries on large amounts of structured and unstructured data.

Prerequisites

- An EMR cluster is created. An inbound rule that contains the following content is added to the security group to which the EMR cluster belongs:
  - Action: Allow
  - Protocol type: Custom TCP
  - Port range: 8898/8898
  - Authorization object: 100.104.0.0/16
- An EMR compute engine instance is associated with the current workspace. The EMR folder is displayed only after you associate an EMR compute engine instance with the workspace on the Workspace Management page. For more information, see Associate an EMR cluster with a workspace.
- If you integrate Hive with Ranger in EMR, you must modify whitelist configurations and restart Hive before you develop EMR nodes in DataWorks. Otherwise, the error message **Cannot modify**

spark.yarn.queue at runtime or **Cannot modify SKYNET_BIZDATE at runtime** is returned if you run EMR nodes.

i. You can modify the whitelist configurations by using custom parameters in EMR. You can append key-value pairs to the value of a custom parameter. In this example, the custom parameter for Hive components is used. The following code provides an example:

```
hive.security.authorization.sqlstd.confwhitelist.append=tez.*|spark.*|mapred.*|mapred
uce.*|ALISA.*|SKYNET.*
```

> **Note** In the code, `ALISA.*` and `SKYNET.*` are configurations in DataWorks.

ii. After the whitelist configurations are modified, restart the Hive service to make the configurations take effect.

Procedure

1. Log on to the DataWorks console.
2. On the DataStudio page, move the pointer over the **Create** icon and choose **EMR > EMR Presto**.

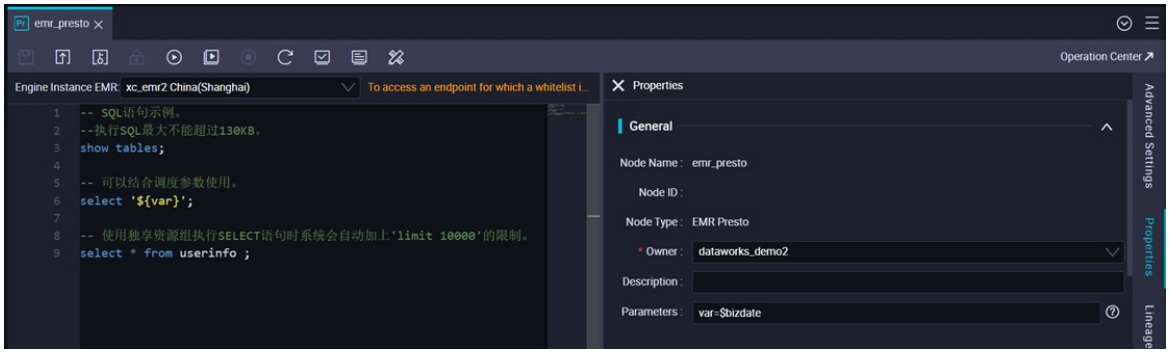
   You can also find the workflow in which you want to create the EMR Presto node, right-click the workflow name, and then choose **Create > EMR > EMR Presto**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**. Then, the configuration tab of the **EMR Presto** node appears.
5. On the node configuration tab, write code for the node.

```
-- SQL statement example
-- The size of SQL statements cannot exceed 130 KB.
show tables;
-- Scheduling parameters are supported.
select '${var}';
-- The system automatically adds limit 10000 to the SELECT statement.
select * from userinfo ;
```

   For more information about the scheduling parameters, see Scheduling parameters.

   If you want to change the values that are assigned to the parameters in the code, click the Run with Parameters icon in the top toolbar.



> **Note** If multiple EMR compute engine instances are associated with the current workspace, you must select one EMR compute engine instance. If only one EMR compute engine instance is associated with the current workspace, you do not need to make a choice.

6. Click the **Advanced Settings** tab in the right-side navigation pane. In the Advanced Settings panel, change the values of the parameters.
   - "USE_GATEWAY":true: If you set this parameter to true, the EMR Presto node is automatically committed to the master node of an EMR gateway cluster.
   - "SPARK_CONF": "--conf spark.driver.memory=2g --conf xxx=xxx": the parameters that are required to run Spark jobs. You can configure multiple parameters in the --conf xxx=xxx format.
   - "queue": the scheduling queue to which jobs are committed. Default value: default.
   - "vcores": the number of CPU cores. Default value:1.
   - "memory": the memory that is allocated to the launcher. Unit: MB. Default value: 2048.
   - "priority": the priority. Default value: 1.
   - "FLOW_SKIP_SQL_ANALYZE": specifies how SQL statements are executed. A value of false indicates that only one SQL statement is executed at a time. A value of true indicates that multiple SQL statements are executed at a time.
7. Configure scheduling properties for the EMR Presto node.

   If you want the system to periodically run the EMR Presto node, you can click the **Properties** tab in the right-side navigation pane to configure scheduling properties for the node based on your business requirements.
8. Commit and deploy the EMR Presto node.
   i. Click the icon in the top toolbar to save the node.
   ii. Click the icon in the top toolbar to commit the node.
   iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.
   iv. Click **OK**.
9. View the EMR Presto node.
   i. On the configuration tab of the EMR Spark Shell node, click **Operation Center** in the upper-right corner to go to Operation Center.
   ii. View the scheduled EMR Presto node. For more information, see Manage auto triggered nodes.

4.5.3.10. Create and use an EMR JAR resource

DataWorks allows you to create EMR JAR resources in the DataWorks console. You can upload a Java Archive (JAR) file that contains user-defined functions (UDFs) or open source MapReduce code as an EMR JAR resource. Then, you can reference the resource in compute nodes such as an EMR MR node. This topic describes how to create an EMR JAR resource by uploading a file, commit the resource, and reference the resource in compute nodes such as an EMR MR node.

Prerequisites

- An EMR cluster is created. An inbound rule that contains the following content is added to the security group to which the EMR cluster belongs:
  - Action: Allow

- Action: Allow
- Protocol type: Custom TCP
- Port range: 8898/8898
- Authorization object: 100.104.0.0/16

- An EMR compute engine instance is associated with the current workspace. The EMR folder is displayed only after you associate an EMR compute engine instance with the workspace on the Workspace Management page.
- If you integrate Hive with Ranger in EMR, you must modify whitelist configurations and restart Hive before you develop EMR nodes in DataWorks. Otherwise, the error message **Cannot modify spark.yarn.queue at runtime** or **Cannot modify SKYNET_BIZDATE at runtime** is returned if you run EMR nodes.
  i. You can modify the whitelist configurations by using custom parameters in EMR. You can append key-value pairs to the value of a custom parameter. In this example, the custom parameter for Hive components is used. The following code provides an example:

  ```
  hive.security.authorization.sqlstd.confwhitelist.append=tez.*|spark.*|mapred.*|mapred
  uce.*|ALISA.*|SKYNET.*
  ```

  > **Note** In the code, `ALISA.*` and `SKYNET.*` are configurations in DataWorks.

  ii. After the whitelist configurations are modified, restart the Hive service to make the configurations take effect.

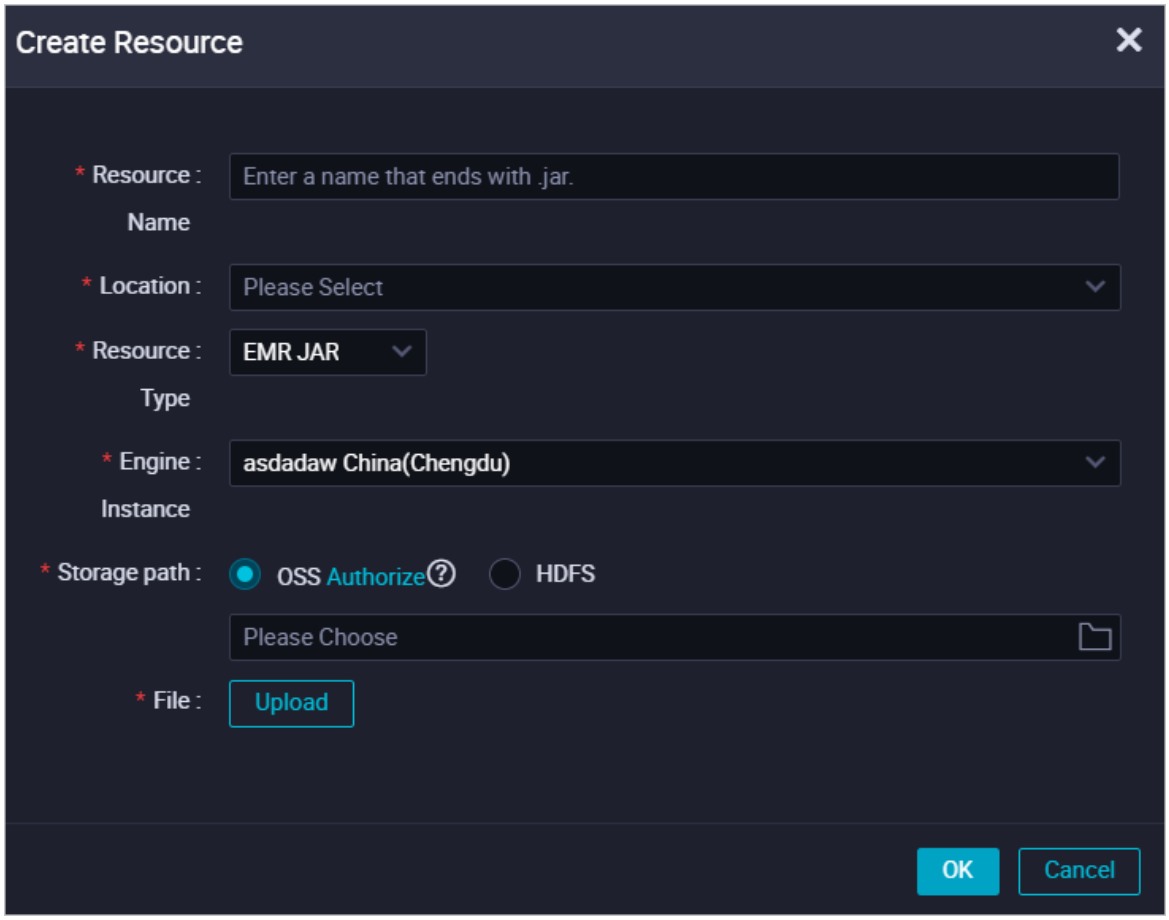
Limits

If Kerberos authentication is enabled for an EMR cluster., you cannot create tables, resources, and functions in a visualized manner for this cluster.

Procedure

1. Log on to the DataWorks console.
2. On the DataStudio page, move the pointer over the **Create** icon and choose **EMR > Resource > EMR JAR**.

   You can also find the required workflow, right-click the workflow name, and then choose **Create > EMR > Resource > EMR JAR**.
3. In the **Create Resource** dialog box, set the following parameters.



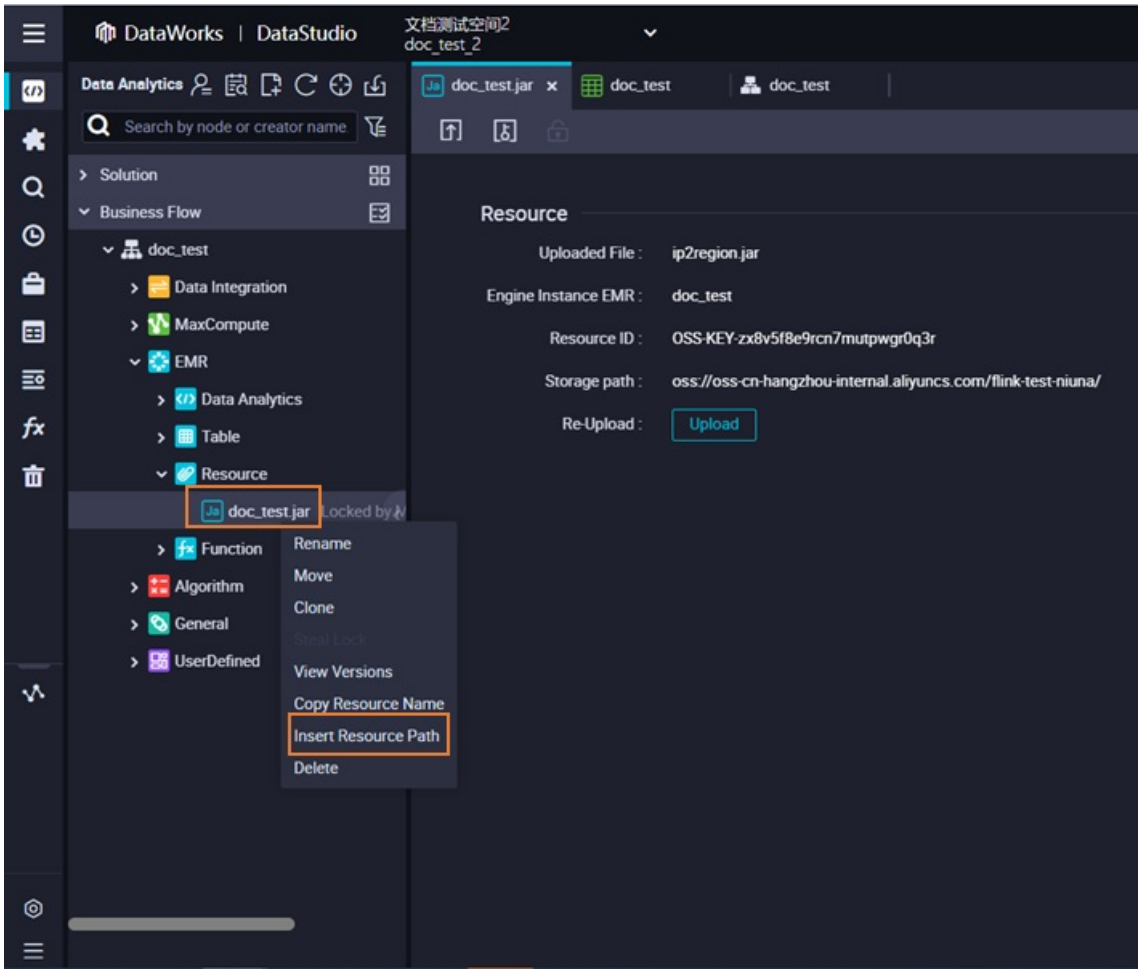
Parameter	Description
Resource Name	The name of the resource that you want to create. The resource name must have the suffix .jar.
Location	The folder for storing the resource. The default value is the path of the current folder. You can modify the path based on your business requirements.
File Type	The type of the resource. Set the parameter to EMR JAR.
Engine Instance	The EMR compute engine instance to which the resource belongs. Select an instance from the drop-down list.
Storage path	The storage path of the resource. Valid values: **OSS** and **HDFS**. ◦ If you select **OSS**, you must click Authorize next to OSS to authorize DataWorks and EMR to access Object Storage Service (OSS). Then, select a folder. ◦ If you select **HDFS**, enter a storage path.
File	The file that you want to upload. You can click **Upload**, select a file from your on-premises machine, and then click **Open**.

4. Click **Create**.

5. Click the and icons in the top toolbar to save and commit the resource to the development environment.

What's next

After you create an EMR JAR resource, you can reference the resource in the code of compute nodes such as an EMR MR node. The following figure shows how to reference the resource. For more information, see Create an EMR MR node.



4.5.3.11. Create an EMR table

This topic describes how to create an EMR table.

Prerequisites

- An EMR compute engine instance is associated with the current workspace. The EMR folder is displayed only after you associate an EMR compute engine instance with the workspace on the Workspace Management page.
- The metadata of an EMR data source is collected in Data Map so that you can select an EMR database when you create a table.

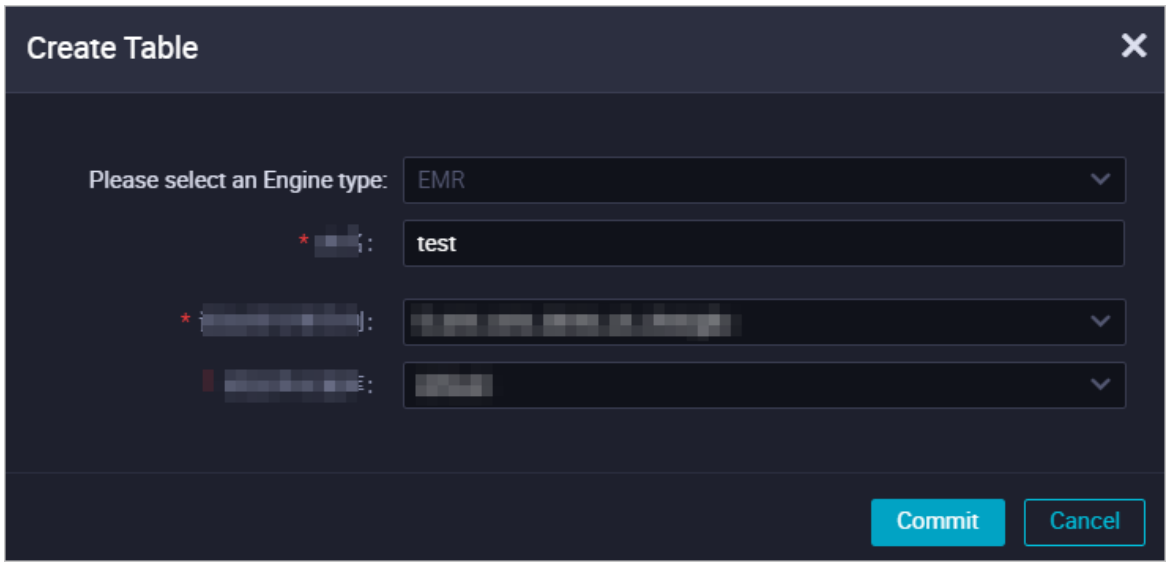
Limits

If Kerberos authentication is enabled for an EMR cluster., you cannot create tables, resources, and functions in a visualized manner for this cluster.

Procedure

1. Log on to the DataWorks console.
2. Move the pointer over the +Create icon and choose **EMR > table**.

   You can also find the workflow in which you want to create an EMR table, right-click **EMR**, and then choose **Create > Table**.
3. In the **Create Table** dialog box, set the parameters as required.



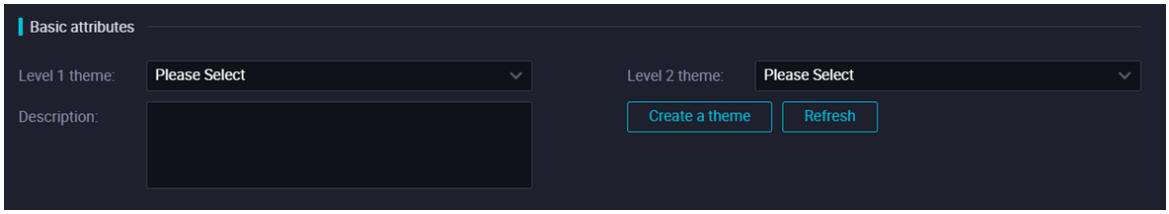
Parameter	Description
Engine type	The default value is EMR, which cannot be changed.
Table Name	The name of the EMR table.
Engine Instance	Select a required compute engine instance from the drop-down list.
Database	Select the database in which the compute engine instance resides from the drop-down list. **Note** You must collect metadata before you can select a database.

4. Click **Create**. The table configuration tab appears.

   The upper part of the tab shows the configurations that you specified in the **Create Table** dialog box. You can change the database where the EMR compute engine instance resides. To create a database, click **Create a database**. In the **Create a database** dialog box, set the parameters as

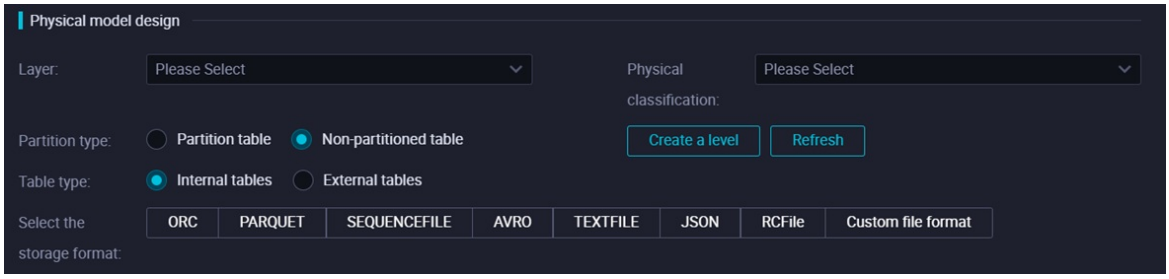
required and click **OK**.

5. In the **Basic attributes** section, set the parameters as required.



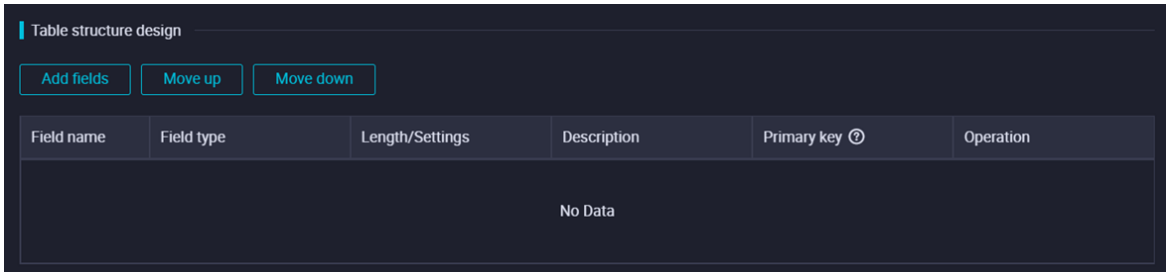
Parameter	Description
Level 1 theme	The name of the level-1 folder where the table resides. **Note** The level-1 and level-2 folders show the table locations in DataWorks for you to manage tables with ease.
Level 2 theme	The name of the level-2 folder where the table resides.
Create a theme	Click **Create a theme** to go to the **Folder Management** tab. On this tab, you can create level-1 and level-2 folders.
Refresh	After you create a folder, click **Refresh**.
Description	The description of the table.

6. In the **Physical model design** section, set the parameters as required.



Parameter	Description
Layer	The levels and categories of the table. Select the appropriate level and category from the drop-down lists. To add levels and categories, click **Create a level** to go to the **Level Management** tab. After you create levels and categories, click **Refresh**.
Physical classification	
Partition type	Valid values: **Partition table** and **Non-partitioned table**.
Table type	Valid values: **Internal tables** and **External tables**.

7. In the **Table structure design** section, set the parameters as required.



Parameter	Description
Add fields	To add a field, click **Add fields**, configure the field information, and then click **Save** in the Operation column.
Move up	Adjusts the field sequence of a table that has not been created. If you want to adjust the sequence of fields in an existing table, you must delete the table and create another table with the same name. These operations are forbidden in the production environment.
Move down	
Field name	The name of the field, which can contain letters, digits, and underscores (_).
Data type	The EMR table supports the following data types: TINYINT, SMALLINT, INT, BIGINT, FLOAT, DOUBLE, DECIMAL, VARCHAR, CHAR, STRING, BINARY, DATETIME, DATE, TIMESTAMP, BOOLEAN, ARRAY, MAP, and STRUCT.
Length/Settings	You must set this parameter if the data type that you specify for the field has a length limit.
Description	The description of the field.
Primary key	Specifies whether the field serves as the primary key. The primary key ensures that each record is unique for your business. DataWorks does not impose a limit on the field that can be specified as the primary key.
Edit	After you save the field, you can click **Edit** to edit the field and then click **Save**.
Delete	Deletes a created field. **Note** If you want to delete a field from an existing table and then commit the table, you must delete the table and create another table with the same name. These operations are forbidden in the production environment.
Add partitions	If you set the **Partition type** parameter to **Partition table** in the **Physical model design** section, you must configure a partition for the table. You can add a partition to the current table. If you want to add a partition to an existing table and then commit the table, you must delete the table and create another table with the same name. These operations are forbidden in the production environment.

8. Click the icon in the top toolbar to commit the EMR table to the production environment.

   If you are using a workspace in standard mode, commit the table to the development environment and the production environment in sequence.

   > **Notice** You cannot create an EMR table in DDL mode.

4.5.3.12. Create an EMR function

This topic describes how to create an EMR function.

Prerequisites

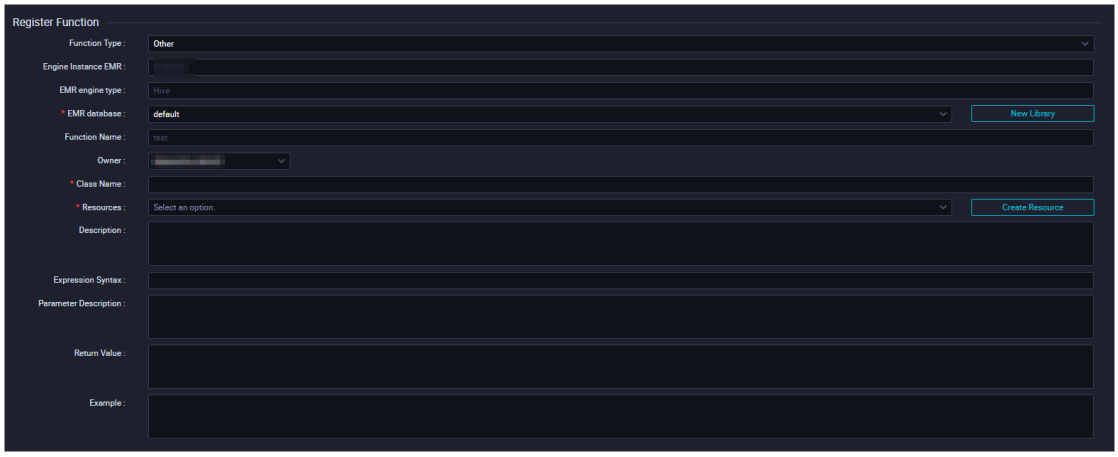
- An EMR cluster is created. An inbound rule that contains the following content is added to the security group to which the EMR cluster belongs:
  - Action: Allow
  - Protocol type: Custom TCP
  - Port range: 8898/8898
  - Authorization object: 100.104.0.0/16
- An EMR compute engine instance is associated with the current workspace. The EMR folder is displayed only after you associate an EMR compute engine instance with the workspace on the Workspace Management page.
- The required resources are uploaded.

Limits

If Kerberos authentication is enabled for an EMR cluster., you cannot create tables, resources, and functions in a visualized manner for this cluster.

Procedure

1. Log on to the DataWorks console.
2. Create a workflow. For more information, see Overview.
3. Write code in a local Java environment and compress the code to a JAR package. Then, create a JAR resource and commit the resource. For more information, see Create and use an EMR JAR resource.
4. Create a function.
   i. Click the workflow in the Scheduled Workflow pane, right-click **EMR**, and then choose **Create > Function**.
   ii. In the **Create Function** dialog box, set the **Function Name**, **Engine Instance**, and **Location** parameters.
   iii. Click **Create**.
   iv. In the **Function information** section of the configuration tab that appears, set the parameters.



Parameter	Description
Function Type	The type of the function. Valid values: **Mathematical Operation Functions**, **Aggregate Functions**, **String Processing Functions**, **Date Functions**, **Window Functions**, and **Other Functions**.
Engine Instance	The EMR compute engine instance. By default, the system automatically selects the EMR compute engine instance. You cannot change the value.
Engine Type	The type of the compute engine instance. By default, the system automatically selects EMR. You cannot change the value.
EMR database	The database where the EMR cluster resides. Select a database from the drop-down list. To create a database, click **New Library**. In the **New Library** dialog box, set the parameters and click **OK**.
Function Name	The name of the function. You can use this name to reference the function in SQL statements. The function name must be globally unique and cannot be changed after the function is created.
Owner	The value of this parameter is automatically displayed.
Class Name	Required. The name of the class that implements the function.
Resource	Required. The resource to be used in the function. Select a resource from the ones that are created in the current workspace from the drop-down list. To create a resource, click **Create Resource**. In the **Create Resource** dialog box, set the parameters and click **Create**.
Description	The description of the function.
Expression Syntax	The syntax of the function. Example: `test`.
Parameter Description	The description of the input and output parameters that are supported.
Return Value	Optional. The return value. Example: 1.
Example	Optional. The example of the function.

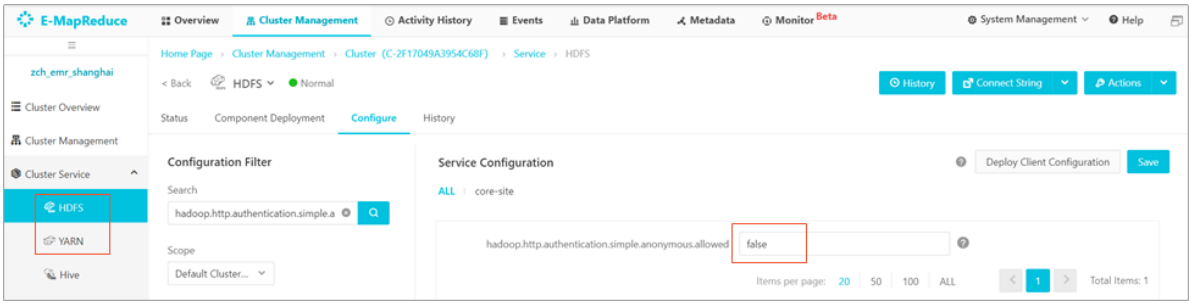
5. Click the icon in the top toolbar.
6. Commit the function.
   i. Click the icon in the top toolbar.
   ii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.
   iii. Click **OK**.

4.5.3.13. Create and use an EMR Spark Streaming node

E-MapReduce (EMR) Spark Streaming nodes can be used to process streaming data with high throughput and support fault tolerance. These nodes help you restore data streams on which errors occur. This topic describes how to create an EMR Spark Streaming node and use the node to develop data.

Prerequisites

- An Alibaba Cloud EMR cluster is created. An inbound rule that contains the following content is added to the security group to which the cluster belongs.
  - Action: Allow
  - Protocol type: Custom TCP
  - Port range: 8898/8898
  - Authorization object: 100.104.0.0/16
- The EMR cluster is associated with your DataWorks workspace as a compute engine instance. The EMR folder is displayed on the DataStudio page only after an EMR cluster is associated with your workspace as a compute engine instance on the Workspace Management page. For more information, see Configure a workspace.
- The hadoop.http.authentication.simple.anonymous.allowed parameter is set to true on the HDFS page in the EMR console. The HDFS and YARN services are restarted.



Create an EMR Spark Streaming node and use the node to develop data

1. Log on to the DataWorks console.
2. Create a workflow.

   If you have a workflow, skip this step.

   i. Move the pointer over the Create icon and select **Workflow**.
   ii. In the **Create Workflow** dialog box, set the **Workflow Name** parameter.

iii. Click **Create**.

3. Create an **EMR Spark Streaming** node.

   i. On the DataStudio page, move the pointer over the **Create** icon and choose **EMR > EMR Spark Streaming**.

   Alternatively, you can find the workflow that you want to manage, right-click the workflow name, and then choose **Create > EMR > EMR Spark Streaming**.

   ii. In the **Create Node** dialog box, set the **Node Name**, **Node Type**, and **Location** parameters.

   > **Note** The name of the node must be 1 to 128 characters in length, and can contain letters, digits, underscores (_), and periods (.).

   iii. Click **Commit**. Then, the configuration tab of the **EMR Spark Streaming** node appears.

4. Use the **EMR Spark Streaming** node to develop data.

   i. Select the EMR compute engine instance.

   On the configuration tab of the **EMR Spark Streaming** node, select the EMR compute engine instance.

   ii. Write code for the EMR Spark Streaming node.

   On the configuration tab of the EMR Spark Streaming node, write code for the node. The following code provides an example:

   ```
   spark-submit --master yarn-cluster --executor-cores 2 --executor-memory 2g --driver
   -memory 1g --num-executors 2 --class com.aliyun.emr.example.spark.streaming.JavaLog
   hubWordCount /tmp/examples-1.2.0-shaded.jar <logService-project> <logService-store>
   <group> <endpoint> <access-key-id> <access-key-secret>
   ```

   - The system automatically generates `spark-submit` after the node is created.
   - `/tmp/examples-1.2.0-shaded.jar` is the name of the JAR package generated by the node code. For more information about the code for the node, see Consume data in real time.

     > **Note** The JAR package can be stored in the master node of the EMR cluster or in Object Storage Service (OSS). We recommend that you store the JAR package in OSS. For more information about how to store the JAR package in OSS, see Operations in the OSS console.

   - You must replace `access-key-id` and `access-key-secret` with the AccessKey ID and AccessKey secret of your Apsara Stack tenant account. To obtain the AccessKey ID and AccessKey secret, you can log on to the DataWorks console, move the pointer over the account name in the upper-right corner, and then select **User Info**.

   For more information about Spark Streaming parameters, see Spark documentation.

   iii. Save and run the EMR Spark Streaming node.

   In the top toolbar, click the icon to save the EMR Spark Streaming node and click the icon to run the EMR Spark Streaming node.

5. Click the **Advanced Settings** tab in the right-side navigation pane. In the Advanced Settings panel, change the values of the parameters.

- "USE_GATEWAY":true: If you set this parameter to true, the EMR Spark Streaming node is automatically committed to the master node of an EMR gateway cluster.
- "SPARK_CONF": "--conf spark.driver.memory=2g --conf xxx=xxx": the parameters for running Spark jobs. You can specify multiple parameters in the format of --conf xxx=xxx.
- "queue": the scheduling queue to which jobs are committed. Default value: default.
- "priority": the priority. Default value: 1.
- "FLOW_SKIP_SQL_ANALYZE": specifies how SQL statements are executed. A value of false indicates that only one SQL statement is executed at a time. A value of true indicates that multiple SQL statements are executed at a time.

6. Configure scheduling properties for the node.

   If you want the system to periodically run the node, you can click the **Properties** tab in the right-side navigation pane to configure scheduling properties for the node based on your business requirements.

   - Configure basic properties for the node. For more information, see Basic properties.
   - You can select a mode to start the node and a mode to rerun the node. For more information, see Scheduling properties.

7. Commit and deploy the node.

   i. Click the icon in the top toolbar to save the node.

   ii. Click the icon in the top toolbar to commit the node.

   iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

   iv. Click **OK**.

   If you use a workspace in standard mode, you must deploy the node to the production environment after you commit the node. Click **Deploy** in the upper-right corner. For more information, see Deploy nodes.

8. View the real-time computing node.

   i. Click **Operation Center** in the top navigation bar of the DataStudio page to go to Operation Center.

   ii. View the real-time computing node that is running. For more information, see Manage real-time computing nodes.

4.5.3.14. Create and use an EMR Streaming SQL node

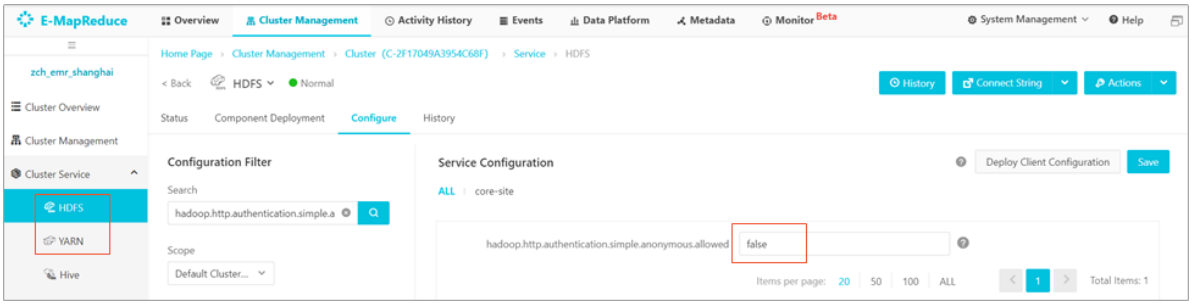
E-MapReduce (EMR) Streaming SQL nodes allow you to use SQL statements to develop streaming analytics jobs. This topic describes how to create an EMR Streaming SQL node and use the node to develop data.

Prerequisites

- An Alibaba Cloud EMR cluster is created. An inbound rule that contains the following content is added to the security group to which the cluster belongs.
  - Action: Allow
  - Protocol type: Custom TCP
  - Port range: 8898/8898

  - Authorization object: 100.104.0.0/16

- The EMR cluster is associated with your DataWorks workspace as a compute engine instance. The EMR folder is displayed on the DataStudio page only after an EMR cluster is associated with your workspace as a compute engine instance on the Workspace Management page. For more information, see Configure a workspace.
- The hadoop.http.authentication.simple.anonymous.allowed parameter is set to true on the HDFS page in the EMR console The HDFS and YARN services are restarted.



Create an EMR Streaming SQL node and use the node to develop data

1. Log on to the DataWorks console.
2. Create a workflow.

   If you have a workflow, skip this step.

   i. Move the pointer over the +Create icon and select **Workflow**.

   ii. In the **Create Workflow** dialog box, set the **Workflow Name** parameter.

   iii. Click **Create**.
3. Create an **EMR Streaming SQL** node.

   i. On the DataStudio page, move the pointer over the +Create icon and choose **EMR > EMR Streaming SQL**.

   Alternatively, you can find the workflow that you want to manage, right-click the workflow name, and then choose **Create > EMR > EMR Streaming SQL**.

   ii. In the **Create Node** dialog box, set the **Node Name**, **Node Type**, and **Location** parameters.

   > **Note** The name of the node must be 1 to 128 characters in length, and can contain letters, digits, underscores (_), and periods (.).

   iii. Click **Commit**. Then, the configuration tab of the **EMR Streaming SQL** node appears.
4. Use the **EMR Streaming SQL** node to develop data.

   i. Select the EMR compute engine instance.

   On the configuration tab of the **EMR Streaming SQL** node, select the EMR compute engine instance.

ii. Write code for the EMR Streaming SQL node.

On the configuration tab of the EMR Streaming SQL node, write code for the node. The following code provides an example:

```
-- dbName: the name of the database.
CREATE DATABASE IF NOT EXISTS ${dbName};
USE ${dbName};
-- Create a Log Service table.
-- slsTableName: the name of the Log Service table.
-- logProjectName: the name of the Log Service project.
-- logStoreName: the name of the Logstore in Log Service.
-- accessKeyId: the AccessKey ID of your Apsara Stack tenant account.
-- accessKeySecret: the AccessKey secret of your Apsara Stack tenant account.
-- endpoint: the endpoint of the Logstore in Log Service.
-- When you specify a field in the Logstore, the field must be of the STRING type.
-- Reserve the following system fields: `__logProject__` (STRING), `__logStore__` (
STRING), `__shard__` (INT), `__time__` (TIMESTAMP), `__topic__` (STRING), and `__so
urce__` (STRING).
CREATE TABLE IF NOT EXISTS ${slsTableName} (col1 dataType[, col2 dataType])
USING loghub
OPTIONS (
sls.project = '${logProjectName}',
sls.store = '${logStoreName}',
access.key.id = '${accessKeyId}',
access.key.secret = '${accessKeySecret}',
endpoint = '${endpoint}');
-- Create an HDFS table and define the fields in the table.
-- hdfsTableName: the name of the HDFS table.
-- location: the data storage path. You can store data in HDFS or Object Storage Se
rvice (OSS).
-- Supported data formats: delta, csv, json, orc, and parquet. Default value: delta
.
CREATE TABLE IF NOT EXISTS ${hdfsTableName} (col1 dataType[, col2 dataType])
USING delta
LOCATION '${location}';
-- The method for reading the tables. Both the STREAM and BATCH methods are support
ed. The default method is BATCH.
CREATE SCAN tmp_read_sls_table
ON ${slsTableName}
USING STREAM;
-- Create a streaming query job.
CREATE STREAM ${queryName}
OPTIONS(
outputMode='Append',
triggerType='ProcessingTime',
triggerInterval='30000',
checkpointLocation='${checkpointLocation}')
INSERT INTO ${hdfsTableName}
SELECT col1, col2
FROM tmp_read_sls_table
WHERE ${condition};
```

For more information about EMR Streaming SQL, see EMR Streaming SQL.

iii. Save and run the EMR Streaming SQL node.

In the top toolbar, click the icon to save the EMR Streaming SQL node and click the icon to run the EMR Streaming SQL node.

5. Click the **Advanced Settings** tab in the right-side navigation pane. In the Advanced Settings panel, change the values of the parameters.
   - "USE_GATEWAY":true: If you set this parameter to true, the EMR Streaming SQL node is automatically committed to the master node of an EMR gateway cluster.
   - "SPARK_CONF": "--conf spark.driver.memory=2g --conf xxx=xxx": the parameters for running Spark jobs. You can specify multiple parameters in the format of --conf xxx=xxx.
   - "queue": the scheduling queue to which jobs are committed. Default value: default.
   - "priority": the priority. Default value: 1.
   - "FLOW_SKIP_SQL_ANALYZE": specifies how SQL statements are executed. A value of false indicates that only one SQL statement is executed at a time. A value of true indicates that multiple SQL statements are executed at a time.
6. Configure scheduling properties for the node.

   If you want the system to periodically run the node, you can click the **Properties** tab in the right-side navigation pane to configure scheduling properties for the node based on your business requirements.
   - Configure basic properties for the node. For more information, see Basic properties.
   - You can select a mode to start the node and a mode to rerun the node. For more information, see Scheduling properties.
7. Commit and deploy the node.

   i. Click the icon in the top toolbar to save the node.

   ii. Click the icon in the top toolbar to commit the node.

   iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

   iv. Click **OK**.

   If you use a workspace in standard mode, you must deploy the node to the production environment after you commit the node. Click **Deploy** in the upper-right corner. For more information, see Deploy nodes.
8. View the real-time computing node.

   i. Click **Operation Center** in the top navigation bar of the DataStudio page to go to Operation Center.

   ii. View the real-time computing node that is running. For more information, see Manage real-time computing nodes.

4.5.4. Hologres

4.5.4.1. Create a Hologres SQL node

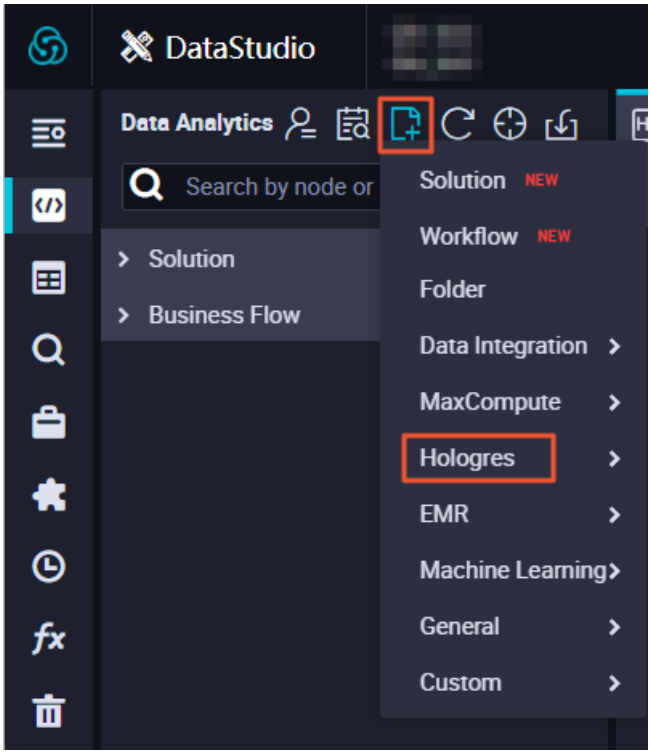
This topic describes how to create a Hologres SQL node. Hologres seamlessly integrates with MaxCompute at the underlying layer. This integration allows you to use standard PostgreSQL statements to query and analyze large volumes of data stored in MaxCompute. In this case, you do not need to transfer data. This allows you to quickly obtain query results.

Prerequisites

A **Hologres** compute engine instance is added on the **Project Management** page. This ensures that the **Hologres** folder is displayed on the page on which you want to create a Hologres SQL node.

Procedure

1. Log on to the DataWorks console.
2. Move the pointer over the +Create icon and choose **Hologres > Hologres SQL**.



3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

> **Note** The name of the node must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.
5. On the tab that appears, edit and run the code in the code editor.

   After the Hologres SQL node is created, edit the code in compliance with the required SQL syntax.
6. On the configuration tab of the node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, configure properties for the node. For more information, see Basic properties.
7. Save and commit the node.

> **Notice** You must set the **Rerun** and **Parent Nodes** parameters before you commit the node.

i. Click the icon in the top toolbar to save the node.

ii. Click the icon in the top toolbar.

iii. (Optional)In the **Commit Node** dialog box, enter your comments in the **Change description** field.

iv. Click **OK**.

If the workspace that you use is in standard mode, you must click **Deploy** in the top navigation bar after you commit the node. For more information, see Deploy nodes.

8. Test the node. For more information, see Manage auto triggered nodes.

4.5.5. AnalyticDB for PostgreSQL

4.5.5.1. Create an AnalyticDB for PostgreSQL node

You can create AnalyticDB for PostgreSQL nodes in the DataWorks console to build an online extract, transform, load (ETL) process.

Prerequisites

The AnalyticDB for PostgreSQL compute engine is associated with the workspace in which you want to create an AnalyticDB for PostgreSQL node. The AnalyticDB for PostgreSQL folder is displayed in a workspace only after you associate an AnalyticDB for PostgreSQL compute engine instance with the workspace on the **Workspace Management** page. For more information, see Create an AnalyticDB for PostgreSQL node.

Procedure

1. Log on to the DataWorks console.
2. On the **DataStudio** page, move the pointer over the +Create icon and choose **AnalyticDB > ADB for PostgreSQL**.

   Alternatively, you can find the required workflow, right-click the workflow name, and then choose **Create > AnalyticDB for PostgreSQL > ADB for PostgreSQL**.
3. In the **Create Node** dialog box, specify **Node Name** and **Location**.

   > **Note** The name of the node must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.
5. Configure the AnalyticDB for PostgreSQL node.

i. Select a data source from the **Select a connection** drop-down list.

> **Notice**
> - When you associate the AnalyticDB for PostgreSQL compute engine with the workspace, DataWorks automatically creates an AnalyticDB for PostgreSQL data source.
> - You can select a data source that is added only by using the connection string mode.

ii. Compile SQL statements.

After you select a data source, compile SQL statements based on the syntax that is supported by AnalyticDB for PostgreSQL.

iii. Click the icon in the top toolbar to save the SQL statements.

iv. Click the icon in the top toolbar to execute the SQL statements.

6. Click **Properties** in the right-side navigation pane. In the Properties panel, configure scheduling properties for the node. For more information, see Basic properties.
7. Save and commit the node.

> **Notice** You must specify the **Rerun** and **Parent Nodes** parameters in the Properties panel before you commit the node.

i. Click in the top toolbar to save the node.

ii. Click the icon in the top toolbar.

iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

iv. Click **OK**.

If the workspace that you use is in standard mode, you must click **Deploy** in the upper-right corner after you commit the AnalyticDB for PostgreSQL node. For more information, see Deploy nodes.

8. Test the node. For more information, see Manage auto triggered nodes.

4.5.5.2. Create an AnalyticDB for PostgreSQL table

This topic describes how to create an AnalyticDB for PostgreSQL table.

Prerequisites

- An AnalyticDB for PostgreSQL instance is associated with the workspace in which you want to create an AnalyticDB for PostgreSQL table. The AnalyticDB for PostgreSQL folder is displayed on the DataStudio page of a workspace only after you associate an AnalyticDB for PostgreSQL instance with the workspace on the **Project Management** page. For more information, see Create an AnalyticDB for PostgreSQL table.
- The metadata of the associated AnalyticDB for PostgreSQL instance is collected on the **Data Map** page. For more information, see Collect metadata from an AnalyticDB for PostgreSQL data source.

Procedure

1. Log on to the DataWorks console.
2. On the **DataStudio** page, move the pointer over the **+Create** icon and choose **AnalyticDB > table**.

   Alternatively, you can find your workflow in the Business Flow section, right-click **AnalyticDB for PostgreSQL**, and then choose **Create > ADB visual table creation**.
3. In the **Create Table** dialog box, set **Table Name**.

   > **Notice**
   > - The table name must be in the format of schema_name.table_name.
   > - The values of schema_name and table_name must be 1 to 63 characters in length and can contain letters, digits, and underscores (_). The values must start with a letter or underscore (_).
   > - If you associate multiple AnalyticDB for PostgreSQL instances with the current workspace, you must select one based on your business requirements.
4. Click **Commit**. The table configuration tab appears.

   The upper part of the table configuration tab displays the table name and AnalyticDB for PostgreSQL instance name.
5. In the **General** section, set the parameters.

Parameter	Description
Level 1 theme	The name of the level-1 folder where the table resides. **Note** Level-1 and level-2 folders show the table locations in DataWorks for you to manage tables more conveniently.
Level 2 theme	The name of the level-2 folder where the table resides.
Create a theme	Click **Create a theme** to go to the **Folder Management** tab. On this tab, you can create level-1 and level-2 folders for tables. After you create a folder, click the icon next to Create Folder to synchronize the folder.
Description	The description of the table.

6. In the **Physical model design** section, set the parameters.

Parameter	Description

Parameter	Description
Level selection	The layer where the table data is stored or processed. A data warehouse consists of the operational data store (ODS), common data model (CDM), and application data store (ADS) layers. You can customize a name for each layer.
Physical classification	The category of the table. Tables are categorized into basic services, advanced services, and other services. You can customize a name for each category. **Note** Categories are designed only for your management convenience and do not involve underlying implementation.
New Level	The levels and categories that you want to create. To add levels and categories, click **New Level** to go to the **Hierarchical management** tab. After levels and categories are created, click the icon.

7. In the **AnalyticDB for PostgreSQL table design** section, set the parameters.

   You can configure the schema of an AnalyticDB for PostgreSQL table on the following tabs: **Column information settings**, **Index settings**, **Sub-table design**, and **Partition settings (optional)**.

Tab	Parameter	Description
Column information settings	**New columns**	Allows you to click the button and set the relevant parameters to create a field.
	Name	The name of the field.
	Field type	The data type of the field.
	Field length	The length of the field. You can specify the length for fields only of some specific data types.
	Default value	The default value of the field.
	Allow to be empty	Specifies whether the field can be empty.
	Is it the primary key?	Specifies whether the field serves as the primary key.
	Foreign key	Specifies whether the field serves as a foreign key.
	Operation	◦ You can perform the following operations on a new field: save, cancel, delete, move up, and move down. ◦ You can perform the following operations on an existing field: modify, delete, move up, and move down.

Tab	Parameter	Description
Index settings	**New columns**	Allows you to click the button and set the relevant parameters to create an index.
	Index name	The name of the index. Make sure that you specify a unique name.
	Include columns	The field on which the index will be created. To select a field, click **Edit**. In the **Select at least one index** dialog box, click the **+** icon. All the created fields appear in the Column information drop-down list. Select the field from the **Column information** drop-down list and click **Save**.
	Index type	The type of the index. Valid values: **Normal**, **Primary Key**, and **Unique**.
	Index mode	The mode for indexing data in the fields. Valid values: **B-tree**, **Bitmap**, and **GiST**.
	Operation	◦ You can perform the following operations on a new index: save, cancel, delete, move up, and move down. ◦ You can perform the following operations on an existing index: modify, delete, move up, and move down.
Sub-table design	**Hash (Recommended)**, **Copy Schema**, and **Random (Not Recommended)**	The way in which the partition key is generated. Take **Hash (Recommended)** as an example. Click **New columns** and select the target field from the **Name** drop-down list. The information about the selected field appears. Click **Save**. For more information, see the **Column information settings** section of this table.
Partition settings (optional)	**Partition settings (optional)**	The partitions of the table. You can configure the partitions based on your business requirements.

8. Click **Submit to development environment** and **Submit to production environment** in sequence.

   If you are using a workspace in basic mode, you need only to click **Submit to production environment**.

9. In the **Submit changes** dialog box, confirm that the table creation statements are correct, select a resource group from the **Select a resource group** drop-down list, and then click **Confirm execution**.

What's next

After the AnalyticDB for PostgreSQL table is created, you can query the table data, modify the table, or delete the table. For more information, see Manage tables.

4.5.6. AnalyticDB for MySQL

4.5.6.1. Create and use an AnalyticDB for MySQL node

You can create an AnalyticDB for MySQL node and use SQL statements to develop data for an AnalyticDB for MySQL data source. This topic describes how to create and use an AnalyticDB for MySQL node.

Prerequisites

- An AnalyticDB for MySQL instance is purchased and associated with the current DataWorks workspace in the **Workspace Management** page of DataWorks. For more information, see Associate an AnalyticDB for MySQL instance with a workspace.
- A workflow is created. For more information, see Create a workflow.

Create an AnalyticDB for MySQL node and use the node to develop data

1. Log on to the DataWorks console.
2. On the **DataStudio** page, move the pointer over the **+Create** icon and choose **AnalyticDB for MySQL > ADB for MySQL**.

   You can also find the required workflow, right-click the workflow name, and then choose **Create > AnalyticDB for MySQL > ADB for MySQL**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The name of the node must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. Configure the AnalyticDB for MySQL node.
   i. Select a data source from the **Select Data Source** drop-down list.

   > **Notice**
   > - When you associate the AnalyticDB for MySQL compute engine instance with the workspace, DataWorks automatically adds an AnalyticDB for MySQL data source.
   > - You can select a data source that is added only by using the connection string mode.

   ii. Write the SQL statements of the node.

   After you select a data source, write the SQL statements of the node based on your business requirements.

   iii. Click the icon in the top toolbar to save the SQL statements.

   iv. Click the icon in the top toolbar to execute the SQL statements.
5. Click the **Properties** tab in the right-side navigation pane. In the Properties panel, configure scheduling properties for the node. For more information, see Basic properties.

6. Save and commit the node.

   > **Notice** You must set the **Rerun** and **Parent Nodes** parameters in the Properties panel before you commit the node.

   i. Click the icon in the top toolbar to save the node.

   ii. Click the icon in the top toolbar.

   iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

   iv. Click **OK**.

   If the workspace that you use is in standard mode, you must click **Deploy** in the upper-right corner after you commit the AnalyticDB for MySQL node. For more information, see Deploy nodes.

7. Test the node. For more information, see Manage auto triggered nodes.

4.5.7. Algorithm

4.5.7.1. Create a PAI node

PAI nodes are used to call tasks that are created on PAI and schedule production activities based on the node configuration.

Prerequisites

To create a PAI node in DataWorks, you must first create a PAI experiment in PAI.

Procedure

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over +Create and choose **Machine Learning > PAI Experiment**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **Algorithm**, and then choose **Create > PAI Experiment**.

3. In the **Create Node** dialog box, specify **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.
5. Select the PAI experiment that you have created from the **Experiment** drop-down list and load it.

   If you want to modify the PAI experiment, click **Edit in PAI Console**.

6. On the configuration tab of the batch synchronization node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, configure parameters in the Schedule section. For more information, see Basic properties.
7. Commit the node.

   i. Click in the toolbar.

ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.

iii. Click **OK**.

In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Publish nodes.

8. Test the node. For more information, see Manage auto triggered nodes.

4.5.8. General

4.5.8.1. Create a for-each node

This topic describes how to use a for-each node to repeat a loop twice and display the loop count.

Prerequisites

The MaxCompute module is available on the DataStudio page only after you bind a MaxCompute compute engine to the current workspace on the **Project Management** page.

Context

You can use a for-each node to repeat a loop for a maximum of 128 times. If the loop count exceeds this limit, an error occurs.

If the for-each node needs to perform logic judgment and result traversal, you can use the branch node. However, the branch node must be used with the merge node for result traversal.

Procedure

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over 

and choose **General > for-each**.

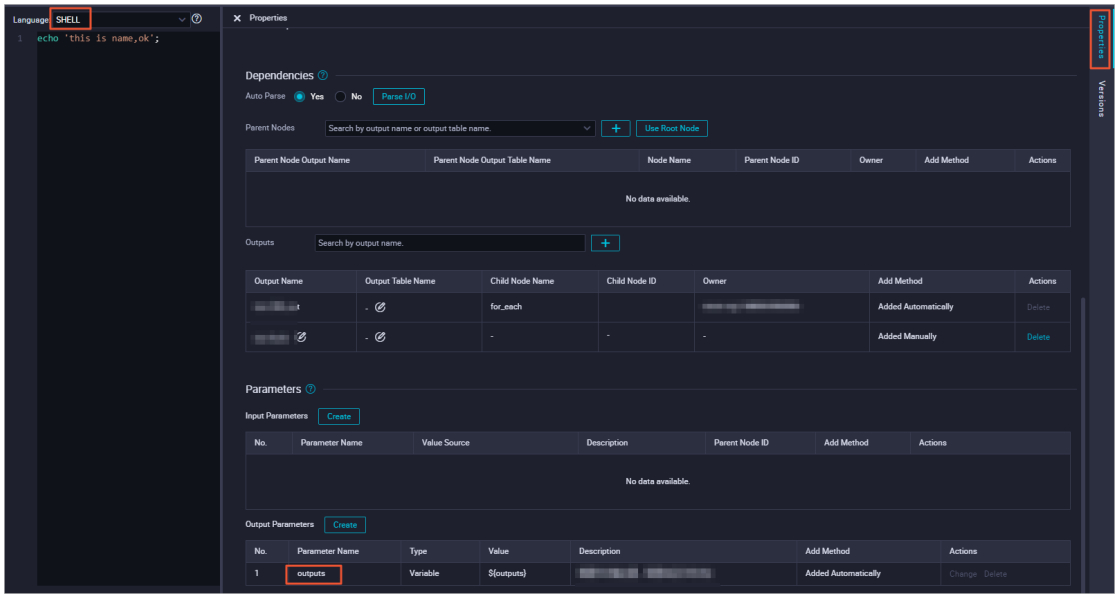
   Alternatively, you can click a workflow in the Business Flow section, right-click **General**, and then choose **Create > for-each**.
3. In the **Create Node** dialog box, specify **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.
5. Create a workflow with an assignment node as the parent node and a for-each node as the child node. For more information, see Create a workflow.

i. Double-click the created assignment node. Set the language of the assignment node to SHELL, and enter the following code:

```
echo 'this is name,ok';
```

On the node configuration tab, click the **Properties** tab in the right-side navigation pane. By default, the **outputs** parameter appears in the **Output Parameters** section.



ii. Double-click the created for-each node. Enter the following code for the for-each node:

```
echo ${dag.loopTimes} ----Display the loop count.
```

> **Note**
> - The start and end nodes of the for-each node have fixed logic and cannot be edited.
> - After you modify the code of the Shell node, save the modification. No message will appear to remind you to save the modification when you commit the node. If you do not save the modification, the code cannot be updated to the latest version in time.

A for-each node supports the following environment variables:

- ${dag.foreach.current}: the current data row.
- ${dag.loopDataArray}: the input dataset.
- ${dag.offset}: the offset of the loop count to 1.
- ${dag.loopTimes}: the loop count, whose value equals to the value of ${dag.offset} plus 1.

```
// Compare the code of the Shell node with that of a common for loop.
data=[]  // It is equivalent to ${dag.loopDataArray}.
// i is equivalent to ${dag.offset}.
for(int i=0;i<data.length;i++) {
  print(data[i]);  // data[i] is equivalent to ${dag.foreach.current}.
}
```

The ${dag.loopDataArray} parameter is the default input parameter of the for-each node. Set this parameter to the value of the outputs parameter of the parent node. If you do not set this parameter, an error occurs when you commit the node.

6. On the configuration tab of the batch synchronization node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, configure parameters in the Schedule section. For more information, see Basic properties.
7. Commit the node.

> **Notice** You can commit the node only after you specify the **Rerun** and **Parent Nodes** parameters.

i. Click the icon in the toolbar.

ii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

iii. Click **OK**.

In a workspace in standard mode, you must click **Publish** in the upper-right corner after you commit the node. For more information, see Publish nodes.

8. Test the node. For more information, see Manage auto triggered nodes.

4.5.8.2. Create a do-while node

You can define mutually dependent nodes, including a loop decision node named end, in a do-while node. DataWorks repeatedly runs the nodes and exits the loop only when the end node returns False.

Context

> **Note** A loop can be repeated for a maximum of 128 times. If the loop count exceeds this limit, an error occurs.

The do-while node supports the MaxCompute SQL, SHELL, and Python languages. If you use MaxCompute SQL, you can use a `CASE WHEN` statement to evaluate whether the specified condition for exiting the loop is met.

Simple example

This section describes how to use a do-while node to repeat a loop five times and display the loop count each time the loop runs.

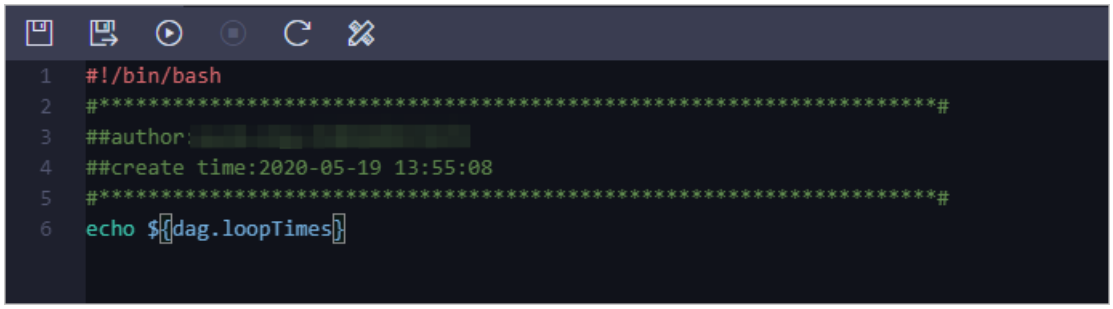
1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over **+Create** and choose **General > do-while**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **General**, and then choose **Create > do-while**.
3. In the **Create Node** dialog box, specify **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.
5. Define the loop body.

   By default, the do-while node consists of the start, SQL, and end nodes.

   - The start node marks the startup of a loop and does not have any business effect.
   - DataWorks provides the SQL node as a sample business processing node. You must replace the SQL node with your own business processing node, for example, a Shell node named Display loop count.

   - The end node marks the end of a loop and determines whether to start the loop again. In this example, it defines the condition for exiting the loop for the do-while node.

     The end node is an assignment node. It generates only True or False, indicating whether to start the loop again or exit the loop.

The ${dag.loopTimes} variable is used in both the Display loop count node and the end node. It is a reserved variable of DataWorks. This variable indicates the loop count and the value increments from 1. All internal nodes of the do-while node can reference this variable.

In the code shown in the preceding figure, the value of the `dag.loopTimes` variable is compared with 5 to limit the loop count. The value of the dag.loopTimes variable is 1 when the loop runs for the first time and is incremented by 1 each time, for example, 2 for the second time. In the fifth loop, the value is 5. In this case, the result of ${dag.loopTimes}<5 is False, and the do-while node exits the loop.

6. Run the do-while node.

   You can configure the scheduling properties for the do-while node as needed and commit it to **Operation Center** for running.

   - do-while node: The do-while node appears as a whole node in Operation Center. To view the loop details about the do-while node, right-click the node in the DAG and select **View Internal Nodes**.
   - Internal loop body: This view is divided into three parts.
     - The left pane of the view lists the rerun history of the do-while node. A record is generated each time a do-while node instance is run.
     - The middle pane of the view shows a loop record list. A record is generated each time the loop of the do-while node is run. The running status of each loop also appears.
     - The right pane of the view shows the details about the do-while node each time the loop is run. You can click a record in the loop record list to view the running details.

7. View the running result.

   View the internal loop body. In the loop record list, click the record corresponding to the third loop. The loop count is 3 in the runtime logs.

   You can also view the runtime logs of the end node that are generated when the loop runs for the third time and for the fifth time, respectively.

   Based on the preceding simple example, the do-while node works in the following way:

   i. Run from the start node.
   ii. Run nodes in sequence based on the defined node dependencies.
   iii. Define the condition for exiting the loop in the end node.
   iv. Run the conditional statement of the end node after the loop ends for the first time.
   v. Record the loop count as 1 and start the loop again if the conditional statement returns True in the runtime logs of the end node.
   vi. Exit the loop if the conditional statement returns False in the runtime logs of the end node.

Complex example

In addition to simple scenarios, do-while nodes can also be used in complex scenarios where each row of data is processed in sequence by using a loop. Before you process data in such scenarios, make sure that:

- You have deployed a parent node that can export queried data to the do-while node. You can use an assignment node to meet this condition.
- The do-while node can obtain the output of the parent node. You can configure the node context and dependencies to meet this condition.

- The internal nodes of the do-while node can reference each row of data. In this example, the existing node context is enhanced and the system variable ${dag.offset} is used to reference the context of the do-while node.

This section describes how to use the do-while node to display the data entries in a table in sequence until all data entries in the table are displayed. Each time the loop runs, a data entry is displayed.

1. On the **Data Analytics** tab, double-click the created do-while node.
2. Define the loop body.
   i. Create an assignment node named Initialize dataset and add it as the parent node of the do-while node. The parent node generates a test dataset.
   ii. On the Properties tab of the do-while node, define an input parameter in the **Parameters** section. Set Parameter Name to input and Value Source to the output of the parent node.
   iii. Write code for the business processing node named Print each data row.
      - `${dag.offset}`: a reserved variable of DataWorks. This variable indicates the offset of the loop count to 1. For example, the offset is 0 when the loop runs for the first time and 1 for the second time. The offset equals to the loop count minus 1.
      - `${dag.input}`: the context that you configure for the do-while node. In the preceding steps, the input parameter is defined for the do-while node and the value of the input parameter is the output of the parent node named Initialize dataset.

        The internal nodes of the do-while node can directly use ${dag.${ctxKey}} to reference the context. In this example, ${ctxKey} is set to input. Therefore, you can use ${dag.input} to reference the context.
      - `${dag.input[${dag.offset}]}`: the data obtained from the table generated by the Initialize dataset node. DataWorks can obtain a row of data from the table based on the specified offset. The value of the ${dag.offset} variable increments from 0. Therefore, the data entries such as ${dag.input[0]} and ${dag.input[1]} are returned until all data entries in the dataset are returned.
   iv. Define the condition for exiting the loop for the end node. The values of the ${dag.loopTimes} and ${dag.input.length} variables are compared, as shown in the following figure. If the value of the former is less than that of the latter, the end node returns True and the do-while node continues the loop. Otherwise, the end node returns False and the do-while node exits the loop.

```
Language: Python
1  if ${dag.loopTimes}<${dag.input.length}:
2      print True;
3  else
4      print False;
```

> **Note** The system automatically sets the ${dag.input.length} variable to the number of rows in the array specified by the input parameter based on the context configured for the do-while node.

3. Run the do-while node and view the running result.

Summary

- Compared with the while, foreach, and do...while statements, a do-while node has the following characteristics:
  - A do-while node contains a loop body that runs a loop before evaluating the conditional statement. This node functions the same as the do...while statement. A do-while node can also use the system variable ${dag.offset} and the node context to implement the feature of the foreach statement.
  - A do-while node cannot achieve the feature of the while statement because a do-while node runs a loop before evaluating the conditional statement.
- A do-while node works in the following way:
  i. Run nodes in the loop body starting from the start node based on node dependencies.
  ii. Run the code defined for the end node.
    - Run the loop again if the end node returns True.
    - Exit the loop if the end node returns False.
- How to use the node context: The internal nodes of a do-while node can use ${dag.${ctxKey}} to reference the context defined for the do-while node.
- System parameters: DataWorks provides the following system variables for the internal nodes of the do-while node:
  - ${dag.loopTimes}: the loop count, starting from 1.
  - ${dag.offset}: the offset of the loop count to 1, starting from 0.

4.5.8.3. Create a merge node

This topic describes the definition of merge nodes and how to create a merge node and define the merging logic. It also provides an example to show the scheduling configuration and running details of a merge node.

A merge node is a logical control node in DataStudio. It can merge the running results of its parent nodes, regardless of their running statuses. It aims at facilitating the running of nodes that depend on the output of the child nodes of a branch node.

You cannot change the running status of a merge node. A merge node merges the running results of multiple child nodes of a branch node and sets the running status to Successful. To guarantee the proper running of a node that depends on the output of the child nodes of a branch node, you can configure the node to directly depend on the merge node.

For example, Branch node C has two logically exclusive branches C1 and C2. These two branches use different logic to write data to the same MaxCompute table. Assume that Node B depends on the output of this MaxCompute table. To make sure that Node B can run properly, you must use Merge node J to merge the running results of branches C1 and C2, and then configure Merge node J as the parent node of Node B. If Node B directly depends on branches C1 and C2, one of the branches will fail to run because only one branch meets the branch condition each time Branch node C runs. In this case, Node B cannot be triggered as scheduled.

Create a merge node

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over +Create and choose **General > MERGE Nodes**.

Alternatively, you can click a workflow in the Business Flow section, right-click **General**, and then choose **Create > MERGE Nodes**.

3. In the **Create Node** dialog box, set **Node Name** and **Location**.

> **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.

4. Click **Commit**.

Define the merging logic

After the merge node is created, the node configuration tab appears. Specify the branches to be merged for the node. Enter the output name or output table name of the parent node, and click the **Add** icon. You can view the running status in the Result section. The available running statuses are **Successful** and **Branch Not Running**.

Click the **Properties** tab in the right-side navigation pane and configure the scheduling properties of the merge node.

Run the merge node

If a branch meets the specified condition, the branch is run. You can select the branch and view the running details on the **Runtime Logs** tab.

If a branch does not meet the specified condition, the branch is skipped. You can select the branch and view related information on the **Runtime Logs** tab.

4.5.8.4. Create a branch node

A branch node is a logical control node in DataStudio. It can define the branch logic and the direction of branches under different logical conditions.

Prerequisites

Generally, branch nodes need to be used with assignment nodes.

Create a branch node

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over 

and choose **General > Branch Node**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **General**, and then choose **Create > Branch Node**.
3. In the **Create Node** dialog box, set **Node Name** and **Location**.

> **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.

4. Click **Commit**.
5. Define the branch logic
   i. In the **Definition** section, click **Add Branch**.

ii. In the **Branch Definition** dialog box, set the parameters.

Parameter	Description
Condition	The condition of the branch. ■ You can only use Python comparison operators to define logical conditions for the branch node. ■ If the result of the expression is *true* when the node is running, the corresponding branch condition is met. ■ If the expression fails to be parsed when the node is running, the whole branch node fails. ■ To define branch conditions, you can use global variables and parameters defined in the node context. For example, the ${input} variable can be used as an input parameter of the branch node.
Associated Node Output	The associated node output of the branch. ■ The node output is used to configure dependencies for the child nodes of the branch node. ■ If the branch condition is met, the child node corresponding to the node output is run. If the child node also depends on the output of other nodes, the status of these nodes is considered. ■ If the branch condition is not met, the child node corresponding to the node output is not run. The child node is set to the `Not Running` state.
Description	The description of the branch. For example, the branches ${input}==1 and ${input}>2 are defined.

iii. Click **OK**.

After you add a branch, you can click **Change** or **Delete** in the Actions column of the branch to modify or delete it.

- Click **Change** to modify the branch and related dependencies.
- Click **Delete** to delete the branch and related dependencies.

6. On the configuration tab of the branch node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, set parameters in the Schedule section.

After the branch conditions are defined, the output names are automatically added to the **Outputs** section on the **Properties** tab. Then, you can associate child nodes with the branch node based on the output names.

> **Note**
>
> - Child nodes inherit dry-run properties of the parent node. Therefore, we recommend that you do not create a node depending on its last-cycle instance as the branch.
> - The dependencies established by drawing lines between nodes on the dashboard of a workflow are not recorded on the Properties tab. You must manually enter these dependencies.

7. Commit the node.

   > **Notice** You must set **Rerun** and **Parent Nodes** before you can commit the node.

   i. Click in the toolbar.

   ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.

   iii. Click **OK**.

   In a workspace in standard mode, you need to click **Deploy** in the upper-right corner after you commit the branch node.

8. Test the node.

Supported Python comparison operators

In the following table, assume that the value of the a variable is 10 and that of the b variable is 20.

Comparison operator	Description	Example
==	Equal: checks whether two objects are equal.	(a==b) returns false.
! =	Not equal: checks whether two objects are not equal.	(a! =b) returns true.
<>	Not equal: checks whether two objects are not equal.	(a<>b) returns true. This operator is similar to ! =.
>	Greater than: checks whether the variable on the left side of the operator is greater than that on the right side.	(a>b) returns false.
<	Less than: checks whether the variable on the left side of the operator is less than that on the right side. If the return result is 0 or 1, 0 indicates false and 1 indicates true. These two results are equivalent to the special variables true and false, respectively.	(a<b) returns true.
>=	Greater than or equal to: checks whether the variable on the left side of the operator is greater than or equal to that on the right side.	(a>=b) returns false.
<=	Less than or equal to: checks whether the variable on the left side of the operator is less than or equal to that on the right side.	(a<=b) returns true.

4.5.8.5. Create an assignment node

An assignment node uses one of the three value assignment languages MaxCompute SQL, SHELL, and Python to assign values by using the outputs parameter. This node is used to transmit data between a parent node and a child node based on context-based parameters.

Context

The outputs parameter has the following limits:

- The value of the outputs parameter is taken only from the output of the last line of the code.
  - If you use MaxCompute SQL, the output of the SELECT statement in the last line is used.
  - If you use SHELL, the output of the ECHO statement in the last line is used.
  - If you use Python, the output of the PRINT statement in the last line is used.
- The passed value of the outputs parameter is limited to 2 MB in size. If the output of the assignment statement exceeds this limit, the assignment node fails to run.

> **Note** If you use Python or SHELL, the value of the outputs parameter is a one-dimensional array where elements are separated with commas (,). If you use MaxCompute SQL, the value of the outputs parameter is passed to child nodes as a two-dimensional array.

Create an assignment node

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over +Create and choose **General > Assignment Node**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **General**, and then choose **Create > Assignment Node**.
3. In the **Create Node** dialog box, set **Node Name** and **Location**.

   > **Notice** A node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.

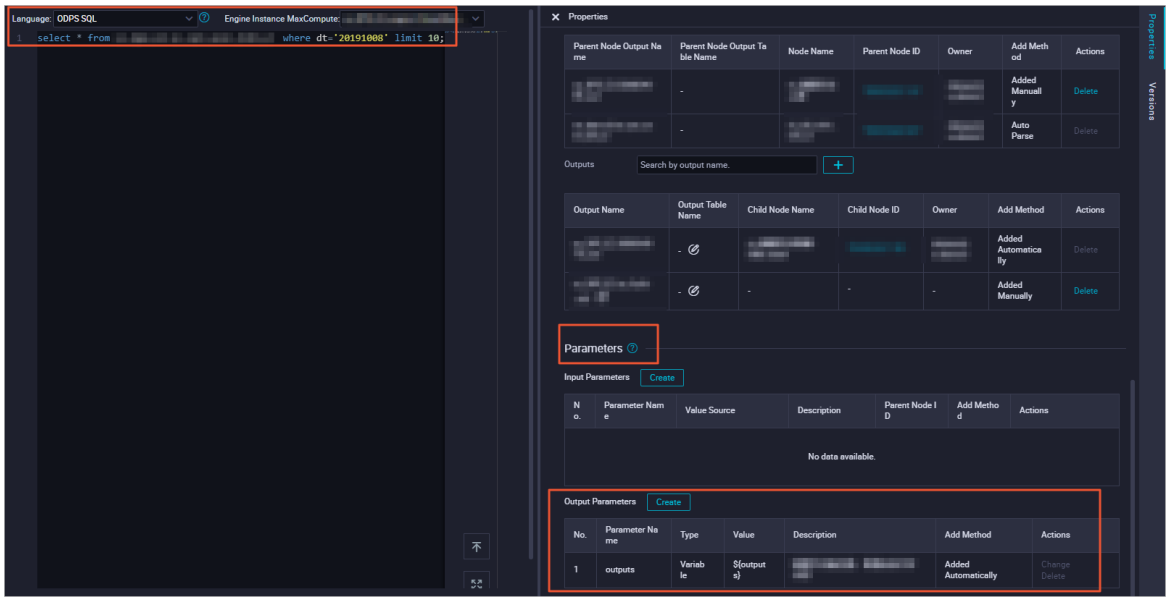
   This following sections describe how to use assignment nodes that use the Python, MaxCompute SQL, and SHELL languages respectively to pass data between a parent node and a child node named Assignment node value comparison_shell by using context-based parameters.

   After the assignment nodes that use the Python, MaxCompute SQL, and SHELL languages are created, you must set the dependencies so that the child node can reference the parameter values passed by these nodes.
4. Click **Commit**.

Configure the child node to reference the output values of the assignment node that uses MaxCompute SQL

1. Find the target workflow and double-click the assignment node fuzhi_sql that uses MaxCompute SQL.
2. On the configuration tab of the fuzhi_sql node that appears, click **Properties** in the right-side navigation pane.
3. Configure the fuzhi_sql node.

The fuzhi_sql node assigns the results queried from a specified table to the outputs parameter.



4. Double-click the Assignment node value comparison_shell node, which is the child node of the fuzhi_sql node.
5. On the configuration tab of the Assignment node value comparison_shell node that appears, click **Properties** in the right-side navigation pane and configure the node.

   The Assignment node value comparison_shell node depends on the fuzhi_sql node and uses the value of the outputs parameter of the fuzhi_sql node as the value of its input parameter sql_inputs.

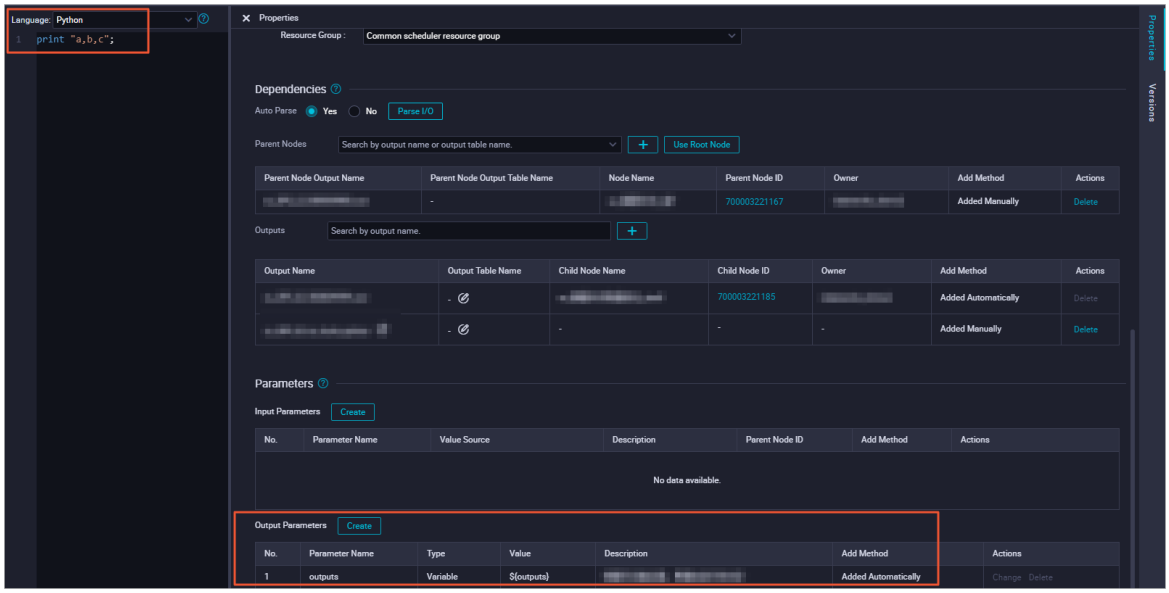
```
echo '${sql_inputs}';
echo 'Use the value in the first line in the output of the fuzhi_sql node as the input'
${sql_inputs[0]};
echo 'Use the value in the second line in the output of the fuzhi_sql node as the input
'${sql_inputs[1]};
echo 'Use the value of the second field in the first line in the output of the fuzhi_sq
l node as the input'${sql_inputs[0][1]};
echo 'Use the value of the third field in the second line in the output of the fuzhi_sq
l node as the input'${sql_inputs[1][2]};
```

6. Click in the toolbar.
7. In the **Warning** message, click **Continue to Run**.
8. View the result.

Configure the child node to reference the output values of the assignment node that uses Python

1. Find the target workflow and double-click the assignment node fuzhi_python that uses Python.
2. On the configuration tab of the fuzhi_python node that appears, click **Properties** in the right-side navigation pane.
3. Configure the fuzhi_python node.

   The fuzhi_python node assigns the values a,b,c to the outputs parameter.



4. Double-click the Assignment node value comparison_shell node, which is the child node of the fuzhi_python node.
5. On the configuration tab of the Assignment node value comparison_shell node that appears, click **Properties** in the right-side navigation pane and configure the node.

   The Assignment node value comparison_shell node depends on the fuzhi_python node and uses the value of the outputs parameter of the fuzhi_python node as the value of its input parameter python_inputs.

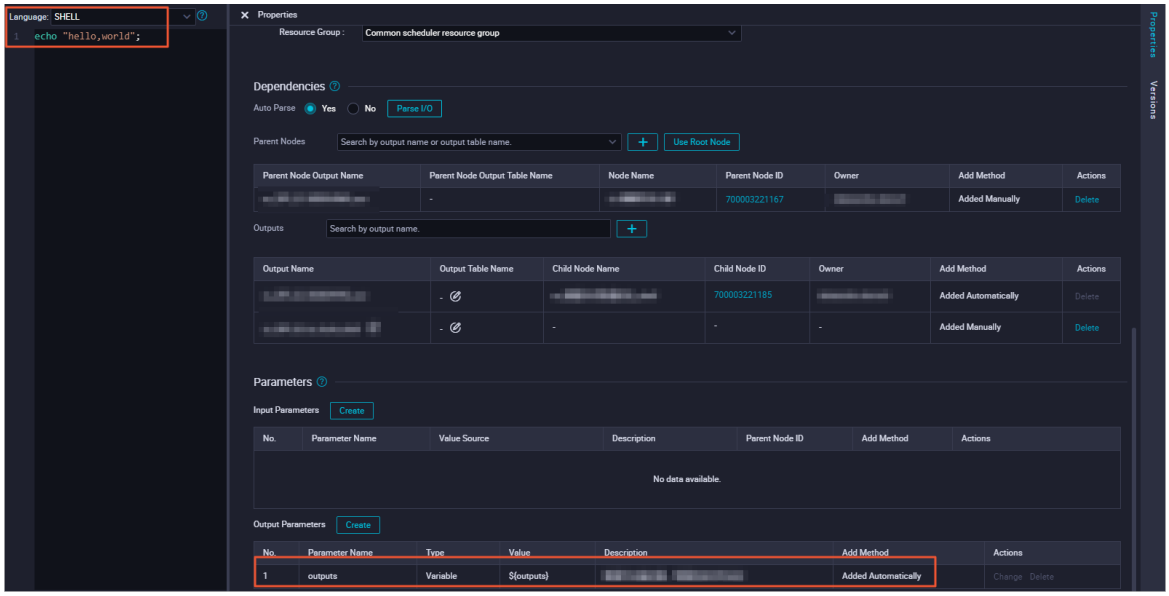
   ```
   echo 'The output of the fuzhi_python node'${python_inputs};
   echo 'Use the first value in the output of the fuzhi_python node as the input'${python_
   inputs[0]};
   echo 'Use the second value in the output of the fuzhi_python node as the input'${python
   _inputs[1]};[1]}
   ```

6. Click  in the toolbar.
7. In the **Warning** message, click **Continue to Run**.
8. View the result.

Configure the child node to reference the output values of the assignment node that uses SHELL

1. Find the target workflow and double-click the assignment node fuzhi_shell that uses SHELL.
2. On the configuration tab of the fuzhi_shell node that appears, click **Properties** in the right-side navigation pane.
3. Configure the fuzhi_shell node.

   The fuzhi_shell node assigns the values hello,world to the outputs parameter.



4. Double-click the Assignment node value comparison_shell node, which is the child node of the fuzhi_shell node.
5. On the configuration tab of the Assignment node value comparison_shell node that appears, click **Properties** in the right-side navigation pane and configure the node.

   The Assignment node value comparison_shell node depends on the fuzhi_shell node and uses the value of the outputs parameter of the fuzhi_shell node as the value of its input parameter shell_inputs.

   ```
   echo 'The output of the fuzhi_shell node'${shell_inputs};
   echo 'Use the first value in the output of the fuzhi_shell node as the input'${shell_inputs[0]};
   echo 'Use the second value in the output of the fuzhi_shell node as the input'${shell_inputs[1]};
   ```

6. Click ▶ in the toolbar.
7. In the **Warning** message, click **Continue to Run**.
8. View the result.

4.5.8.6. Create a Shell node

Shell nodes support standard shell syntax but not interactive syntax.

Procedure

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over **+Create** and choose **General > Shell**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **General**, and then choose **Create > Shell**.
3. In the **Create Node** dialog box, set **Node Name** and **Location**.

> **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.

4. Click **Commit**.
5. Edit the Shell node.
   i. Edit the code on the configuration tab of the Shell node.

      To call the system scheduling parameters for the Shell node, execute the following statement:

      ```
      echo "$1 $2 $3"
      ```

      > **Note** Separate multiple parameters with spaces.

   i. Click in the toolbar to save the SQL statement to the server.

   ii. Click in the toolbar to execute the SQL statement you have saved.

      If you need to change the resource group used to test the Shell node on the **DataStudio** page, click in the toolbar and select your desired exclusive resource group.

6. On the configuration tab of the Shell node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, set parameters in the Schedule section.
7. Commit the node.

   > **Notice** You must set **Rerun** and **Parent Nodes** before you can commit the node.

   i. Click in the toolbar.

   ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.

   iii. Click **OK**.

   In a workspace in standard mode, you need to click **Deploy** in the upper-right corner after you commit the branch node.
8. Test the node.

4.5.8.7. Create a zero-load node

A zero-load node is a control node, which only supports dry-run scheduling and does not generate any data. It usually serves as the root node of a workflow.

Context

You can configure an output table for a zero-load node so that the output table can be used as an input table of another node. However, the zero-load node does not process the table data.

Procedure

1. Log on to the DataWorks console.
2. On the **Data Analytics** tab, move the pointer over +Create and choose **General > Zero-Load**

Node.

Alternatively, you can click a workflow in the Business Flow section, right-click **General**, and then choose **Create > Zero-Load Node**.

3. In the **Create Node** dialog box, set **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.

4. Click **Commit**.
5. On the configuration tab of the zero-load node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, set parameters in the Schedule section. For more information, see Basic properties.
6. Commit the node.

   > **Notice** You must set **Rerun** and **Parent Nodes** before you can commit the node.

   i. Click in the toolbar.

   ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.

   iii. Click **OK**.

   In a workspace in standard mode, you need to click **Deploy** in the upper-right corner after you commit the branch node.

7. Test the node.

4.5.8.8. Create a cross-tenant collaboration node

Cross-tenant collaboration nodes are used to associate nodes from different tenants. Cross-tenant collaboration nodes are classified into sender nodes and receiver nodes.

Prerequisites

A sender node and its receiver node use the same CRON expression. You can click the **Properties** tab in the right-side navigation pane of a node configuration tab and view the CRON expression in the **Schedule** section.

Create a cross-tenant collaboration node

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over Create and choose **General > Cross-Tenant Collaboration**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **General**, and then choose **Create > Cross-Tenant Collaboration**.

3. In the **Create Node** dialog box, set **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.

4. On the node configuration tab, set the parameters in the **Cross-Tenant Collaboration** section.

Parameter	Description
Type	The type of the cross-tenant collaboration node. Valid values: **Sender** and **Receiver**.
Location	The path of the cross-tenant collaboration node. The node path cannot be modified.
Collaborative Workspaces	The workspace name and Apsara Stack tenant account of the peer node. This example sets the node type to Sender. Therefore, you must enter the workspace name and Apsara Stack tenant account of the receiver node.

5. After the sender node is created, follow the same procedure to create the receiver node under the Apsara Stack tenant account and workspace to which the receiver node belongs.

   Set the node type to **Receiver**. The information about available sender nodes appears. You must also set **Timeout**. This parameter indicates the timeout period of the receiver node after it starts running.

   The sender node first sends a message to the message center. After the message is delivered, the status of the sender node is set to successful. The receiver node continuously pulls messages from the message center. If a message is received within the timeout period, the status of the receiver node is set to successful.

   If the receiver node does not receive any messages within the timeout period, the receiver node fails. The lifecycle of a message is 24 hours.

   Assume that an auto triggered instance was run on October 8, 2018. A message indicating the completion of the instance was then sent to the message center. If you create a retroactive instance for the receiver node with the data timestamp set to October 7, 2018, the status of the generated receiver node instance is set to successful.

6. After the configuration is completed, save and commit the node.

4.5.8.9. Create a data analysis report node

A data analysis report node is used to associate a report in the DataAnalysis module with the parent nodes on which the report depends and update the report as scheduled.

Prerequisites

A table is created on the **Report** page of the **DataAnalysis** module.

Procedure

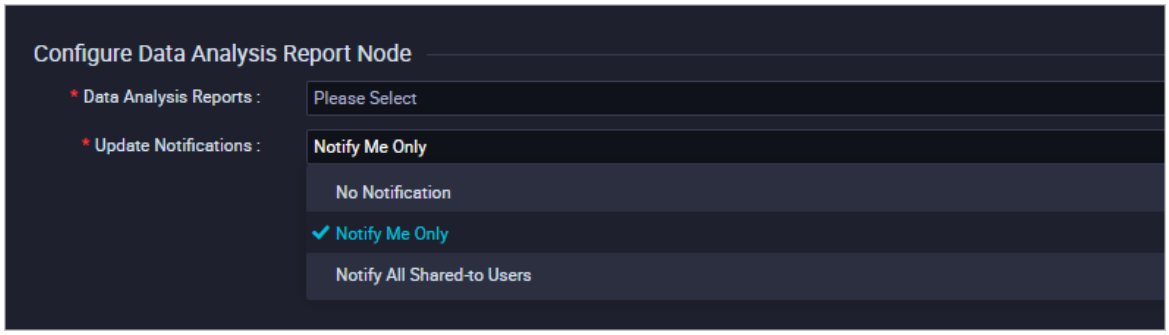
1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over 

and choose **General > Data Analysis Reports**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **General**, and then choose **Create > Data Analysis Reports**.
3. In the **Create Node** dialog box, specify **Node Name** and **Location**.

> **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.
5. On the node configuration tab, set the parameters in the **Configure Data Analysis Report Node** section.



Parameter	Description
Data Analysis Reports	The report for which you want to receive the notifications about the updates. Select a report created on the **Report** page of the **DataAnalysis** module.
Update Notifications	Specifies the users who can receive the notifications when the report is updated. Valid values: **No Notification**, **Notify Me Only**, and **Notify All Shared to Users**.

6. On the configuration tab of the batch synchronization node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, configure parameters in the Schedule section. For more information, see Basic properties.
7. Commit the node.
   i. Click in the toolbar.
   ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.
   iii. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Publish nodes.
8. Test the data analysis report node. For more information, see Manage auto triggered nodes.

   After the node is run in the production environment, you can view updates of the report on the Report page of the **DataAnalysis** module.

4.5.9. Custom

4.5.9.1. Create a Hologres development node

This topic describes how to create and modify a Hologres development node and update the node version.

Procedure

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over +Create and choose **Custom > Hologres Development**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **UserDefined**, and then choose **Create > Hologres Development**.
3. In the **Create Node** dialog box, specify **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.
5. On the node configuration tab that appears, select a Hologres development node.

   If no Hologres node is available, click **Create** to create one. You can also click **Change** to modify an existing node.
6. On the configuration tab of the batch synchronization node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, configure parameters in the Schedule section. For more information, see Basic properties.
7. Commit the node.
   i. Click in the toolbar.
   ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.
   iii. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Publish nodes.
8. Test the node. For more information, see Manage auto triggered nodes.

4.6. Schedule

4.6.1. Basic properties

On the Properties tab of a node, you can set parameters of the node in the General, Schedule, Dependencies, and Parameters sections. The General section allows you to set the basic properties of the node.

On the **Data Analytics** tab of the DataStudio page, double-click a node. On the node configuration tab that appears, click the **Properties** tab in the right-side navigation pane and set the parameters in the **General** section.

Parameter	Description

Parameter	Description
Node Name	The name of the node that you set when creating the node. To modify the name, right-click the node in the left-side navigation pane and select **Rename**.
Node ID	The unique ID of the node. The node ID is generated when the node is committed at the first time. The node ID cannot be modified.
Node Type	The type of the node that you set when creating the node. The node type cannot be modified.
Owner	The owner of the node. By default, the owner of a newly created node is the current logon user. You can change the owner. **Note** Only a member in the workspace where the node resides can be selected as the owner.
Description	The description of the node, such as the business and usage.
Arguments	The parameter used to assign a value to a variable in the code during node scheduling. You can enter multiple parameters. Separate multiple parameters with spaces.

Parameter value assignment formats for various node types

- Format for ODPS SQL and ODPS MR nodes: `Variable name 1=Parameter 1 Variable name 2=Parameter 2`. Separate multiple parameters with spaces.
- Format for Shell nodes: `Parameter 1 Parameter 2`. Separate multiple parameters with spaces.

For more information about the built-in scheduling parameters, see Parameter configuration.

4.6.2. Scheduling parameters

In common data development scenarios, the code of different types of nodes may be subject to change from time to time. You must dynamically modify the values of some parameters, such as the date and time, based on the requirement changes and time changes.

In this case, you can use the scheduling parameter configuration feature of DataWorks. After relevant parameters are set, auto triggered nodes can automatically parse the code to obtain required data. Configurable parameters in DataWorks are classified into system parameters and custom parameters. We recommend that you use custom parameters.

```
{
    "data":[
    {
        "beginRunningTime":"1564019679966",
        "beginWaitResTime":"1564019679966",
        "beginWaitTimeTime":"1564019679506",
        "bizdate":"1559318400000",
        "createTime":"1564019679464",
        "dagId":332455685,
        "dagType":5,
        "finishTime":"1564019679966",
        "instanceId":2427622331,
        "modifyTime":"1564019679966",
        "nodeName":"vi","status":6
    }
    ],
    "errCode":"0",
    "errMsg":"",
    "requestId":"E17535-8C06-43F6-B1EA-6236FE9",
    "success":true
    }
```


Auto-completion is supported when you specify a data type for a parameter.

Parameter types

Parameter type	Configuration method	Applicable to	Example
System parameters: including bdp.system.bizdate and bdp.system.cyctime	To use the system parameters in the scheduling system, reference ${bdp.system.bizdate} and ${bdp.system.cyctime} in the code, instead of setting them in the Arguments field. The system can automatically replace the values of the parameters that reference the system parameters in the code.	All nodes	N/A
Non-system parameters: custom	Reference ${key1} and ${key2} in the code and set them in the Arguments field, for example, `"key1=value1 key2=value2"`.	Non-Shell nodes	• **Constant parameters**: param1="abc"param2=1234. • **Variables**: param1=$[yyyymmdd], the value of which is calculated based on the value of bdp.system.cyctime.

parameters Parameter type (recommended)	Configuration method	Applicable to	Example
	Reference $1, $2, and $3 in the code and set them in the Arguments field, for example, `"value1 value2 value3"`.	Shell nodes	• **Constant parameters**: "abc" 1234. • **Variables**: $[yyyymmdd], the value of which is calculated based on the value of bdp.system.cyctime.

As described in the preceding table, the values of custom variables are calculated based on the values of system parameters. You can use custom variables to flexibly define the data to be obtained and the data format. For custom parameters, the following types of brackets are used:

- Braces { } define the data timestamp. For example, the value of {yyyymmdd} is calculated based on the value of bdp.system.bizdate.
- Brackets [ ] define the running time. For example, the value of [yyyymmddhh] is calculated based on the value of bdp.system.cyctime.

> **Note** Nodes can be scheduled only in the production environment. Therefore, the values of scheduling variables are replaced only after nodes are run in the production environment.
>
> After you set the scheduling variables for a node, you can click the **Run Smoke Test in Development Environment** icon on the node configuration tab to test whether the values of scheduling variables can be replaced as expected during node scheduling.
>
> You can click the **Properties** tab in the right-side navigation pane, and assign values to scheduling variables in the **Arguments** field in the **General** section. Note the following issues when you set parameters:
>
> - Do not add spaces on either side of the equal sign (=) for a parameter. For example, enter `bizdate=$bizdate`.
> - Separate multiple parameters (if any) with spaces. For example, enter `bizdate=$bizdate datetime=${yyyymmdd}`.

System parameters

DataWorks provides the following system parameters:

- ${bdp.system.cyctime}: the scheduled time to run an instance. Default format: yyyymmddhh24miss. This parameter can specify the hour and minutes of the scheduled time.
- ${bdp.system.bizdate}: the timestamp of data to be analyzed by an instance. Default format: yyyymmdd. The default data timestamp is one day before the scheduled time.

Use the following formula to calculate the running time based on the data timestamp: `Running time = Data timestamp + 1`.

To use the system parameters, you can reference them in the code, instead of setting them in the Arguments field. The system can automatically replace the values of the parameters that reference the system parameters in the code.

> **Note** The scheduling properties of an auto triggered node are configured to define the scheduling rules of the running time. Therefore, you can calculate the data timestamp based on the scheduled time to run an instance and obtain the values of system parameters for the instance.

Example of system parameters

For example, to set an ODPS SQL node to run once per hour from 00:00 to 23:59 every day, perform the following steps if you want to use system parameters in the code:

1. Reference system parameters in the code.

```
insert overwrite table tb1 partition(ds ='20150304') select
c1,c2,c3
from (
select * from tb2
where ds ='${bdp.system.cyctime}') t
full outer join(
select * from tb3
where ds = '${bdp.system.bizdate}') y
on t.c1 = y.c1;
```

2. After the preceding step, your node is partitioned by using the system parameters. Set the scheduling properties and dependencies. For more information, see Schedule and Dependencies. In this example, the node is scheduled by hour.
3. After you set the recurrence and dependencies, commit and deploy the node. Then, you can check the node in Manage auto triggered nodes. The scheduling system generates instances for the auto triggered node from the second day. You can right-click an instance in the directed acyclic graph (DAG) and select **View Runtime Log** to view the parsed values of the system parameters.

   For example, the scheduling system generated 24 running instances for the node on January 14, 2019. The data timestamp is January 13, 2019 for all instances. Therefore, the value of ${bdp.system.bizdate} is 20190113. The running time is the running date appended with the scheduled time. Therefore, the value of ${bdp.system.cyctime} is 20190114000000 plus the scheduled time of each instance.

   Open the runtime logs of each instance and search for the replaced values of the system parameters in the code:

   - The scheduled time for the first instance is January 14, 2019 00:00:00. Therefore, bdp.system.bizdate is replaced with 20190113 and bdp.system.cyctime is replaced with 20190114000000.
   - The scheduled time for the second instance is January 14, 2019 01:00:00. Therefore, bdp.system.bizdate is replaced with 20190113 and bdp.system.cyctime is replaced with 20190114010000.
   - Similarly, the scheduled time for the twenty-fourth instance is January 14, 2019 23:00:00. Therefore, bdp.system.bizdate is replaced with 20190113 and bdp.system.cyctime is replaced with 20190114230000.

Custom parameters for non-Shell nodes

To set scheduling variables for a non-Shell node, add ${Variable name} in the code to reference the function and assign a value to the scheduling variable.

> **Note** The name of a variable in the SQL code can contain only letters, digits, and underscores (_). If the variable name is date, the value of $bizdate is automatically assigned to this variable. For more information, see the "Built-in scheduling parameters" section in this topic. You do not need to assign a value in the Arguments field. Even if another value is assigned, it is not used in the code because the value of $bizdate is automatically assigned in the code.

Example of custom parameters for non-Shell nodes

For example, to set an ODPS SQL node to run once per hour from 00:00 to 23:59 every day, perform the following steps if you want to use the hour-related custom variables thishour and lasthour in the code:

1. Reference the parameters in the code.

```
insert overwrite table tb1 partition(ds ='20150304')  select
 c1,c2,c3
from (
 select * from tb2
 where ds ='${thishour}') t
full outer join(
 select * from tb3
 where ds = '${lasthour}') y
on t.c1 = y.c1;
```

2. Click the **Properties** tab in the right-side navigation pane of the node configuration tab. Assign values to the custom parameters referenced in the code in the **Arguments** field in the **General** section.

   Set the custom parameters in the following formats:

   - thishour=$[yyyy-mm-dd/hh24:mi:ss]
   - lasthour=$[yyyy-mm-dd/hh24:mi:ss-1/24]

   > **Note** The value of yyyy-mm-dd/hh24:mi:ss corresponds to that of cyctime. For more information, see the "Custom parameters" section in this topic.

   You can enter `thishour=$[yyyy-mm-dd/hh24:mi:ss] lasthour=$[yyyy-mm-dd/hh24:mi:ss-1/24]` in the **Arguments** field.

3. Set the node to run once per hour.
4. After you set the recurrence and dependencies, commit and deploy the node. Then, you can check the node in Manage auto triggered nodes. The scheduling system generates instances for the auto triggered node from the second day. You can right-click an instance in the DAG and select **View Runtime Log** to view the parsed values of the custom parameters. The value of cyctime is 20190114010000. Therefore, the value of thishour is 2019-01-14/01:00:00 and the value of lasthour, which indicates the last hour, is 2019-01-14/00:00:00.

Custom parameters for Shell nodes

The parameter configuration procedure of a Shell node is similar to that of a non-Shell node, except that the variable naming rules are different. Variable names for a Shell node cannot be customized, but must follow the $1,$2,$3... format. For example, add $1 in the code of a Shell node and enter the built-in scheduling parameter $xxx in the Arguments field. Then, the value of $xxx can replace that of $1 in the code.

> **Note** If the number of parameters in a Shell node reaches 10, use ${10} to declare the tenth variable.

Example of custom parameters for Shell nodes

For example, set a Shell node to run at 01:00 every day. To use the custom constant parameter myname and the custom variable ct in the code, perform the following steps:

1. Reference the parameters in the code.

   ```
   echo "hello $1, two days ago is $2, the system param is ${bdp.system.cyctime}";
   ```

2. Click the **Properties** tab in the right-side navigation pane of the node configuration tab. Assign values to the custom parameters referenced in the code in the **Arguments** field in the **General** section. Separate multiple parameters with spaces, for example, enter Parameter 1 Parameter 2 Parameter 3. The custom parameters are parsed based on the parameter sequence. For example, $1 is replaced with the value of Parameter 1. In this example, enter abcd $[yyyy-mm-dd-2] in the Arguments field to set $1 and $2 to abcd and $[yyyy-mm-dd-2], respectively.
3. Set the node to run at 01:00 every day.
4. After you set the recurrence and dependencies, commit and deploy the node. Then, you can check the node in Operation Center. The scheduling system generates instances for the auto triggered node from the second day. Right-click an instance in the DAG and select **View Runtime Log**. The logs show that $1 in the code is replaced with abcd, $2 is replaced with 2019-01-12 (two days before the running date), and ${bdp.system.cyctime} is replaced with 20190114010000.

Custom parameters

Custom parameters are divided into constant parameters and variables based on the value type. DataWorks provides some built-in scheduling parameters as variables.

- Constant parameters

  For example, for an SQL node, add ${Variable name} in the code and set the following parameter for the node: Variable name=Fixed value.

  - Code: `select xxxxxx type='${type}'`
  - Value assigned to the scheduling variable: type='aaa'. When the node is run, the variable in the code is replaced with type='aaa'.

- Variables

  Variables are built-in scheduling parameters whose values depend on the system parameters ${bdp.system.bizdate} and ${bdp.system.cyctime}.

  For example, for an SQL node, add ${Variable name} in the code and set the following parameter for the node: Variable name=Scheduling parameter.

  - Code: `select xxxxxx dt=${datetime}`
  - Value assigned to the scheduling variable: datetime=$bizdate

    If the node is run on July 22, 2017, the variable in the code is replaced with dt=20170721.

Built-in scheduling parameters

- $bizdate

- Parameter description: the data timestamp in the format of yyyymmdd. By default, the value of this parameter is one day before the scheduled time to run a node.
- For example, the code of an ODPS SQL node includes pt=${datetime}, and the parameter configured for the node is datetime=$bizdate. If the node is run on July 22, 2017, $bizdate is replaced with pt=20170721.

- $cyctime
  - Parameter description: the scheduled time to run a node. If no scheduled time is configured for a node scheduled by day, $cyctime is set to 00:00 of the day. The time is accurate to seconds. This parameter is usually used for nodes scheduled by hour or minute.

    > **Note**
    > - Pay attention to the difference between the time parameters configured by using $[] and ${}. $bizdate specifies the data timestamp, which is one day before the current day by default.
    > - $cyctime specifies the scheduled time to run a node. If no scheduled time is configured for a node scheduled by day, $cyctime is set to 00:00 of the day. The time is accurate to seconds. This parameter is usually used for nodes scheduled by hour or minute.
    >
    >   For example, if a node is scheduled to run at 00:30 on the current day, $cyctime is set to yyyy-mm-dd 00:30:00.
    > - If a time parameter is configured by using ${}, $bizdate is used as the benchmark for running nodes. The time parameter is replaced with the data timestamp selected for retroactive data generation.
    > - If a time parameter is configured by using $[], $cyctime is used as the benchmark for running nodes. The time is calculated in the same way as the time in Oracle. The time parameter is replaced with the data timestamp selected for retroactive data generation plus one day.
    >
    >   For example, if the data timestamp is set to 20140510 for retroactive data generation, $cyctime is replaced with 20140511.

  - The following examples show the values of custom parameters when $cyctime is set to 20140515103000:
    - $[yyyy] = 2014, $[yy] = 14, $[mm] = 05, $[dd] = 15, $[yyyy-mm-dd] = 2014-05-15, $[hh24:mi:ss] = 10:30:00, $[yyyy-mm-dd hh24:mi:ss] = 2014-05-1510:30:00
    - $[hh24:mi:ss - 1/24] = 09:30:00
    - $[yyyy-mm-dd hh24:mi:ss -1/24/60] = 2014-05-1510:29:00
    - $[yyyy-mm-dd hh24:mi:ss -1/24] = 2014-05-15 09:30:00
    - $[add_months(yyyymmdd,-1)] = 20140415
    - $[add_months(yyyymmdd,-12*1)] = 20130515
    - $[hh24] =10
    - $[mi] =30
  - Method for testing the $cyctime parameter:

    After an instance starts to run, right-click the instance in the DAG and select **More**. Check whether the scheduled time is the time at which the instance is run.

- $jobid
  - Parameter description: the ID of the workflow to which a node belongs.
  - Example: jobid=$jobid.
- $nodeid
  - Parameter description: the ID of a node.
  - Example: nodeid=$nodeid.
- $taskid
  - Parameter description: the instance ID of a node.
  - Example: taskid=$taskid.
- $bizmonth
  - Parameter description: the month of the data timestamp in the format of yyyymm. If the month of a data timestamp is the current month, the value of $bizmonth is the month of the data timestamp minus 1. Otherwise, the value of $bizmonth is the month of the data timestamp.
  - For example, the code of an ODPS SQL node includes pt=${datetime}, and the parameter configured for the node is datetime=$bizmonth.

    Assume that the current day is July 22, 2017. If the node is run on July 22, 2017, $bizmonth is replaced with pt=201706.
- ${...} custom parameters
  - You can customize a time format based on the value of $bizdate, where yyyy indicates the four-digit year, yy indicates the two-digit year, mm indicates the month, and dd indicates the day. You can use any combination of these parameters, for example, ${yyyy}, ${yyyymm}, ${yyyymmdd}, and ${yyyy-mm-dd}.
  - $bizdate is accurate to the day. Therefore, ${...} can specify only the year, month, or day.
  - The following table describes how to specify other intervals based on $bizdate.

Interval	Expression
N years later	${yyyy+N}
N years before	${yyyy-N}
N months later	${yyyymm+N}
N months before	${yyyymm-N}
N weeks later	${yyyymmdd+7*N}
N weeks before	${yyyymmdd-7*N}
N days later	${yyyymmdd+N}
N days before	${yyyymmdd-N}

- $gmtdate

- Parameter description: the current date in the format of yyyymmdd. By default, the value of this parameter is the current date. During retroactive data generation, the input value is the data timestamp plus one day.
- For example, the code of an ODPS SQL node includes pt=${datetime}, and the parameter configured for the node is datetime=$gmtdate. Assume that the current day is July 22, 2017. If the node is run on July 22, 2017, $gmtdate is replaced with pt=20170722.

- ${yyyymmdd}
  - Parameter description: the data timestamp in the format of yyyymmdd. The value of this parameter is the same as that of $bizdate. This parameter supports delimiters, for example, yyyy-mm-dd.

    By default, the value of this parameter is one day before the scheduled time to run a node. You can customize a time format for this parameter, for example, yyyy-mm-dd for ${yyyy-mm-dd}.
  - Examples:
    - The code of an ODPS SQL node includes pt=${datetime}, and the parameter configured for the node is datetime=${yyyy-mm-dd}. If the node is run on July 22, 2018, ${yyyy-mm-dd} is replaced with pt=2018-07-21.
    - The code of an ODPS SQL node includes pt=${datetime}, and the parameter configured for the node is datetime=${yyyymmdd-2}. If the node is run on July 22, 2018, ${yyyymmdd-2} is replaced with pt=20180719.
    - The code of an ODPS SQL node includes pt=${datetime}, and the parameter configured for the node is datetime=${yyyymm-2}. If the node is run on July 22, 2018, ${yyyymm-2} is replaced with pt=201805.
    - The code of an ODPS SQL node includes pt=${datetime}, and the parameter configured for the node is datetime=${yyyy-2}. If the node is run on July 22, 2018, ${yyyy-2} is replaced with pt=2016.
    - You can assign values to multiple parameters when configuring an ODPS SQL node. For example, set startdatetime=$bizdate enddatetime=${yyyymmdd+1} starttime=${yyyy-mm-dd} endtime=${yyyy-mm-dd+1}.

FAQ

- Q: The table partition format is pt=yyyy-mm-dd hh24:mi:ss, but spaces are not allowed in scheduling parameters. How can I configure the format of $[yyyy-mm-dd hh24:mi:ss]?

  A: Use the custom variables datetime=$[yyyy-mm-dd] and hour=$[hh24:mi:ss] to obtain the date and time. Then, join them together to form pt="${datetime} ${hour}" in the code. Separate the two variables with a space.

- Q: The table partition is pt="${datetime} ${hour}" in the code. To obtain the data for the last hour when the node is run, the custom variables datetime=$[yyyymmdd] and hour=$[hh24-1/24] can be used to obtain the date and time, respectively. However, for an instance running at 00:00, it analyzes data for 23:00 of the current day, instead of 23:00 of the previous day. What measures can I take in this case?

  A: Modify the formula of datetime to $[yyyymmdd-1/24] and keep the formula of hour unchanged at $[hh24-1/24]. Then, the node can be run properly.
  - For an instance that is scheduled to run at 2015-10-27 00:00:00, the values of $[yyyymmdd-1/24] and $[hh24-1/24] are 20151026 and 23, respectively. This is because the scheduled time minus 1 hour is a time value that belongs to yesterday.

- For an instance that is scheduled to run at 2015-10-27 01:00:00, the values of $[yyyymmdd-1/24] and $[hh24-1/24] are 20151027 and 00, respectively. This is because the scheduled time minus 1 hour is a time value that belongs to the current day.

DataWorks offers the following node running modes:

- Manually run a node in DataStudio: You must assign temporary values to parameters in the code to ensure proper running of the node. The assigned values are not saved as node properties and do not take effect in other node running modes.
- Automatically run a node at specified intervals: No configuration is needed in the Arguments field. The scheduling system automatically replaces the values of parameters based on the scheduled time of the current instance.
- Test a node or generate retroactive data: You must specify the data timestamp. The scheduled time of each instance can be calculated based on the formula described earlier in this topic.

4.6.3. Scheduling properties

This topic describes how to configure the scheduling properties of a node, including the recurrence and dependencies.

You can click the **Properties** tab in the right-side navigation pane of the node configuration tab and set the parameters in the **Schedule** section.

Node status

- **Normal**: If you select this option, the node is run based on the recurrence. By default, this option is selected for a node.
- **Dry Run**: If you select this option, the node is run based on the recurrence. However, the scheduling system does not actually run the code but directly returns a success response.
- **Retry Upon Error**: If you select this check box, the node is rerun when it encounters an error. By default, a node can be automatically rerun for a maximum of three times at an interval of 2 minutes.
- **Skip Execution**: If you select this check box, the node is run based on the recurrence. However, the scheduling system does not actually run the code but directly returns a failure response. You can select this check box if you want to suspend a node and run it later.

Recurrence

After a node is committed and deployed, the scheduling system generates instances every day from the next day based on the scheduling properties of the node. Then, the scheduling system runs the instances based on the running results of ancestor instances and the scheduled time. If a node is committed and deployed after 23:30, the scheduling system generates instances for it from the third day.

> **Note**
>
> If you schedule a node to run every Monday, the node is run only on Mondays. On the other days, the scheduling system does not actually run the code but directly returns a success response. When you test a node scheduled by week or generate retroactive data for the node, you must set the data timestamp to one day earlier than the scheduled time to run the node.
>
> For an auto triggered node, its dependencies take priority over other scheduling properties. That is, when the scheduled time arrives, the scheduling system does not immediately run a node instance but first checks whether all the ancestor instances are run.
>
> - The node instance is in the Not Running state if any ancestor instances are not run when the scheduled time arrives.
> - The node instance is in the Pending (Schedule) state if the scheduled time does not arrive but all the ancestor instances are run.
> - The node instance is in the Pending (Resources) state if all the ancestor instances are run and the scheduled time arrives.

Cross-cycle dependencies

DataWorks supports the following three types of cross-cycle dependencies:

- Dependency on instances of child nodes
  - Node dependency: The current node depends on the last-cycle instances of its child nodes. For example, Node A has three child nodes B, C, and D. If you select this node dependency, Node A depends on the last-cycle instances of nodes B, C, and D.
  - Business scenario: The current node depends on instances of child nodes in the last cycle to cleanse the output tables of the current node and check whether the final result is generated properly.
- Dependency on instances of the current node
  - Node dependency: The current node depends on its last-cycle instances.
  - Business scenario: The current node depends on the data output result of its last-cycle instances.
- Dependency on instances of custom nodes: If you select this node dependency, enter the IDs of the nodes on which the current node depends. You can specify multiple nodes and separate their IDs with commas (,). For example, enter 12345,23456.
  - Node dependency: The current node depends on the last-cycle instances of custom nodes.
  - Business scenario: In the business logic, the current node depends on the proper output of other business data that is not processed by the current node.

> **Note** The difference between cross-cycle dependencies and dependencies in the current cycle lies in that cross-cycle dependencies are displayed as dotted lines in Operation Center.
>
> Before deleting a node from Operation Center, you must delete all dependencies of the node so that other nodes can run properly.

Scheduled by day

Nodes scheduled by day are automatically run once per day. When you create an auto triggered node, the node is set to run at 00:00 every day by default. You can specify another time as needed. In the example shown in the following figure, the time is specified as 13:00.

- If you select Customize Runtime, the node is run at the specified time every day. The time format is YYYY-MM-DD HH:MM:SS.

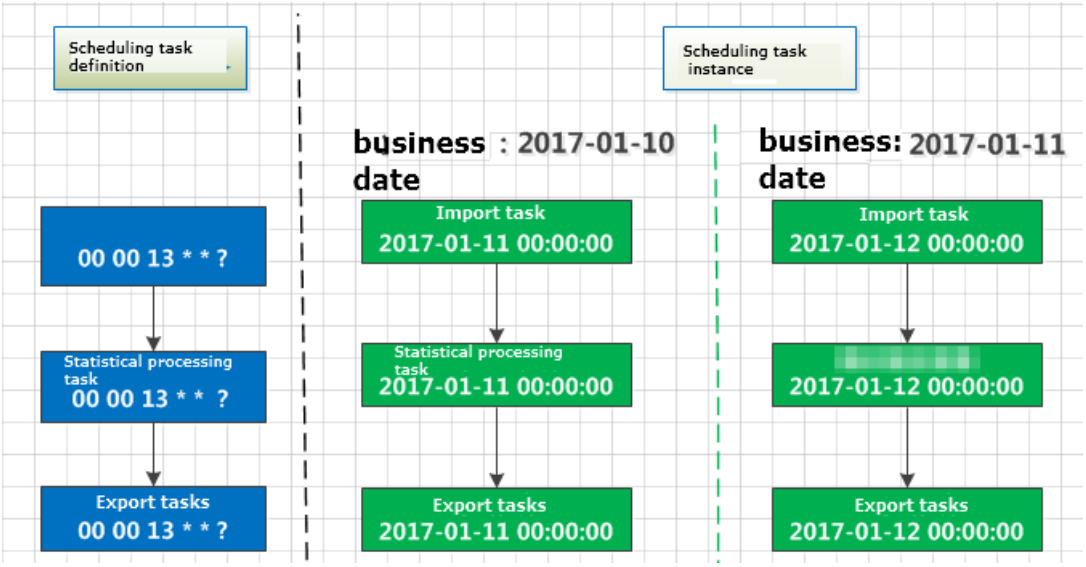
  > **Note** An auto triggered node can be run only when all the ancestor instances are run and the scheduled time arrives. Both prerequisites are indispensable and have no specific chronological order.

- If you clear Customize Runtime, the scheduled time of the node is randomly set in the range of 00:00 to 00:30.

Scenarios:

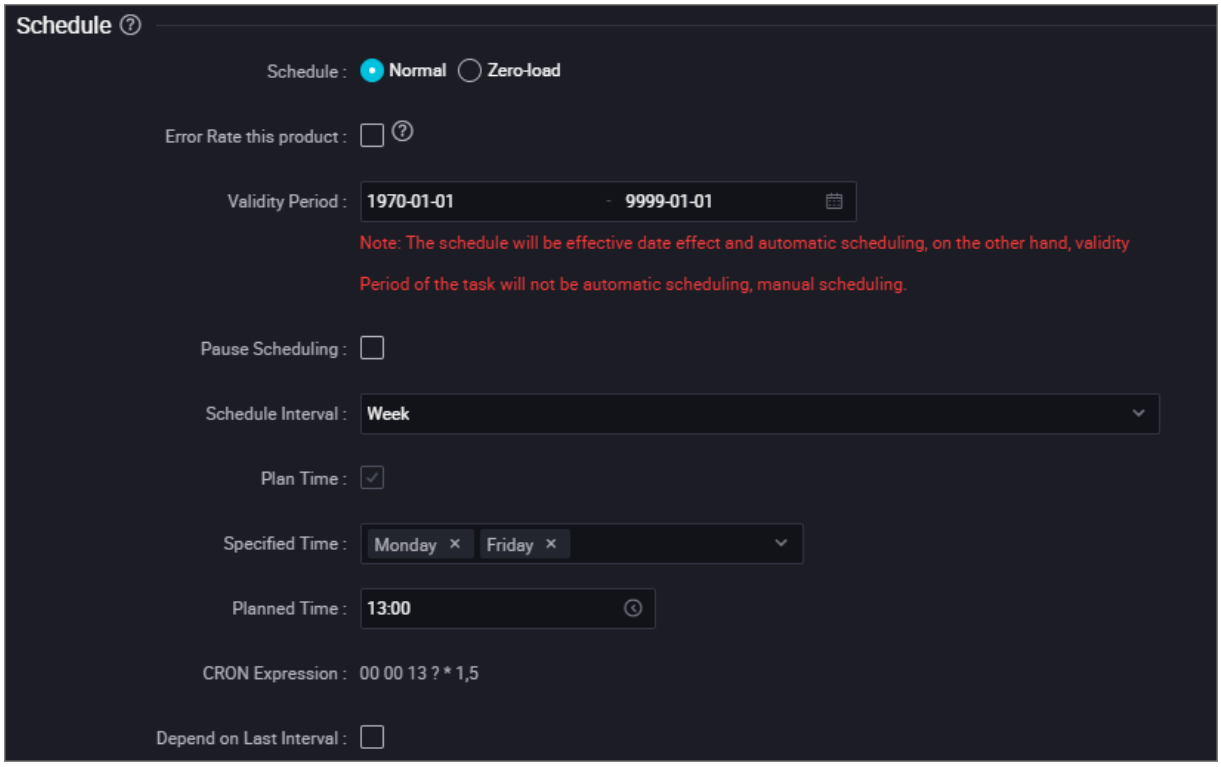
For example, you have created an import node, an analytics node, and an export node. They are all scheduled to run at 13:00 every day. The analytics node depends on the import node, and the export node depends on the analytics node. The following figure shows that the analytics node is configured to depend on the import node.

Based on the preceding node scheduling properties, the scheduling system automatically generates and runs instances for the nodes, as shown in the following figure.



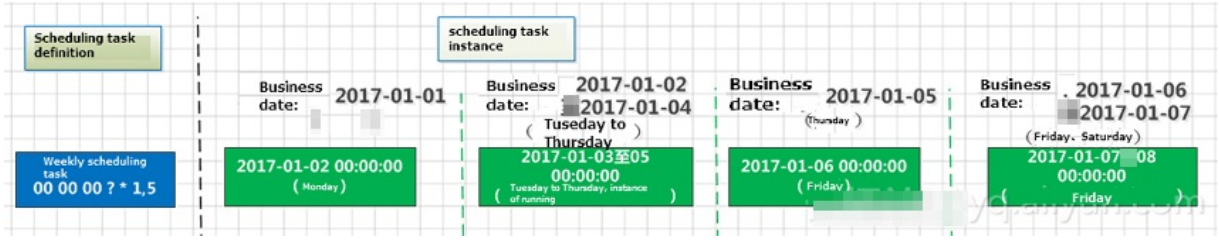
Scheduled by week

Nodes scheduled by week are automatically run at a specified time of specified days every week. On the other days, the scheduling system still generates instances to make sure the proper running of descendant instances. However, the system does not actually run the code or consume resources but directly returns a success response.



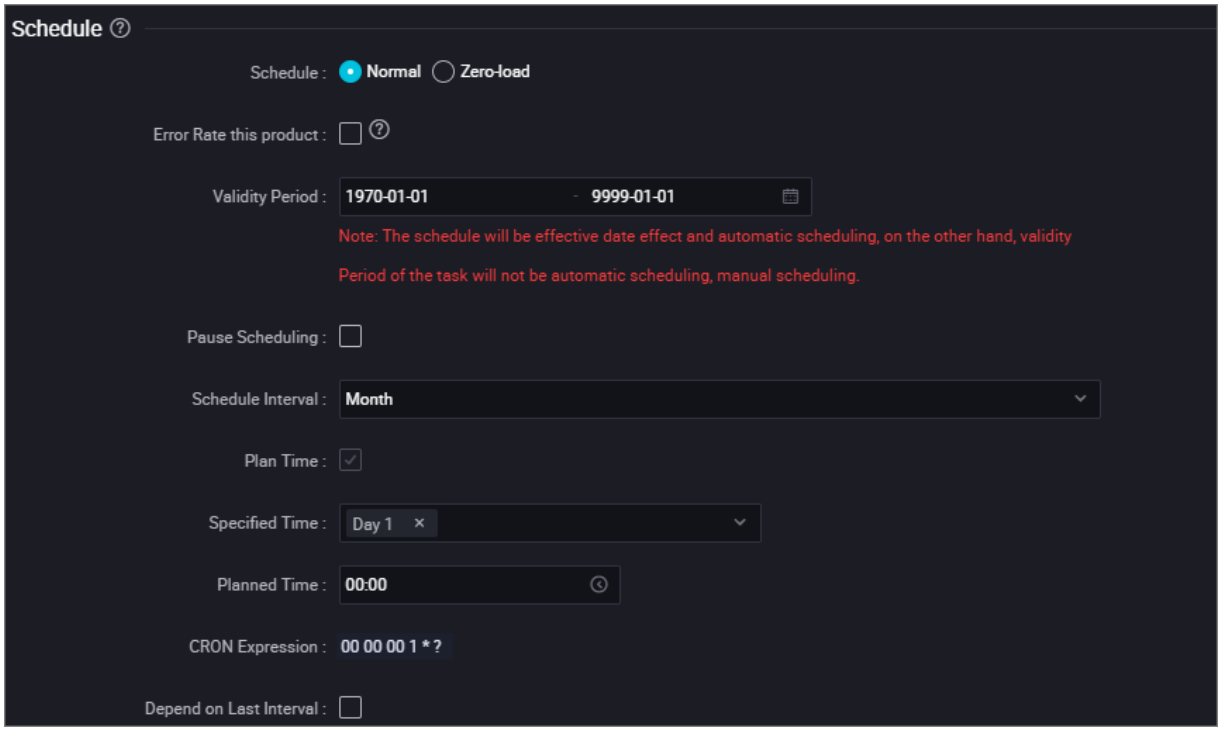
For example, you have created a node. As shown in the preceding figure, the scheduling system runs instances generated on Mondays and Fridays, but returns success responses without running the code for instances generated on Tuesdays, Wednesdays, Thursdays, Saturdays, and Sundays.

Based on the preceding node scheduling properties, the scheduling system automatically generates and runs instances for the node, as shown in the following figure.



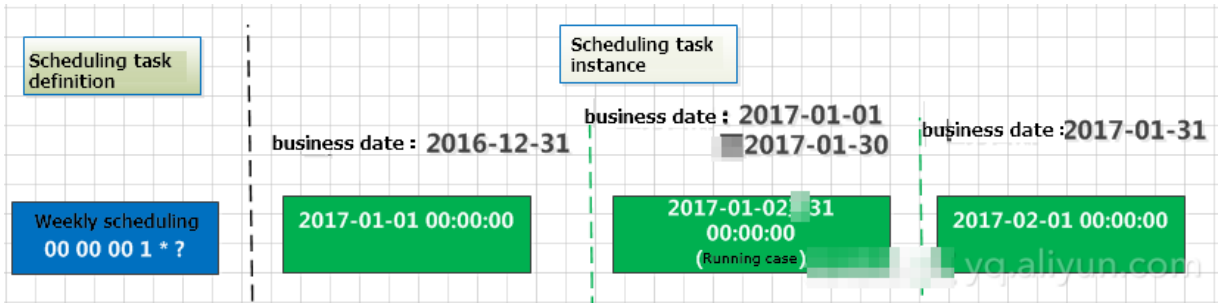
Scheduled by month

Nodes scheduled by month are automatically run at a specified time of specified days every month. On the other days, the scheduling system still generates instances to make sure the proper running of descendant instances. However, the system does not actually run the code or consume resources but directly returns a success response.



For example, you have created a node. As shown in the preceding figure, the scheduling system runs the instance generated on the first day of each month, but returns success responses without running the code for instances generated on the other days.

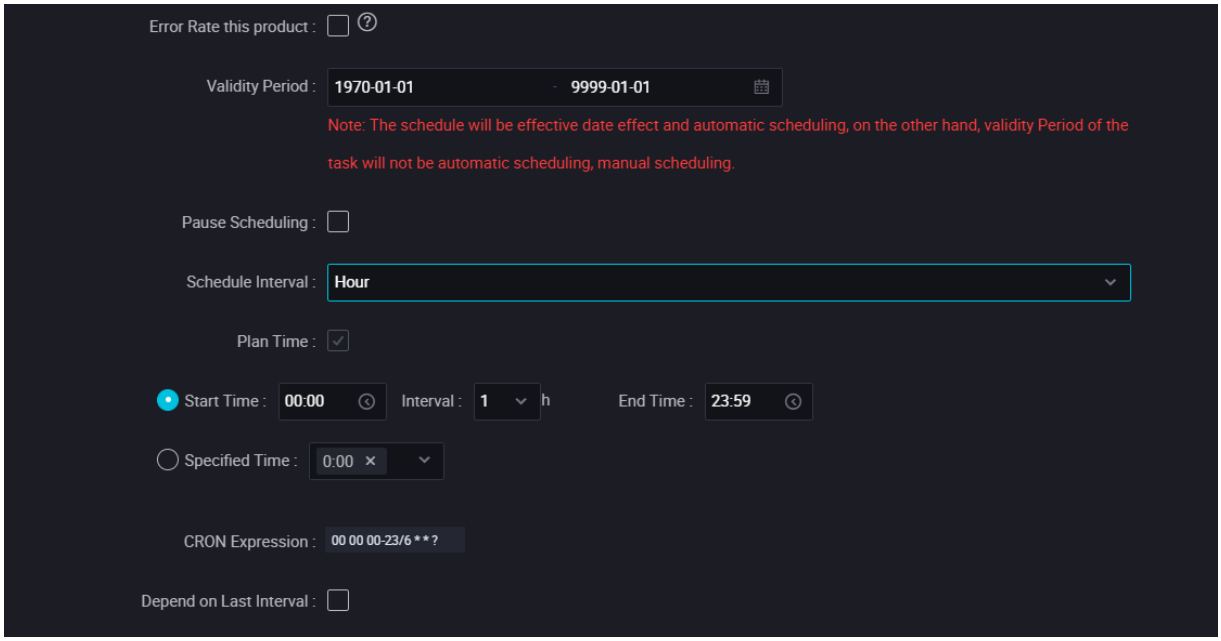
Based on the preceding node scheduling properties, the scheduling system automatically generates and runs instances for the node, as shown in the following figure.



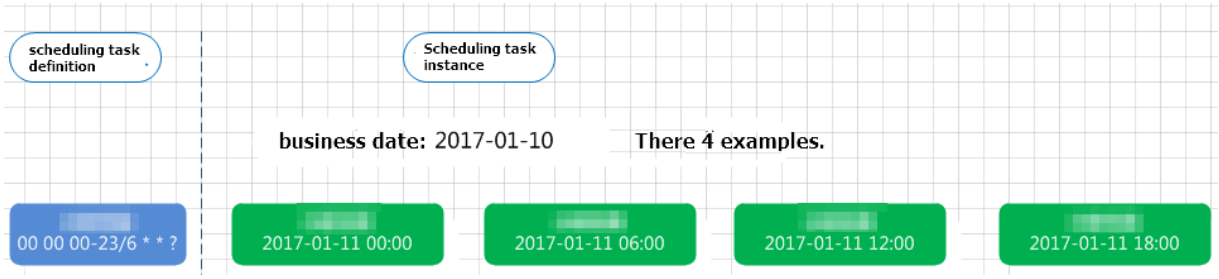
Scheduled by hour

Nodes scheduled by hour are automatically run once every N hours in a specific time period every day. For example, a node is run once per hour from 01:00 to 04:00 every day.

> **Note** The time period is a closed interval. For example, if a node is scheduled to run once per hour in the period from 00:00 to 03:00, the scheduling system generates four instances every day, which are run at 00:00, 01:00, 02:00, and 03:00, respectively.



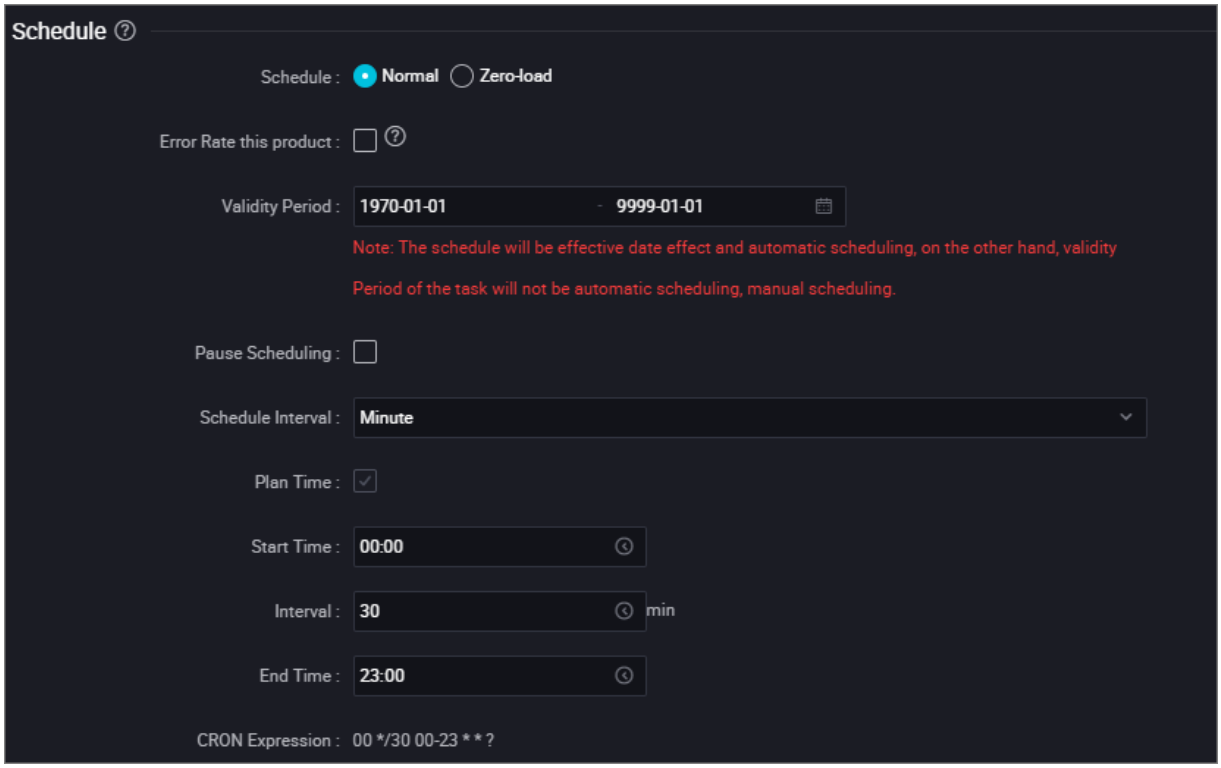
For example, you have created a node. As shown in the preceding figure, the node is automatically run every 6 hours in the period from 00:00 to 23:59 every day. In this case, the scheduling system automatically generates and runs instances for the node, as shown in the following figure.



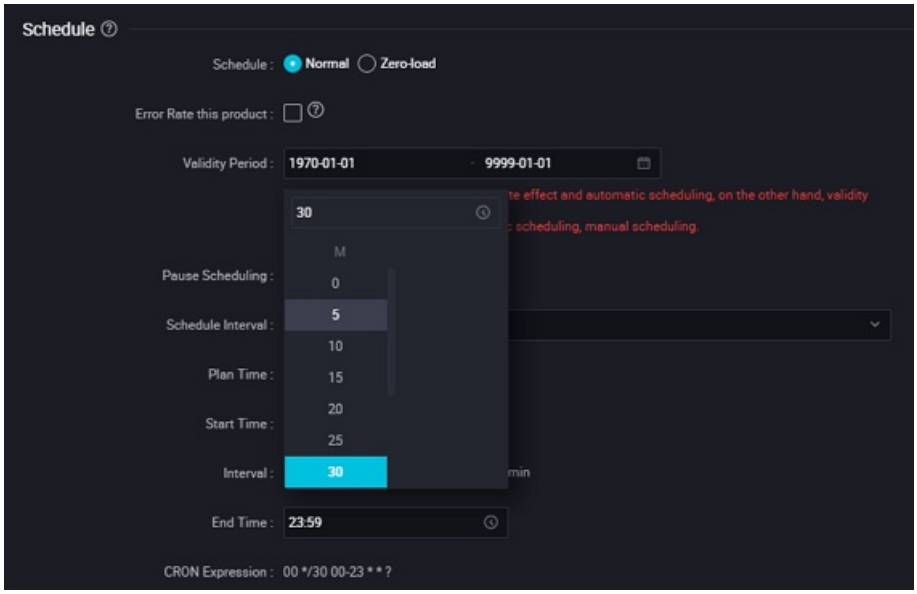
Scheduled by minute

Nodes scheduled by minute are automatically run once every N minutes in a specific time period every day.

For example, you have created a node. As shown in the following figure, the node is run every 30 minutes in the period from 00:00 to 23:00 every day.



Currently, a minimum interval of 5 minutes is supported. The time expression is automatically generated based on the time you select and cannot be modified.



FAQ

- Q: Node A is scheduled by hour and Node B is scheduled by day. How do I enable Node B to automatically run every day after all instances of Node A are run?

A: A node scheduled by day can depend on a node scheduled by hour. To enable Node B to automatically run every day after all 24 instances of Node A are run, do not specify the time to run Node B every day. Then, configure Node A as an ancestor of Node B. For more information, see the Dependencies topic. A node can depend on any other node, regardless of the recurrence. The recurrence of each node is specified in its scheduling properties.

- Q: Node A is run once per hour on the hour every day and Node B is run once per day. How do I enable Node B to automatically run after Node A is run for the first time every day?

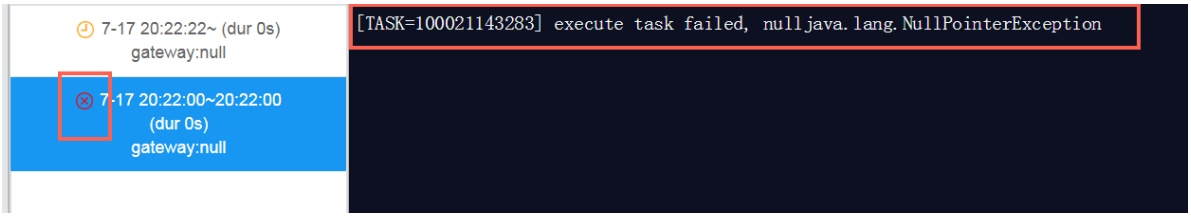
  A: When configuring the scheduling properties of Node A, select **Cross-Cycle Dependencies** and select **Instances of Current Node** from the Depend On drop-down list. When configuring the scheduling properties of Node B, configure Node B to depend on Node A and set the scheduled time of Node B to 00:00 every day. In this way, instances of Node B only depend on the instance of Node A generated at 00:00 every day, that is, the first instance of Node A.

- Q: Node A is run once every Monday and Node B depends on Node A. How do I enable Node B to run once every Monday?

  A: Set the scheduling properties of Node B to be the same as those of Node A. That is, select **Week** as the instance recurrence and select **Monday**.

- Q: How are the instances of a node affected when the node is deleted?

  A: When a node is deleted, its instances are retained because the scheduling system still generates one or more instances for the node based on the scheduling properties. However, when the scheduling system runs such instances after the node is deleted, an error message appears because the required code is unavailable, as shown in the following figure.

- Q: Can I enable a node to process monthly data on the last day of each month?

  A: No, DataWorks does not support setting a node to run on the last day of each month. If you enable a node to run on the thirty-first day of each month, the scheduling system runs a node instance in each month that has 31 days and returns a success response without running the code in any other month.

  We recommend that you configure a node to process the data of the past month on the first day of each month.

- Q: If a node scheduled by day depends on a node scheduled by hour, how do I enable the node scheduled by day to run at 00:00 every day?

  A: You can configure the node scheduled by day to depend on the data generated on the day before for the node scheduled by hour. If the node scheduled by day depends on the data generated on the current day for the node scheduled by hour, the instances of the node scheduled by day can be run only on the next day.

  In the Schedule section of the node scheduled by day, select **Cross-Cycle Dependencies**, select **Instances of Custom Nodes** from the Depend On drop-down list, and then enter the ID of the node scheduled by hour on which the node scheduled by day depends. Commit and deploy the node scheduled by day.

- Q: What can I do if I do not know when the output data of the ancestor node is generated?

A: You can set the cross-cycle dependency for the current node to depend on the last-cycle instances of the ancestor node.

- Q: After a modified node is committed and deployed to the production environment, will the node instances that were originally faulty in the production environment be overwritten?

  A: No, the node instances that have been generated will not be overwritten. The updated code is used to run the node instances that are newly generated and have not been run. If the scheduling properties are modified, the modified configuration also applies to the newly generated node instances.

4.6.4. Dependencies

Scheduling dependencies are the foundation for building orderly workflows. You must configure correct dependencies between nodes to make sure that business data is produced effectively and in a timely manner. This helps standardize data development activities.

DataWorks allows you to automatically parse node dependencies from the code or manually customize node dependencies. You can configure correct relationships between ancestor and descendant nodes and monitor the running status of nodes to make sure the orderly production of business data.

The purpose of configuring node dependencies is to check the data output time of the table queried by SQL statements and check whether data is properly produced from an ancestor node based on the node status.

You can set the output of an ancestor node as the input of a descendant node to configure a dependency between the two nodes.

Regardless of the dependency configuration mode, the overall scheduling logic is that descendant nodes can be run only after ancestor nodes are run. Therefore, each node in a workflow must have at least one parent node. The dependencies between the parent nodes and child nodes are the core of scheduling dependencies. The following sections describe the principles and configuration methods of scheduling dependencies in detail.

Differences between automatic parsing and custom dependencies

DataWorks can automatically parse the input and output of a node based on the lineage parsed from the code.

If the lineage parsed from the code is inaccurate, you can add custom dependencies as needed. We recommend that you write the code correctly to parse the lineage from the code and reduce custom dependencies. The following example shows how to configure the input and output of a node.

Auto Parse: If you select Yes, node dependencies are automatically parsed from the code.

For example, the code of an ODPS SQL node is as follows:

```
insert overwrite table table_a as select * from project_b_name.table_b;
```

From the code, DataWorks determines that the current node depends on the node that generates table_b and the current node generates table_a. Therefore, the output name of the parent node is project_b_name.table_b and the output name of the current node is project_name.table_a.

- If you do not want to parse node dependencies from the code, select **No** for Auto Parse.
- If a table in an SQL statement is both an output table and a referenced table on which another node depends, the table is parsed only as an output table.

- If a table in an SQL statement is used as an output table or a referenced table multiple times, only one scheduling dependency is parsed.
- If the SQL code contains a temporary table, the table is not involved in a scheduling dependency. Temporary tables are prefixed with t_. For more information, see Project Configuration.

Parent nodes

In the Dependencies section of a node, you must specify an ancestor node as the parent node on which the current node depends. You must enter the output name of the ancestor node, rather than the ancestor node name. A node may have multiple output names. Enter an output name as needed. You can search for an output name of the ancestor node to be added, or click Parse I/O to parse the output name based on the lineage parsed from the code.

> **Note** You must enter an output name or output table name to search for the ancestor node.
>
> If you enter an output name to search for the ancestor node, DataWorks searches for the output name among the output names of nodes that have been committed to the scheduling system.

- Search by entering an output name

  You can enter an output name to search for the ancestor node and configure the node as the parent node of the current node to create a dependency.

- Search by entering an output table name

  When using this method, make sure that the entered output table name of the ancestor node is the table name used in the INSERT or CREATE statement of the current node, such as Project name.Table name . Such output names can be automatically parsed.

  After you click the **Submit** icon, the output table name of the parent node configured for the current node can be found when you enter an output table name to search for the ancestor node for other nodes.

Outputs

You can click the **Properties** tab in the right-side navigation pane to view and configure the output of the current node.

DataWorks assigns a default output name that ends with .out to each node. You can also customize an output name or click Parse I/O to parse the output name based on the lineage parsed from the code.

> **Note** The output name of each node must be globally unique.

FAQ

- Q: After DataWorks automatically parses the input and output of a node, the node fails to be committed. An error message appears to indicate that the parsed output name workshop_yanshi.tb_2 of the parent node does not exist and you must commit the parent node before committing the current node. Why does this error occur?

  A: The possible causes are as follows:

  - The ancestor node is not committed. Commit the ancestor node and try again.
  - The ancestor node is committed, but workshop_yanshi.tb_2 is not an output name of the ancestor node.

> **Note** Usually, the output names of the parent node and the current node are automatically parsed based on the table name that is used in the INSERT or CREATE statement or follows the FROM keyword. Make sure that you follow the principles of automatic parsing in the Differences between automatic parsing and custom dependencies section.

- Q: In the output of the current node, the descendant node name and ID are empty and cannot be specified. Why does this happen?

  A: If the current node does not have a descendant node, the descendant node name and ID are empty. After a descendant node is configured for the current node, the corresponding content can be automatically parsed.

- Q: What is the output name of a node used for?

  A: The output name of a node is used to establish dependencies between nodes. For example, if the output name of Node A is ABC and Node B uses ABC as its input, a dependency is established between nodes A and B.

- Q: Can a node have multiple output names?

  A: Yes, a node can have multiple output names. If a descendant node references an output name of the current node as the output name of the parent node, a dependency is established between the descendant node and the current node.

- Q: Can multiple nodes have the same output name?

  A: No, the output name of each node must be unique under your Apsara Stack tenant account. If multiple nodes export data to the same MaxCompute table, we recommend that you use Table name_Partition ID as the output name format of these nodes.

- Q: How can I avoid intermediate tables when I enable DataWorks to automatically parse node dependencies?

  A: Right-click an intermediate table name in the SQL code and select **Delete Input** or **Delete Output**. Then, click Parse I/O to parse the input and output of the node.

- Q: How do I configure dependencies for the upmost node in a workflow?

  A: You can set the node to depend on the root node of the current workspace.

- Q: Why do I find a non-existent output name of Node B when I enter an output name to search for the ancestor node for Node A?

  A: DataWorks searches for the output name among the output names of nodes that have been committed to the scheduling system. After Node B is committed, if you delete the output name of Node B and does not commit Node B to the scheduling system again, the deleted output name of Node B can still be found.

- Q: How do I enable nodes A, B, and C to run in sequence once per hour?

  A: Set the output of Node A as the input of Node B and the output of Node B as the input of Node C. Also, set nodes A, B, and C to run once per hour.

- Q: An error message is returned to indicate that the parent node ID fails to be automatically parsed based on an output table name. Why does this error occur?

  A: This error does not indicate that the table does not exist. Instead, it indicates that the table is not the output of a specific node. Therefore, the table name cannot be used to find the node that generates the table data. In this case, the dependency on the node cannot be created.

According to the principles of automatic parsing described in this topic, a dependency is created after the output of an ancestor node is set as the input of a descendant node. If no ancestor node can be parsed based on the xc_demo_partition table referenced in SQL statements, no node uses the xc_demo_partition table as its output.

You can resolve this problem in the following way:

i. Find the node that generates the table data and view the node output.

   If you do not know which the target node is, you can enter keywords to search the code for the node in fuzzy match mode.

ii. If the table data is uploaded from a local server or you do not need to depend on the node, you can right-click the table name in the code and select **Delete Input**.

> **Note** We recommend that you write the code correctly to parse the lineage from the code and reduce custom dependencies.

4.7. Components

4.7.1. Create a script template

This topic describes the definition and composition of script templates and how to create a script template.

Definition

A script template defines an SQL code process that involves multiple input and output parameters. Each SQL code process references one or more source tables. You can filter source table data, join source tables, and aggregate them to generate a result table required for new business.

Value

In actual business, many SQL code processes are similar. The input and output tables in these processes may have the same or compatible schema but different names. In this case, developers can abstract an SQL code process as a script template to reuse the SQL code. The script template extracts input parameters from input tables and generates output parameters in output tables.

To create SQL script templates, you can select script templates from the script template list based on your business process and configure specific input and output tables in your business for the selected script templates, without repeatedly copying the code. This greatly improves the development efficiency and avoids repeated development. You can deploy and run the created SQL script templates in the same way as other SQL nodes.

Composition

Similar to a function, a script template consists of input parameters, output parameters, and an SQL code process.

Input parameters

The input parameters of a script template have the properties such as the parameter name, parameter type, parameter description, and parameter definition. The parameter type can be table or string.

- A table-type parameter specifies the table to be referenced in an SQL code process. When you use a

script template, you can specify the input table required for the specific business.

- A string-type parameter specifies the variable control parameter in an SQL code process. For example, to export only the sales amount of the top N cities in each region in a result table of an SQL code process, you can use a string-type parameter to specify the value of N.

  To export the total sales amount of a province in a result table of an SQL code process, you can set a string-type parameter to specify the province and obtain the sales data of the specified province.
- The parameter description specifies the role of a parameter in an SQL code process.
- The parameter definition is a text definition of the table schema, which is required only for table-type parameters. When you specify the parameter definition for a table-type parameter, you must provide an input table that contains the same field names and compatible types defined by the table-type parameter so that the SQL code process can run properly. Otherwise, an error is returned when the SQL code process runs because the specified field name cannot be found in the input table. The input table must contain the field names and types defined by the table-type parameter. The input table can also contain other fields. The field names and types in the input table can be in any order. The parameter definition is for reference only.
- We recommend that you enter the parameter definition in the following format:

  ```
  Name of field 1 Type of field 1 Description of field 1
  Name of field 2 Type of field 2 Description of field 2
  Name of field n Type of field n Description of field n
  ```

  Examples:

  ```
  area_id string 'Region ID'
  city_id string 'City ID'
  order_amt double 'Order amount'
  ```

Output parameters

- The output parameters of a script template have the properties such as the parameter name, parameter type, parameter description, and parameter definition. The parameter type must be table. A string-type output parameter has no logical meaning.
- A table-type parameter specifies the table to be generated in an SQL code process. When you use a script template, you can specify the result table that the SQL code process generates for the specific business.
- The parameter description specifies the role of a parameter in an SQL code process.
- The parameter definition is a text definition of the table schema. When you specify the parameter definition for a table-type parameter, you must provide an output table that contains the same number of fields and compatible types defined by the table-type parameter so that the SQL code process can run properly. Otherwise, an error is returned when the SQL code process runs because the number of fields does not match or the field type is incompatible. The field names of the output table do not need to be consistent with those defined by the table-type parameter. The parameter definition is for reference only.
- We recommend that you enter the parameter definition in the following format:

  ```
  Name of field 1 Type of field 1 Description of field 1
  Name of field 2 Type of field 2 Description of field 2
  Name of field n Type of field n Description of field n
  ```

  Examples:

```
area_id string 'Region ID'
city_id string 'City ID'
order_amt double 'Order amount'
rank bigint 'Ranking'
```

SQL code process

The parameters in an SQL code process are referenced in the following format: `@@{Parameter name}`.

By containing an abstract SQL code process, a script template controls and processes an input table based on input parameters to generate an output table with business value.

To develop an SQL code process, you must use input and output parameters in the code properly to make sure that they can be set as needed and correct SQL code can be generated and run during the process.

Create a script template

1. Log on to the DataWorks console.
2. On the left-side navigation submenu, click the **Snippets** icon.
3. On the Snippets tab, move the pointer over **+Create** and choose **Create > Snippet**.
4. In the **Create Snippet** dialog box, set **Snippet Name**, **Description**, and **Location**.
5. Click **Commit**.

Source table schema

The following table describes the schema of a source MySQL table that contains sales data.

Field	Data type	Description
order_id	varchar	The ID of the order.
report_date	datetime	The date of the order.
customer_name	varchar	The name of the customer.
order_level	varchar	The level of the order.
order_number	double	The number of orders.
order_amt	double	The amount of the order.
back_point	double	The discount.
shipping_type	varchar	The transportation method.
profit_amt	double	The amount of the profit.
price	double	The unit price.
shipping_cost	double	The transportation cost.
area	varchar	The region.

Field	Data type	Description
province	varchar	The province.
city	varchar	The city.
product_type	varchar	The type of the product.
product_sub_type	varchar	The subtype of the product.
product_name	varchar	The name of the product.
product_box	varchar	The packaging of the product.
shipping_date	datetime	The shipping date.

Business implication

Script template name: get_top_n

This script template uses the specified sales data table as the table-type input parameter, the number of the top cities as the string-type input parameter, and the total sales amount of the cities for ranking. By using this SQL code process, you can obtain the rankings of the specified top cities in each region with ease.

Script template parameters

Input parameter 1

- Parameter name: myinputtable
- Type: table

Input parameter 2

- Parameter name: topn
- Type: string

Output parameter 3

- Parameter name: myoutput
- Type: table

Parameter definition:

- area_id string
- city_id string
- order_amt double
- rank bigint

You can execute the following statement to create a table for storing the sales data of a specified number of top cities:

```
CREATE TABLE IF NOT EXISTS company_sales_top_n
(
area STRING COMMENT 'Region',
city STRING COMMENT 'City',
sales_amount DOUBLE COMMENT 'Sales amount',
rank BIGINT COMMENT 'Ranking'
)
COMMENT 'Company sales rankings'
PARTITIONED BY (pt STRING COMMENT '')
LIFECYCLE 365;
```

Example of defining an SQL code process

```
INSERT OVERWRITE TABLE @@{myoutput} PARTITION (pt='${bizdate}')
    SELECT r3.area_id,
    r3.city_id,
    r3.order_amt,
    r3.rank
from (
SELECT
    area_id,
    city_id,
    rank,
    order_amt_1505468133993_sum as order_amt ,
    order_number_150546813****_sum,
    profit_amt_15054681****_sum
FROM
    (SELECT
    area_id,
    city_id,
    ROW_NUMBER() OVER (PARTITION BY r1.area_id ORDER BY r1.order_amt_1505468133993_sum DESC
)
AS rank,
    order_amt_15054681****_sum,
    order_number_15054681****sum,
    profit_amt_1505468****_sum
FROM
    (SELECT area AS area_id,
     city AS city_id,
     SUM(order_amt) AS order_amt_1505468****_sum,
     SUM(order_number) AS order_number_15054681****_sum,
     SUM(profit_amt) AS profit_amt_1505468****_sum
FROM
    @@{myinputtable}
WHERE
    SUBSTR(pt, 1, 8) IN ( '${bizdate}' )
GROUP BY
    area,
    city )
    r1 ) r2
WHERE
    r2.rank >= 1 AND r2.rank <= @@{topn}
ORDER BY
    area_id,
    rank limit 10000) r3;
```

Sharing scope

Script templates can be shared within a workspace or made public.

By default, a deployed script template is visible and available to users within the current workspace. The developer of a script template can click the **Publish Snippet** icon to make the general-purpose script template public to the current tenant account so that all users under the account can view and use the script template.

You can view the **Publish Snippet** icon in the toolbar of the configuration tab of a script template. If the icon is clickable, the script template is made public.

Use of script templates

For more information about how to use a developed script template, see Use components.

Reference records

In the script template list, double-click a script template. On the configuration tab that appears, click the **Snippet Nodes** tab in the right-side navigation pane to view the reference records of the script template.

4.7.2. Use a script template

To improve development efficiency, you can create data analytics nodes by using the script templates provided by workspace members and tenants.

Note the following points when you use script templates:

- The script templates provided by members of the current workspace are available on the **Workspace-Specific** tab.
- The script templates provided by tenants are available on the **Public** tab.

GUI elements

Icon or tab	Description
Save icon	Saves the settings of the current script template.
Steal Lock icon	Allows you to steal the lock of the current script template and then edit it if you are not the owner of the script template.
Submit icon	Commits the current script template to the development environment.
Publish Snippet icon	Makes the current general-purpose script template public to the current tenant account so that all users under the account can view and use the script template.
Parse I/O Parameters icon	Parses input and output parameters from the code. **Note** Typically, the parameters entered here are table names instead of scheduling parameters.
Run icon	Runs the current script template in the development environment.
Stop icon	Stops running the current script template.
Format icon	Formats the code based on keywords.
Parameters tab	Allows you to view the basic information and set input and output parameters for the current script template.

Icon or tab	Description
Versions tab	Allows you to view the deployed versions of the current script template.
Snippet Nodes tab	Lists the reference records of the current script template.

4.8. Custom node type

4.8.1. Overview

DataStudio supports default node types such as ODPS SQL and Shell nodes. You can also create custom node types to meet your requirements.

To create a custom node type, you need to create a custom wrapper and use it to define a custom node type.

Entry

1. Log on to the DataWorks console.
2. Click **Node Market** in the upper-right corner to go to the node configuration page.

> **Note** Only the workspace owner and administrators can access this page.

View the list of wrappers

The Wrappers page displays all the wrappers you have created. You can click **Create** in the upper-right corner to create a custom wrapper.

The values displayed in the Latest Version, Version in Development, and Version in Production Environment columns for the created wrappers follow these rules:

- If a created wrapper has not been deployed, the values of both the Version in Development and Version in Production Environment columns are **Not Deployed**.
- If a wrapper has been deployed, the version and the deployment time appear in these columns.
- If a wrapper is under deployment, the values of both the Version in Development and Version in Production Environment columns are **Deploying**.

You can click **Settings**, **View Versions**, or **Delete** in the Actions column of each wrapper.

Action	Description
Settings	You can click **Settings** to configure the wrapper. The page that appears depends on the wrapper status. The **Deploy in Production Environment** page appears if the wrapper has been deployed in the production environment.

Action	Description
View Versions	You can click **View Versions** to view all historical versions of the wrapper. • **View**: You can click this button to view the settings of the selected version. • **Roll Back**: You can click this button to roll back to the selected version. After you click this button, the system creates a new version for the wrapper. In the new version, the wrapper uses the basic settings and the resource file of the selected version. The new version number equals the latest version number among all the versions plus 1. • **Download**: You can click **Download** to download the resource file of the selected version.
Delete	If an error occurs while a node type is using the wrapper, you need to delete the node type. **Note** Before deleting a wrapper, ensure that no node type is associated with the wrapper.

Create a custom wrapper

A wrapper is the core processing logic of a node type. For example, after you compile an SQL statement in the editor for an ODPS SQL node and submit the statement, the system calls the corresponding wrapper to parse and run the statement. You need to create a wrapper before creating a custom node type. Currently, only the Java programming language is supported.

The procedure of creating a wrapper includes four steps: specify settings for the wrapper, deploy the wrapper in the development environment, test the wrapper in the development environment, and deploy the wrapper in the production environment. For more information, see Create a custom wrapper.

View the list of custom node types

The Custom Node Types page displays all custom node types in the workspace. You can click **Create** in the upper-right corner to create a custom node type. For more information, see Create a custom node type.

Currently, you can only create custom node types in DataStudio.

The workspace owner or node type creator can **change** or **delete** existing node types.

- **Change**: You can click **Change** to edit the settings for the node type as needed.
- **Delete**: You can click this button to delete the node type that no node uses. If any node uses the node type, a message appears, indicating that you need to disable the node first before deleting the node type.

Use a custom node type

After creating a custom node type, go to the **Data Analytics** page.

Move the pointer over the **Create** icon and click **Data Analytics**. In the list that appears, select the created node type to create a node.

4.8.2. Create a custom wrapper

The procedure of creating a wrapper includes four steps: specify settings for a wrapper, deploy the wrapper in the development environment, test the wrapper in the development environment, and deploy the wrapper in the production environment.

Specify settings for a wrapper

1. Click **Wrappers** in the left-side navigation pane. On the page that appears, click **Create** in the upper-right corner.
2. Specify the parameters in the **Settings** step.

Parameter	Description
Name	The name of the wrapper. It must start with a letter and can only contain letters, digits, and underscores (_).
Owner	The owner of the wrapper. You can select an owner from the workspace members. You are not allowed to edit wrappers owned by other members even if you are an administrator. Only the workspace owner can edit the wrappers of other members.
Resource Type	The type of the resource package for configuring the wrapper. Valid values: **JAR** and **Archive**. The size of the resource package can be up to 50 MB.
Resource File	The local resource file or OSS object for configuring the wrapper. **Note** The size of a local file can be up to 50 MB, and the size of a file that is stored in an OSS bucket can be up to 200 MB.
Class Name	The full path of the class for implementing the user wrapper.
Parameter Example	The parameters designed based on the JAR package you upload.
Version	The version of the configured wrapper. Select Create Version if you are creating a new wrapper. Select Overwrite Version if you are editing and rolling back a version. **Note** The version number is automatically generated.
Description	The description of the wrapper version.

3. Click **Save** and then click **Next**.

Note The settings are updated to the database after you click Save.

- If you only modify basic settings of a wrapper without changing the resource file, the modification takes immediate effect after you click Save.
- If you change the resource file, the change only applies after deployment.

Deploy the wrapper in the development environment

After you specify the parameters in the **Settings** step and click **Next**, the information in the Deploy in Development Environment step is updated accordingly. You can identify the changes by checking the file name and MD5 checksum.

Click **Deploy in Development Environment**. You can view the **deployment progress** in real time. After the wrapper is deployed, click **Next**.

Test the wrapper in the development environment

Specify the parameters for testing and click **Test** to send the parameters to the wrapper. This step is to validate the deployment and logic of the wrapper. You can also locally test the wrapper before uploading it for deployment.

After the test, review the output logs in the Test Results section on the right to determine whether the test is passed. If the test is passed, select **Test Passed** and click **Next**.

> **Note** The **Next** button is operable only after you select **Test Passed**.

Deploy the wrapper in the production environment

Click **Deploy in Production Environment**. In the **Confirm** dialog box that appears, click **OK**. The wrapper is deployed in the production environment. You can view the deployment progress in real time.

> **Note** The wrapper to be deployed in the production environment must be of the latest version, have been deployed in the development environment, and have passed the test. Otherwise, a message appears, indicating that the deployment in the production environment fails.

Click **Complete**. You can view and edit the created wrapper on the **Wrappers** page.

4.8.3. Create a custom node type

The Configure Custom Node Type page consists of three sections: Basic Information, Interaction, and Wrapper.

1. On the DataStudio page, click **Node Market** in the top navigation bar. On the page that appears, click **Custom Node Types** in the left-side navigation pane.
2. Click **Create** in the upper-right corner.
3. Specify the parameters in the **Basic Information** section.

Parameter	Description
Name	The name of the node type, which cannot be changed after being saved. Each node type has a unique name within the workspace. The name can be up to 20 characters in length, and can only contain letters, spaces, and underscores (_).
Icon	The icon of the node type.
Tabs	The template of the node type. Currently, only **Data Analytics** is available.
Folder	The folder where the node type belongs. You can select **Data Integration** or **Data Analytics**.

4. Specify the parameters in the **Interaction** section.

Parameter	Description
Shortcut Menu	○ The options to appear in the shortcut menu. The following options are selected by default: Rename, Move, Clone, Steal Lock, View Versions, Locate in Operation Center, Delete, and Submit for Review. ○ More options include Edit, Copy Resource Name, and Send to DataWorks Desktop (Shortcut).
Tool Bar	○ The options to appear in the top navigation bar. The following options are selected by default: Save, Commit, Commit and Unlock, Steal Lock, Run, Show/Hide, Run with Arguments, Stop, Reload, Run Smoke Test in Development Environment, View Smoke Test Log in Development Environment, Run Smoke Test, View Smoke Test Log, Go to Operation Center of Development Environment, and Format. ○ More options include Operation Center, Deploy, and Precompile.
Editor Type	The type of the editor. Currently, only **Editor Only** is available.
Right-Side Bar	○ The options to appear in the right-side bar. The following options are selected by default: Code Structure and Properties. ○ More options include Version, Lineage, and Parameters.
Auto Parse Option	Specifies whether to display the Auto Parse option for this type of node. If you turn on this switch, the Auto Parse option is displayed on the Properties tab. Otherwise, it is not displayed. In an automatic parsing process, the system parses the input and output of a node based on the lineage specified in the code.

5. Specify the parameters in the **Wrapper** section.

Parameter	Description
Wrapper	The wrapper used for running the type of node. Select a wrapper that has been deployed.
Editor Language	The language used for writing the code in the editor. Currently, only ODPS SQL is available.
Use MaxCompute as Engine	Specifies whether to use MaxCompute as the compute engine. If your wrapper uses MaxCompute as the compute engine, select Yes. Otherwise, select No. Default value: Yes.

6. Click **Save and Exit**. Then, go to the **Data Analytics** page to use the custom node type that is created.

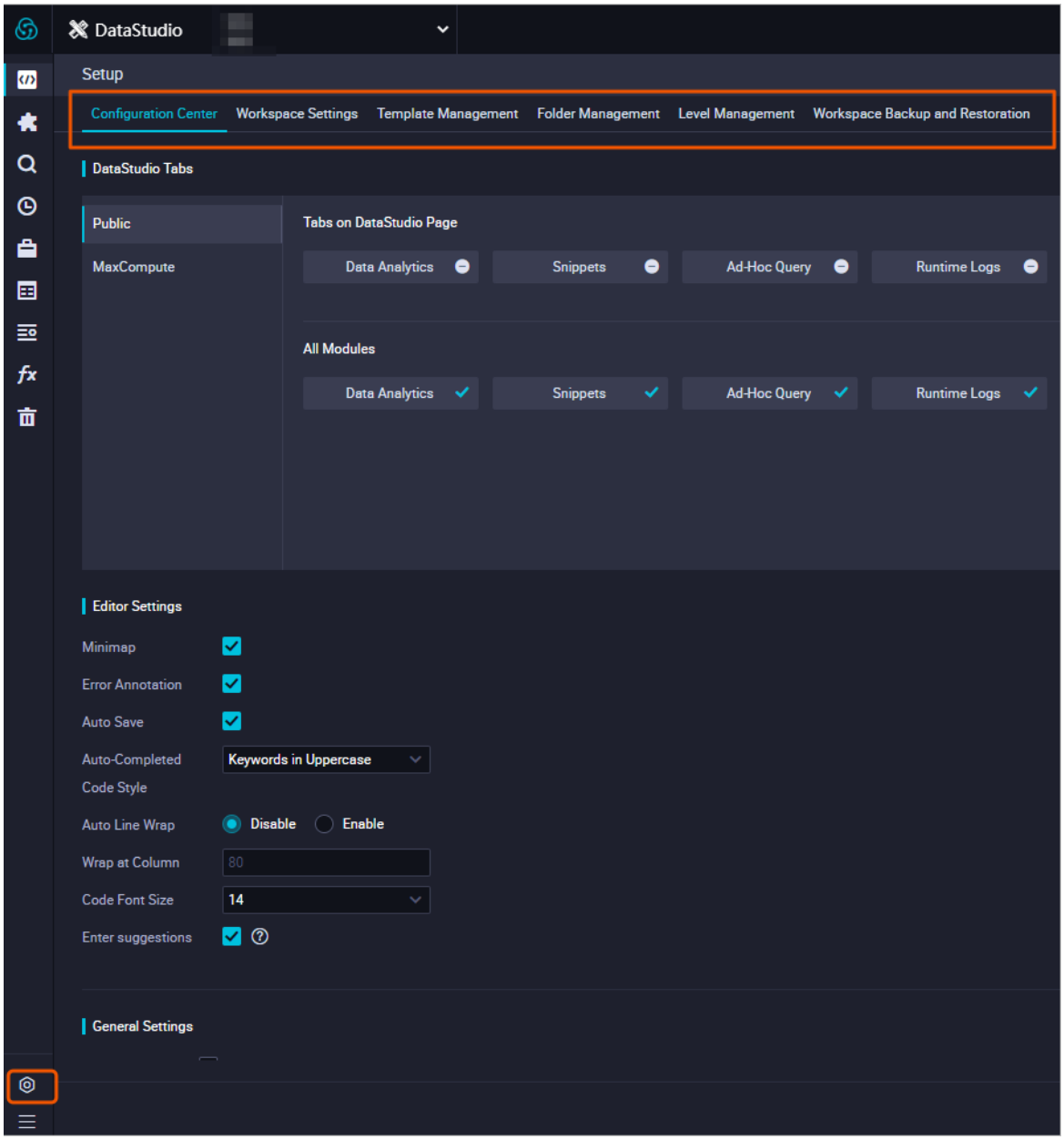
4.9. Manage configurations

4.9.1. Setup

On the Setup page, you can add and delete modules. You can also configure code templates, folders, and table levels on this page.

Procedure

1. Log on to the DataWorks console.
2. Click in the lower-left corner of the **DataStudio** page to go to the **Setup** page.



You can perform operations on the following tabs:

- Configuration Center
- Project Configuration
- Template management

- Folder management
- Level management

4.9.2. Configuration center

You can combine your DataStudio modules and specify editor settings on the Configuration Center tab.

Go to Configuration Center

1. Log on to the DataWorks console.
2. Click in the lower-left corner of the **DataStudio** page. The **Configuration Center** tab appears.

   The Configuration Center tab includes three sections: **DataStudio Tabs**, **Editor Settings**, and **General Settings**. After the configurations are completed, you can click **Apply to All Workspaces** in the lower-right corner of the page to apply the settings to all existing workspaces.

DataStudio tabs

On the **DataStudio Tabs** tab, you can add and delete public and MaxCompute functional modules, and drag modules to change their orders.

- Under **Tabs on DataStudio Page**, click next to a module to delete it. Deleted modules will not appear in the left-side navigation pane of the **DataStudio** page.
- Under **All Modules**, click the desired module to add it. Added modules will appear in the left-side navigation pane of the **DataStudio** page.

> **Note** The module settings take effect immediately for the current workspace. To make the module settings effective for all workspaces, click **Apply to All Workspaces** in the lower-right corner of the page.

Editor settings

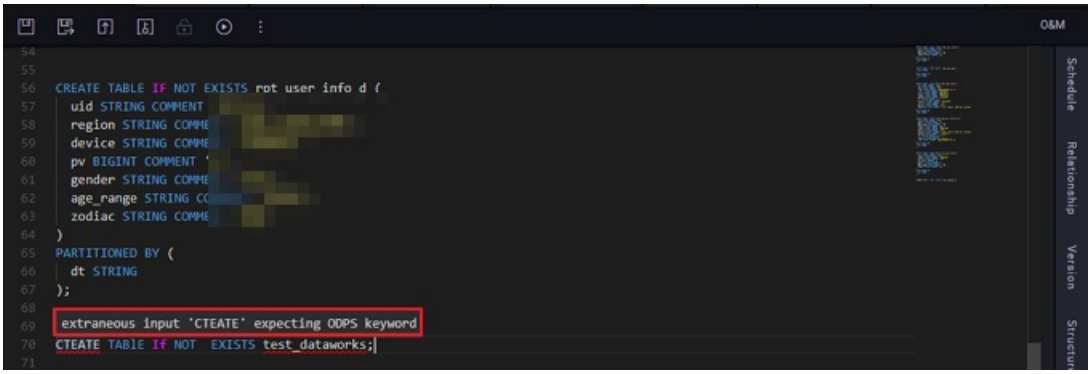
You can configure the code editor in the Editor Settings section. The editor settings take effect immediately for the current workspace without requiring you to refresh the page.

- **Minimap**

The masked code in the current interface is displayed in the minimap in the upper-right corner of the page. When the code is long, you can move the pointer to specify the code block to be displayed in the minimap.

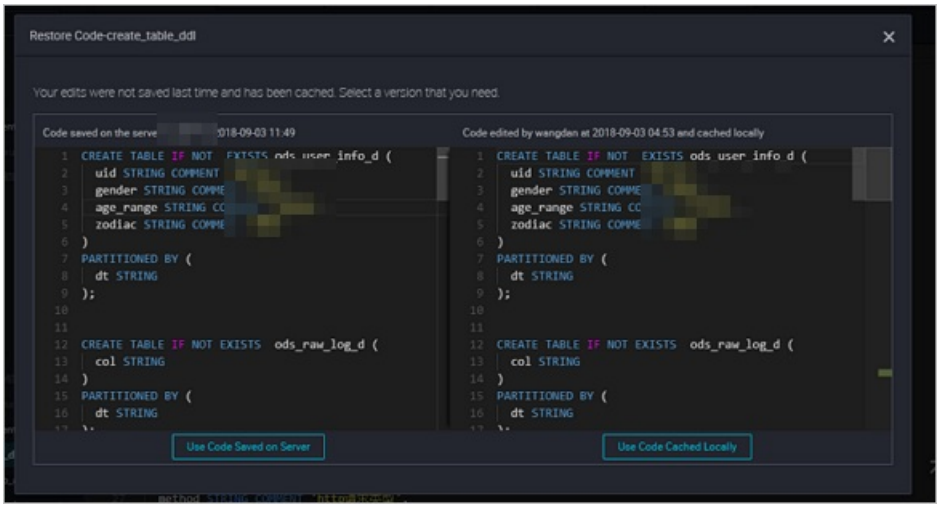
- **Error Annotation**

  If you select this check box, DataWorks marks potential syntax errors with a red squiggly line. When you see a syntax error, you can move the pointer over the underlined code to view the error message.



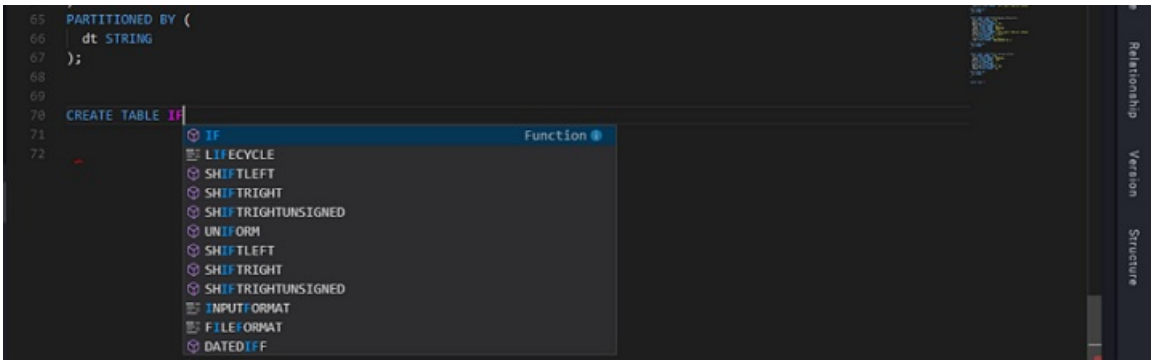
- **Auto Save**

  If you select this check box, DataWorks automatically saves the code being edited at a specific interval. In this way, if the code editor of a node is closed unexpectedly, you can click **Use Version Saved on Server** or **Use Version Saved in Local Cache** to re-open the node.



- **Auto-Completed Code Style**

You can set the code style to uppercase or lowercase as required.



- **Auto Line Wrap**

  You can set Auto Line Wrap to **Disable** or **Enable**.

- **Wrap at Column**
  - If Auto Line Wrap is set to **Disable**, the value of Wrap at Column is 80 by default, which cannot be modified.
  - If Auto Line Wrap is set to **Enable**, you can set a value for Wrap at Column as required.

- **Code Font Size**

  Valid values: 12 to 18. You can change the font size based on your habits and code size.

- **Enter suggestions**

  If you select this check box, the system automatically displays suggestions on how to set a field when you press Enter. If you clear this check box, a new line is started after you press Enter. In addition to the Enter key, you can also use the Tab key to enter prompted suggestions.

- **Auto Completion**

  You can specify whether to enable the following code hints when you enter the code:
  - **Continuous Smart Tips**: specifies whether to automatically add a space after each auto-completed term such as a keyword, table name, or field name.
  - **Keyword**: specifies whether to enable keyword hints.
  - **Syntax Template**: specifies whether to enable syntax template hints.
  - **Project**: specifies whether to enable project name hints.
  - **Table Name**: specifies whether to enable table name hints. When this feature is enabled, the system gives higher priority to tables used recently.
  - **Field**: specifies whether to enable field name hints.

General settings

- **Display Node Engine Information on DAG**

  You can specify whether to display node engine information on the DAG.

- **Theme**

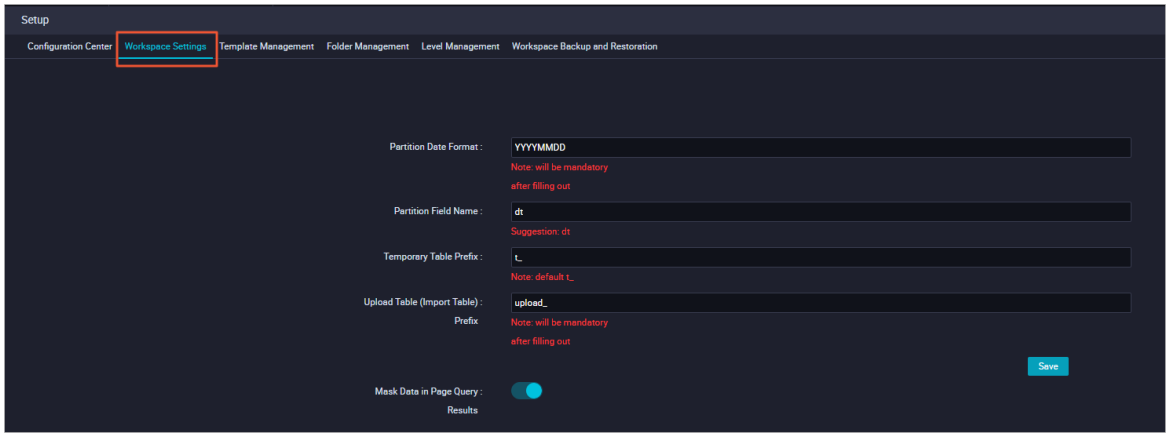
  You can set the DataStudio theme to black or white.

4.9.3. Workspace settings

The Workspace Settings tab displays five parameters: Partition Date Format, Partition Field Name, Temporary Table Prefix, Upload Table (Import Table) Prefix, and Mask Data in Page Query Results.

Go to the Workspace Settings tab

1. Log on to the DataWorks console.
2. On the **Data Analytics** tab, click in the lower-left corner.
3. On the **Setup** page, click the **Workspace Settings** tab.



Parameter	Description
Partition Date Format	The default date format of partition field values. You can modify the format as required.
Partition Field Name	The default name of a partition field.
Temporary Table Prefix	The prefix of temporary table names. By default, tables with the prefix t_ in their names are identified as temporary tables.
Upload Table (Import Table) Prefix	The prefix of the names of tables uploaded on the **DataStudio** page.
Mask Data in Page Query Results	Specifies whether to de-identify data in the query results. When the switch is turned on, the result returned for an ad hoc query node in the current workspace will be de-identified.

Enable de-identification for DataWorks workspaces

Data de-identification for DataWorks needs to be enabled in workspaces one by one. After data de-identification is enabled, the result returned for an ad hoc query node in the current workspace will be de-identified. The underlying storage data is not affected because only dynamic de-identification is performed.

> **Note** For example, data de-identification is enabled in workspace A but not workspace B. If you initiate a request in workspace B to query tables in workspace A, the query result is displayed in plaintext.

On the **Workspace Settings** tab, turn on **Mask Data in Page Query Results**. After you click **Save**, the result returned for an ad hoc query node in the current workspace will be de-identified.

> **Note** By default, the Mask Data in Page Query Results switch is turned off and you are not allowed to download de-identified data.

After data de-identification is enabled for DataWorks workspaces, the data types listed in the following table are de-identified by default.

Type	Data de-identification rule	Raw data	De-identified data
ID card number	Only the first and last digits in a 15-digit or an 18-digit ID card number are displayed in plaintext. All the other digits are displayed as asterisks (*).	512345678943215678	5**********8
Mobile number	Only the first three and last two digits in a mobile number in mainland China are displayed in plaintext. All the other digits are displayed as asterisks (*).	18112345678	181******78
Email address	If the string before the at sign (@) in an email address contains three or more characters, only the leftmost three characters are displayed in plaintext, followed by three asterisks (*). If the string before the at sign (@) contains only one or two characters, the entire string is displayed in plaintext, followed by three asterisks (*).	• eftry.abc@gmail.com • af@abc.com	• eft***@gmail.com • af***@abc.com
Bank card number	Only the last four digits in a credit card number or deposit card number are displayed in plaintext. All the other digits are displayed as asterisks (*).	• 1234576834509782 • 643257829145430986	• **********9782 • **********0986
IP address or MAC address	Only the first segment in an IP address or a MAC address is displayed in plaintext. All the other characters are displayed as asterisks (*).	• 192.000.0.0 • ab:cd:11:a3:a0:50	• 192.***. *. * • ab:**:**:**:**:**
License plate number	Only the one-character provincial abbreviation and the last three characters in a license plate number in mainland China are displayed in plaintext. All the other characters are displayed as asterisks (*).	• (One-character provincial abbreviation)AP555B • (One-character provincial abbreviation)ADP555T	• (One-character provincial abbreviation)A**55B • (One-character provincial abbreviation)A***55T

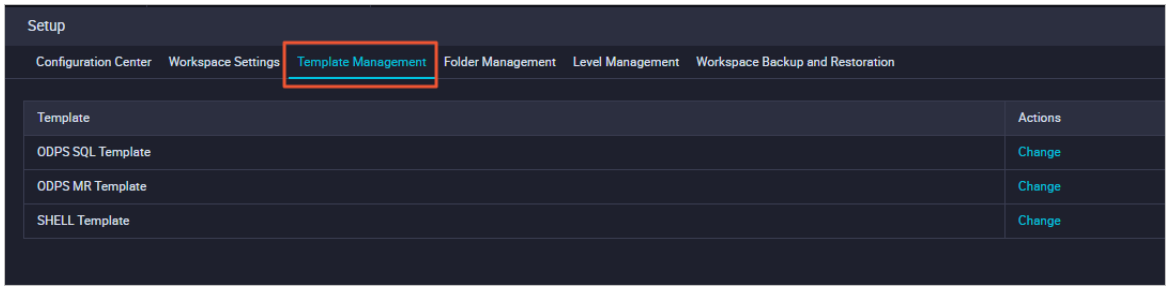
Note If you want to de-identify more data types or have special requirements on the de-identified data formats, complete your de-identification settings in Data Security Guard. The feature of de-identifying data for DataWorks workspaces must work with Data Security Guard. For more information, see Data security guard.

4.9.4. Template management

The Template Management page displays code templates. Workspace administrators can change the display formats of the templates as required.

Procedure

1. Log on to the DataWorks console.
2. On the **Data Analytics** tab, click in the lower-left corner.
3. On the **Setup** page, click the **Template Management** tab.



Note Templates are only available for ODPS SQL, ODPS MR, and Shell nodes.

4. Find the target template and click **Change** in the Actions column.
5. In the **Node Template** dialog box, enter the template as required.
6. Click **Save**.

4.9.5. Folder management

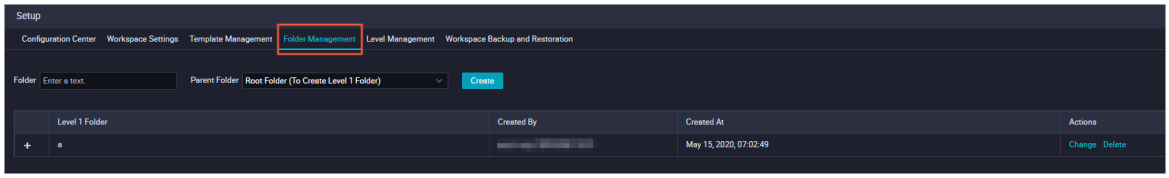
Each workspace can hold a great number of tables. For easy management, you can organize tables in two levels of folders.

Context

Folders are used to store tables. A workspace administrator can add multiple folders and classify tables by purpose and name.

Procedure

1. Log on to the DataWorks console.
2. On the **Data Analytics** tab, click in the lower-left corner.
3. On the **Setup** page, click the **Folder Management** tab.



On the page that appears, you can add, modify, and delete folders.

- Add a folder

  Enter a custom folder name in the **Folder** field, select a parent folder from the **Parent Folder** drop-down list, and then click **Create**.

- Modify a folder

  Find the target folder and click **Change** in the Actions column. In the **Change Folder** dialog box, enter a new folder name and click **OK**.

- Delete a folder

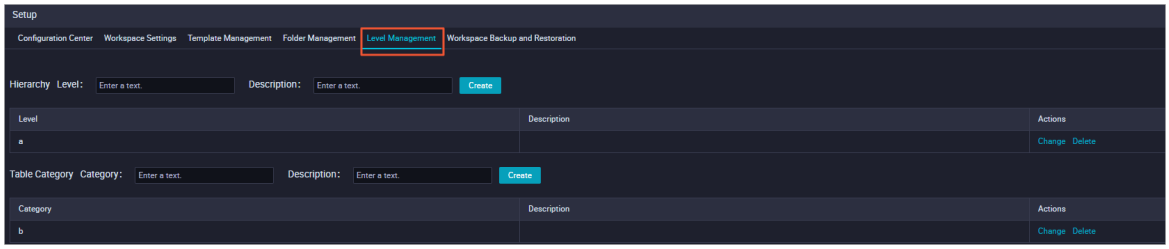
  Find the target folder and click **Delete** in the Actions column. In the **Delete Folder** message, click **OK**.

4.9.6. Level management

On the Level Management tab, you can design physical levels of tables.

Procedure

1. Log on to the DataWorks console.
2. On the **Data Analytics** tab, click  in the lower-left corner.
3. On the **Setup** page, click the **Level Management** tab.



You can classify tables based on their importance. Level management allows you to precisely locate incorrectly organized tables and ensures normal running of published jobs.

If a workspace does not contain default table levels, the workspace owner or workspace administrator must add them as required.

On the **Level Management** tab, you can add, modify, and delete table levels.

- To add a table level, perform the following steps:
  a. In the **Hierarchy** section, set **Level** and **Description**.
  b. Click **Create**.
- To modify a table level, perform the following steps:
  a. Find the target table level and click **Change** in the Actions column.
  b. In the **Change Level** dialog box, modify **Level** and **Description** as needed.

c. Click **OK**.

- To delete a table level, perform the following steps:
  a. Find the target table level and click **Delete** in the Actions column.
  b. In the **Delete Level** message, click **OK**.

On the **Level Management** tab, you can also add, modify, and delete table categories.

- To add a table category, perform the following steps:
  a. In the **Table Category** section, set **Category** and **Description**.
  b. Click **Create**.
- To modify a table category, perform the following steps:
  a. Find the target table category and click **Change** in the Actions column.
  b. In the **Change Category** dialog box, modify **Category** and **Description** as needed.
  c. Click **OK**.
- To delete a table category, perform the following steps:
  a. Find the target table category and click **Delete** in the Actions column.
  b. In the **Delete Category** message, click **OK**.

4.9.7. Workspace backup and restore

On the Workspace Backup and Restoration tab, you can migrate code between workspaces. This topic describes how to back up and restore a workspace.

Prerequisites

Workspaces are created. For more information, see Create a workspace.

Go to the Workspace Backup and Restoration tab

1. Log on to the DataWorks console.
2. On the **Data Analytics** tab, click  in the lower-left corner.
3. On the **Setup** page, click the **Workspace Backup and Restoration** tab.

   On the page that appears, you can back up and restore workspaces.

   - On the Backup tab, you can compress the node code, node dependencies, resources, and functions in a workspace into one package.
   - On the Restore tab, you can restore a workspace to its original scheduling settings. After the workspace is restored, all nodes in the workspace are saved but not committed.

Back up a workspace

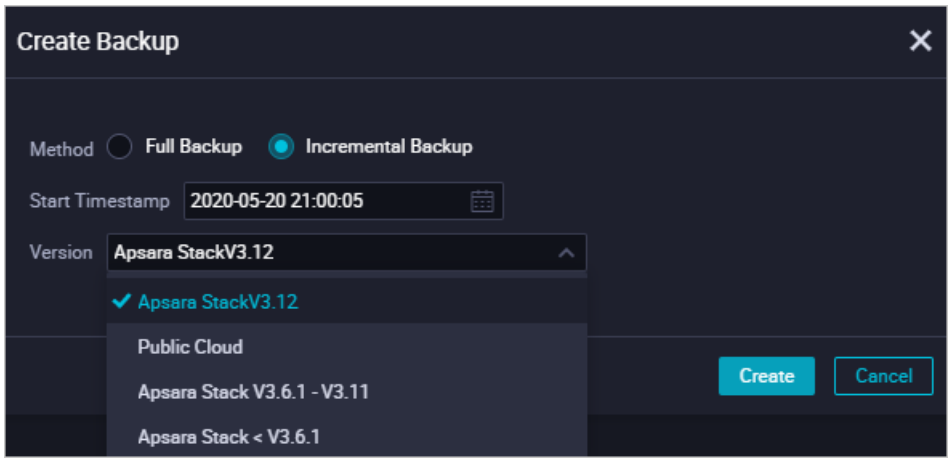
A workspace backup is a compressed package containing the node code, node dependencies, resources, and functions in the workspace.

- Only workspace administrators can export backups and restore data from backups.
- Workflows and node groups of earlier versions cannot be backed up. We recommend that you use the latest version for data analytics.
- A node backed up to a path in the workspace will override the original node with the same name in

the path. We recommend that you create another workspace to restore data.

- Data in tables is not backed up when you back up a workspace. You can synchronize the table data in the following ways:
  - Click the **Workspace Manage** icon in the upper-right corner. On the page that appears, click **Data Source** and configure a MaxCompute connection. Then, create a sync node to back up the data.
  - In workspace A, run the data definition language (DDL) statement `create table select * from workspace B. Table name` to migrate data.

1. On the **Workspace Backup and Restoration** tab, click **Backup**.
2. Click **Create Backup** in the upper-right corner.
3. In the **Create Backup** dialog box, set the parameters as required.



Parameter	Description
Method	The method used to back up the workspace. Valid values: ○ **Full Backup**: Back up all the node code, node dependencies, resources, and functions in the workspace. ○ **Incremental Backup**: Back up all the new or modified nodes from the timestamp specified by the **Start Timestamp** parameter to the current time. **Note** If you use the incremental backup method, make sure that the dependencies between incremental sync nodes are correct. Otherwise, the workspace may fail to be restored. We recommend that you set this parameter to **Full Backup**.
Start Timestamp	The start time point at which data in the workspace is backed up. This parameter is available only when you set **Method** to **Incremental Backup**.
Version	The version of the workspace to be backed up. Valid values: **Apsara StackV3.12**, **Public Cloud**, **Apsara Stack V3.6.1 - V3.11**, and **Apsara Stack < V3.6.1**.

4. Click **Create**.

   After the data is backed up, click **Download** to download the backup data to a local device.

Restore a workspace

1. On the **Workspace Backup and Restoration** tab, click **Restore**.
2. Click **Restore** in the upper-right corner.
3. In the **Restore** dialog box, click **Select File**.

   > **Note** You can upload the compressed package that you previously backed up to the workspace.

4. Click **Restore**.
5. Click **Set Compute Engine Mapping**. In the dialog box that appears, set the mapping between the compute engines of the current workspace and the destination workspace.
6. Click **OK**.

   If the workspace that you backed up contains multiple compute engines, the system scans all compute engine instances during restoration. The system only restores nodes of the existing compute engines in the workspace to be restored. In this case, you must configure the mappings between the compute engines before restoring the destination workspace.

   > **Note**
   > - If the workspace to be restored does not contain a compute engine type such as E-MapReduce, or no instance is available for the compute engine type, nodes of this engine type are not restored.
   > - Compute engine mappings must be configured for custom node types.

4.10. Deploy

4.10.1. Deploy nodes

In a rigorous data development process, developers develop and debug code and configure dependencies and scheduling properties for nodes in the development environment. Then, developers commit and deploy the nodes to run them in the production environment.

DataWorks workspaces in standard mode can process data seamlessly from the development environment to the production environment within a single workspace. We recommend that you use workspaces in standard mode to develop and produce data.

Deploy nodes in a workspace in standard mode

Each DataWorks workspace in standard mode is linked with two MaxCompute projects, one as the development environment and the other as the production environment. You can directly commit and deploy nodes from the development environment to the production environment.

Follow these steps:

1. On the **DataStudio** page, configure and debug the code of nodes. Then, double-click the target workflow in the left-side navigation pane. On the dashboard of the workflow that appears, click the **Submit** icon to check whether the dependencies between nodes are correct and commit the nodes.
2. After the nodes are committed, click the **Deploy** icon.

3. On the **Create Deploy Task** page that appears, select the target nodes and click **Add to List**. The nodes are added to the to-be-deployed node list.

   You can search for nodes by condition, such as the committer, node type, change type, time when a node is committed, node name, and node ID. If you click Deploy Selected, the selected nodes are deployed to the production environment.

4. Click **View List**. In the Nodes to Deploy dialog box that appears, click **Deploy All**. All nodes in the list are deployed to the production environment.

   > **Note** Workspaces in standard mode protect tables in the production environment from being manipulated, and therefore provide the stable, secure, and reliable production environment. We recommend that you use workspaces in standard mode to deploy and run nodes.

Clone nodes between workspaces in basic mode

You cannot deploy nodes in workspaces in basic mode. If you want to isolate the development environment from the production environment for workspaces in basic mode, create two workspaces, one for development and the other for production. You can clone nodes from the development workspace to the production workspace.

As shown in the following figure, two workspaces in basic mode are created, one for development and the other for production. You can use the cross-workspace cloning feature to clone nodes from Workspace A to Workspace B, and then commit the cloned nodes to the scheduler for scheduling.

> **Note**
> - Permission requirement: Only workspace administrators and Resource Access Management (RAM) users who have the O&M permissions can clone nodes.
> - Workspace type: You can only clone nodes in workspaces in basic mode, but cannot clone those in workspaces in standard mode.
> - Prerequisites: The source workspace in basic mode and the destination workspace in basic mode are created.

1. Commit nodes.

   After you create and configure nodes in the source workspace, commit the nodes on the dashboard of the target workflow.

2. Click **Cross-Workspace Cloning**.
3. On the Create Clone Task page that appears, select the target nodes and the destination workspace, and then click **Add to List**.
4. Clone the nodes. Click **View List**. In the To-Be-Cloned Nodes dialog box that appears, check the nodes to be cloned and click **Clone All**.

   In the Create Clone Task dialog box that appears, click **Clone**.

5. View the cloned nodes.

   You can view the successfully cloned nodes on the View Clone Tasks page of the source workspace.

   Switch to the destination workspace. You can find that the nodes are cloned from the source workspace.

4.10.2. Overview of cross-workspace cloning

For workspaces under the same Apsara Stack tenant account, you can use the cross-workspace cloning feature to clone and deploy workflows across these workspaces. You can also use this feature to clone nodes, such as computing or sync nodes, across workspaces. This topic describes how to process the dependencies between nodes during cross-workspace cloning.

If you clone nodes across workspaces by using the **cross-workspace cloning** feature, DataWorks automatically changes the output names of the cloned nodes in the destination workspace to distinguish nodes in different workspaces under the same Apsara Stack tenant account. This allows you to successfully clone node dependencies.

> **Note** **Cross-workspace cloning** cannot be used to clone nodes across workspaces in different regions.

You can set the owner of cloned nodes in the destination workspace to **Default** or **Clone Task Creator**.

- If you clone nodes owned by the workspace administrator:

  After the nodes are cloned to the destination workspace, their owner is set to the original owner preferentially. If the original owner is not added to the destination workspace, you will become the owner.

- If you clone nodes owned by yourself:

  After the nodes are cloned to the destination workspace, their owner is set to you preferentially. If you are not added to the destination workspace, you are asked whether to change the owner. If you agree to change the owner, you will be added to the destination workspace and become the owner of the cloned nodes. If you do not agree to change the owner, the clone task is canceled.

Clone a workflow

Assume that the output of the task_A node in the project_1 workspace is project_1.task_A_out. If you clone a workflow that contains the task_A node to the destination workspace project_2, the node output name changes to project_2.task_A_out in the destination workspace.

Clone node dependencies

Assume that the task_ B node in the project_1 workspace depends on the task_A node in the project_3 workspace. If you clone the task_B node in the project_1 workspace to the destination workspace project_2, the dependency between the task_A and task_B nodes is also cloned. The task_B node in the project_2 workspace also depends on the task_A node in the project_3 workspace.

4.10.3. Clone nodes across workspaces

This topic describes how to clone nodes across workspaces with an example of cloning a workflow from one workspace to another.

Prerequisites

Two workspaces named Weisong_dataworks_test and Weisong_dataworks_test2 respectively are created. For more information about how to create a workspace, see Create a workspace.

Context

You can clone nodes across workspaces in the following scenarios:

- Clone nodes from a workspace in the basic mode to another workspace in the basic mode.
- Clone nodes from a workspace in the basic mode to another workspace in the standard mode.

After you clone a node, the folder and workflow to which the node belongs are cloned to the destination workspace. Any change to the node, folder, or workflow can also be cloned to the destination workspace.

Procedure

1. Log on to the DataWorks console. On the DataStudio page that appears, switch to the Weisong_dataworks_test workspace in the top navigation bar.
2. Select the target workflow.

   In the Data Analytics section, double-click the target workflow. On the workflow configuration page that appears, click **Cross-Workspace Cloning** in the upper-right corner. The Create Clone Task page appears.
3. Select the destination workspace, node type, and change type.

   On the **Create Clone Task** page, set **Target Workspace** to Weisong_dataworks_test2. Select the node type and change type of the node for the clone task as required, and select one or more target nodes that appear in the list. Then, click **Clone Selected**.
4. In the **Create Clone Task** dialog box that appears, check the destination workspace, target node, and change type, and then click **Clone**.
5. After the system message that indicates the target node is cloned successfully and is being committed to the destination workspace appears, switch to the Weisong_dataworks_test2 workspace. In the **Data Analytics** section, view the workflow that has been successfully cloned to the current workspace.

4.11. Create an ad hoc query node

The Ad-Hoc Query tab allows you to test your code in the development environment. You can check for errors and check whether your code works as expected.

Context

You do not need to commit and deploy ad hoc query nodes or configure scheduling policies for ad hoc query nodes. You can configure scheduling policies only for nodes created under **Business Flow** on the **Data Analytics** tab.

Create a folder

1. Log on to the DataWorks console.
2. On the left-side navigation submenu, click the **Ad-Hoc Query** icon.

   Click ☰ in the lower-left corner to show or hide the left-side navigation pane.
3. On the Ad-Hoc Query tab, move the pointer over +Create and select **Folder**.
4. In the **Create Folder** dialog box, set **Folder Name** and **Location**.

> Note
>
> - The folder name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.
> - DataWorks supports multi-level folders. You can save a newly created folder under another folder that already exists.

5. Click **Commit**.

Create an ad hoc query node

You can create **ODPS SQL** and **Shell** nodes on the **Ad-Hoc Query** tab. This topic describes how to create an ODPS SQL node.

1. On the **Ad-Hoc Query** tab, right-click the target folder and choose **Create Node > ODPS SQL**.
2. In the **Create Node** dialog box, set **Node Name** and **Location**.

> Note The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.

3. Click **Commit**.
4. On the node configuration tab that appears, enter an SQL statement.
5. Click  in the toolbar.

4.12. View runtime logs

The Runtime Logs tab displays the records of all nodes that have been run in the last three days. You can click a node to view its runtime logs.

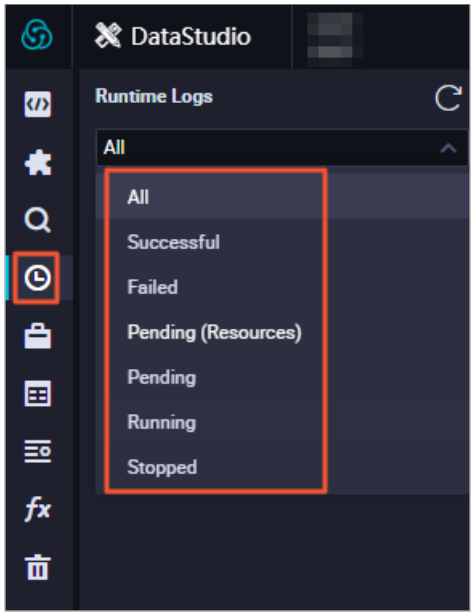
Context

The runtime logs are retained for only three days.

Procedure

1. Log on to the DataWorks console.
2. On the left-side navigation submenu, click the **Runtime Logs** icon.

   Click  in the lower-left corner to show or hide the left-side navigation pane.
3. Select a node state from the drop-down list to view the runtime logs of nodes in the specified state.



4. Click a record to view the runtime log on the right.

   If you need to save the SQL statements in the runtime log, click in the toolbar. In the **Create Node** dialog box, set the parameters and click **Commit** to save the SQL statements that have been run as an ad hoc query node.

4.13. View tenant tables

On the Tenant Tables tab, you can view tables of all workspaces of the current tenant account.

Prerequisites

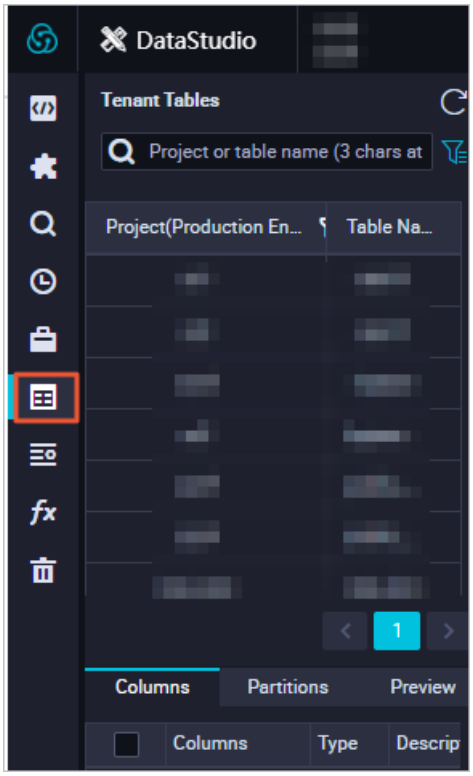
The **Tenant Tables** tab appears only after you bind a MaxCompute compute engine on the **Project Management** page. For more information, see Configure a workspace.

Procedure

1. Log on to the DataWorks console.
2. On the left-side navigation submenu, click the **Tenant Tables** icon.

   Click in the lower-left corner to show or hide the left-side navigation pane.

3. View MaxCompute tenant tables.



Parameter or tab	Description
Project Name	The name of the workspace in the corresponding environment. Click next to the search box and select the target environment to switch to the environment. **Note** ○ For a workspace in standard mode, the **Tenant Tables** tab displays tables in both the development environment and the production environment. For a workspace in basic mode, the **Tenant Tables** tab displays only the tables in the production environment. ○ The current environment is marked in blue.
Table Name	The name of the table in the corresponding workspace.
Columns tab	Displays the name, data type, and description of fields in the table.
Partitions tab	Displays the partition information of the current table. A maximum of 60,000 partitions are supported. If you have specified the TTL for partitions, the number of partitions depends on the TTL. **Notice** The partition information is displayed only for MaxCompute tenant tables.

Parameter or tab	Description
Preview tab	Displays the data of the current table. **Notice** You can preview only the data of MaxCompute tenant tables.

4.14. Manage tables

This topic describes how to view, modify, and delete MaxCompute tables, and the basic knowledge about data hierarchy.

Prerequisites

The **Workspace Tables** tab appears only after you bind a MaxCompute compute engine on the **Project Management** page. For more information, see Configure a workspace.

Manage tables

1. Log on to the DataWorks console.
2. On the left-side navigation submenu, click the **Workspace Tables** icon.

   Click in the lower-left corner to show or hide the left-side navigation pane.
3. View and manage tables.

   The following section describes how to view, modify, and delete a MaxCompute table. For more information about how to create a table, see Create a MaxCompute table.

Operation	Description
View a table	Click next to the search box and select the target environment to switch to the environment. **Note** ◦ For a workspace in standard mode, the **Workspace Tables** tab displays tables in both the development environment and the production environment. For a workspace in basic mode, the **Workspace Tables** tab displays only the tables in the production environment. ◦ The current environment is marked in blue. Double-click a table to view its details on the table configuration tab.
Import data to a table	On the **Workspace Tables** tab, click to import data to a table. For more information, see Create tables and import data.

Divide a data warehouse into layers

In the **Physical Model** section of the configuration tab of a table, you can define table layers for a data warehouse. This allows you to have better planning and control over your data.

Typically, a data warehouse consists of the following layers:

- ODS: The ODS layer stores raw data of the source system based on the original data structure. The ODS layer serves as the data staging area of the data warehouse. It imports basic data to MaxCompute and records historical changes of basic data.
- CDM: The CDM layer consists of the dimension data (DIM), data warehouse detail (DWD), and data warehouse service (DWS) layers. The CDM layer processes and integrates the data of the ODS layer to define conformed dimensions, create reusable detailed fact tables for data analysis and statistics collection, and aggregate common metrics.
  - The DIM layer defines conformed dimensions for an enterprise based on the concepts of dimensional modeling. It reduces the risk of inconsistent statistical criteria and algorithms.

    Tables at the DIM layer are also called logical dimension tables. Generally, each dimension corresponds to a logical dimension table.
  - The DWS layer is driven by analyzed subjects during data modeling. Based on the metric requirements of upper-layer applications and products, the DWS layer creates fact tables to aggregate common metrics and builds a physical data model by using wide tables. The DWS layer creates statistical metrics in compliance with uniform naming conventions and statistical criteria, provides common metrics for the upper layer, and generates aggregate wide tables and detailed fact tables.

    Tables at the DWS layer are also called logical aggregate tables, which are used to store derived metrics.
  - The DWD layer is driven by business processes during data modeling. It creates detailed fact tables at the finest granularity based on each specific business process. In combination with the data usage habits of an enterprise, you can duplicate some key attribute fields of dimensions in detailed fact tables to create wide tables.

    Tables at the DWD layer are also called logical fact tables.
- ADS: The ADS layer stores personalized statistical metrics of data products. It processes the data of the CDM and ODS layers.

4.15. View built-in functions

The Built-In Functions tab displays functions built in MaxCompute. You can view the types, description, and examples of functions on this tab.

Procedure

1. Log on to the DataWorks console.
2. On the left-side navigation submenu, click the **Built-In Functions** icon.

   Click  in the lower-left corner to show or hide the left-side navigation pane.
3. View the types, description, and examples of the built-in functions.

   Functions are categorized into aggregate functions, analytic functions, date functions, mathematical functions, string functions, and other functions. The preceding functions are built in MaxCompute. You can click a function to view its description.

4.16. Manage deleted nodes

DataWorks provides a recycle bin to store all deleted nodes in the current workspace. You can restore or permanently delete the nodes.

Go to the Recycle Bin tab

1. Log on to the DataWorks console.
2. On the left-side navigation submenu, click the **Recycle Bin** icon.

   Click in the lower-left corner to show or hide the left-side navigation pane.
3. View all the deleted nodes in the current workspace.

   On this tab, you can delete or restore a deleted node.

> Notice
> - The recycle bin displays only 100 nodes. If more than 100 nodes are deleted, the nodes deleted earlier are deleted permanently from the recycle bin.
> - Deleted node groups are not displayed in the recycle bin.

Restore a node in the recycle bin

1. On the **Recycle Bin** tab, right-click a deleted node.
2. Select **Restore**.
3. In the **Restore Node** message, click **OK**.

> Note After you restore a node, a new node ID is generated for scheduling and all the information about the node is restored.

Permanently delete a node from the recycle bin

1. On the **Recycle Bin** tab, right-click a deleted node.
2. Select **Delete**.
3. In the **Delete** message, click **OK**.

> Note Nodes permanently deleted from the recycle bin cannot be restored. The recycle bin displays only deleted nodes.

4.17. Create a manually triggered workflow

In a manually triggered workflow, all nodes must be manually triggered, and cannot be automatically scheduled by DataWorks. Therefore, you do not need to specify parent nodes or outputs for nodes in manually triggered workflows.

Create a manually triggered workflow

1. Log on to the DataWorks console.
2. On the left-side navigation submenu, click the **Manually Triggered Workflows** icon.

   Click ☰ in the lower-left corner to show or hide the left-side navigation pane.
3. Right-click **Manually Triggered Workflows** and select **Create Workflow**.
4. In the **Create Workflow** dialog box, set **Workflow Name** and **Description**.

   > **Notice** The workflow name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.
5. Click **Create**.

Composition of a manually triggered workflow

> **Note** We recommend that you create a maximum of 100 nodes in a manually triggered workflow.

A manually triggered workflow consists of the nodes of the following modules. After you create a manually triggered workflow, open this workflow and create nodes of various types for each module. For more information, see Node types.

- **Data Integration**

  Double-click **Data Integration** under the created workflow to view all the data integration nodes.

  Right-click **Data Integration** and choose **Create > Batch Synchronization** to create a batch sync node. For more information, see Create a batch sync node.
- **MaxCompute**

  The MaxCompute compute engine consists of data analytics nodes, such as ODPS SQL, SQL Snippet, ODPS Spark, PyODPS, ODPS Script, and ODPS MR nodes. You can also view and create tables, resources, and functions.
  - **Data Analytics**

    Show **MaxCompute** under the created workflow and right-click **Data Analytics** to create a data analytics node. For more information, see Create an ODPS SQL node, Create an SQL Snippet node, Create an ODPS Spark node, Create a PyODPS node, Create an ODPS Script node, and Create an ODPS MR node.
  - **Table**

    Show **MaxCompute** under the created workflow and right-click **Table** to create a table. You can also view all the tables created for the current MaxCompute compute engine. For more information, see Create a MaxCompute table.
  - **Resource**

    Show **MaxCompute** under the created workflow and right-click **Resource** to create a resource. You can also view all the resources created for the current MaxCompute compute engine. For more information, see Create, reference, and download resources.

- **Function**

  Show **MaxCompute** under the created workflow and right-click **Function** to create a function. You can also view all the functions created for the current MaxCompute compute engine. For more information, see Register a UDF.

- **Algorithm**

  Click the created workflow and right-click **Algorithm** to create an algorithm. You can also view all the PAI nodes created in the current manually triggered workflow. For more information, see Create a PAI node.

- **General**

  Click the created workflow and right-click **General** to create relevant nodes. For more information, see Create a Shell node and Create a zero-load node.

- **UserDefined**

  Click the created workflow and right-click **UserDefined** to create relevant nodes. For more information, see Create a Hologres development node.

GUI elements

The following table describes the icons and tabs on the Manually Triggered Workflows page.

No.	Icon or tab	Description
1	**Submit icon**	Commits all nodes in the current manually triggered workflow.
2	**Run icon**	Runs all nodes in the current manually triggered workflow. Nodes in this workflow do not have dependencies, and therefore they can run at a time.
3	**Stop icon**	Stops all running nodes in the current manually triggered workflow.

No.	Icon or tab	Description
4	**Deploy icon**	Navigates to the Deploy page. On this page, you can deploy some or all nodes that are committed but not deployed to the production environment. **Note** This icon is available only when the workspace is in standard mode.
5	**Go to Operation Center icon**	Navigates to the Operation Center page.
6	**Box. icon**	Box-selects a node group consisting of required nodes.
7	**Refresh icon**	Refreshes the page of the current manually triggered workflow.
8	**Auto Layout icon**	Sorts the nodes in the current manually triggered workflow.
9	**Zoom In icon**	Zooms in the current page.
10	**Zoom Out icon**	Zooms out the current page.
11	**Search icon**	Searches for a node in the current manually triggered workflow.
12	**Toggle Full Screen View icon**	Displays nodes in the current manually triggered workflow in the full screen.
13	**Show Engine Information/Hide Engine Information icon**	Shows or hides engine information.
14	**Workflow Parameters tab**	Allows you to set parameters. Parameters set on this tab have a higher priority than those specified on the corresponding node configuration tab. If two values are set separately, the value set on the Workflow Parameters tab takes effect.
15	**Change History tab**	Allows you to view the operation records of all nodes in the current manually triggered workflow.
16	**Versions tab**	Allows you to view the deployment records of all nodes in the current manually triggered workflow.

4.18. Editor keyboard shortcuts

This section describes keyboard shortcuts available for the code editor.

Google Chrome in Windows OS

`Ctrl + S`: Save changes to a node.

`Ctrl + Z` : Undo an action.

`Ctrl + Y` : Redo an action.

`Ctrl + D` : Select occurrences.

`Ctrl + X` : Cut a line.

`Ctrl + Shift + K` : Delete a line.

`Ctrl + C` : Copy a line.

`Ctrl + I` : Select a line.

`Alt + Shift + Drag` : Select a block.

`Alt + Click` : Insert an additional cursor.

`Ctrl + Shift + L` : Select all occurrences.

`Ctrl + F` : Search for text in a node.

`Ctrl + H` : Replace text in a node.

`Ctrl + G` : Locate a line.

`Alt + Enter` : Select all matched strings.

`Alt + Up or down arrow` : Move a line up or down.

`Alt + Shift + Up or down arrow` : Duplicate a line.

`Ctrl + Shift + K` : Delete a line.

`Ctrl + Enter or Ctrl + Shift + Enter` : Insert a line break downwards or upwards.

`Ctrl + Shift + Back slash (\)` : Jump to the parenthesis, bracket, or brace that matches the adjacent one.

`Ctrl + Left bracket (]) or right bracket ([)` : Increase or decrease the indent of a line.

`Home or End` : Move the cursor to the beginning or end of a line.

`Ctrl + Home or End` : Move the cursor to the top or bottom of a node.

`Ctrl + Left or Right arrow` : Move the cursor one word to left or right.

`Ctrl + Shift + Left bracket (]) or right bracket ([)` : Hide or show a block.

`Ctrl + K + Left bracket (]) or right bracket ([)` : Hide or show sub-blocks in a block.

`Ctrl + K + 0 or J` : Hide or show all blocks.

`Ctrl + Slash (/)` : Comment out or uncomment the selected lines or blocks.

Google Chrome in Mac OS

`Command-S` : Save changes to a node.

`Command-Z` : Undo an action.

`Command-Y` : Redo an action.

`Command-D` : Select occurrences.

`Command-X` : Cut a line.

`Shift-Command-K` : Delete a line.

`Command-C` : Copy a line.

`Command-I` : Select a line.

`Command-F` : Search for text in a node.

`Option-Command-F` : Replace text in a node.

`Option-Up or down arrow` : Move a line up or down.

`Option-Shift-Up or down arrow` : Duplicate a line.

`Shift-Command-K` : Delete a line.

`Command-Enter or Shift-Command-Enter` : Insert a line break downwards or upwards.

`Shift-Command-Back slash (\)` : Jump to the parenthesis, bracket, or brace that matches the adjacent one.

`Command-Left bracket (]) or right bracket ([)` : Increase or decrease the indent of a line.

`Command-Left or right arrow` : Move the cursor to the beginning or end of a line.

`Command-Up or down arrow` : Move the cursor to the top or bottom of a node.

`Option-Left or right arrow` : Move the cursor one word to left or right.

`Option-Command-Left bracket (]) or right bracket ([)` : Hide or show a block.

`Command-K-Left bracket (]) or right bracket ([)` : Hide or show sub-blocks in a block.

`Command-K-0 or J` : Hide or show all blocks.

`Command-Slash (/)` : Comment out or uncomment the selected lines or blocks.

Insert multiple cursors and select multiple occurrences or lines

`Option-Click` : Insert an additional cursor.

`Option-Command-Up or down arrow` : Insert an additional cursor to the previous or next line.

`Command-U` : Undo a cursor-related operation.

`Option-Shift-I` : Insert a cursor at the end of each selected line.

`Command-G or Shift-Command-G` : Select the next or previous matched string.

`Command-F2` : Select the nearest character of each cursor.

`Shift-Command-L` : Select the nearest word of each cursor.

`Option-Enter` : Select all the matched strings.

`Option-Shift-Drag` : Multi-select lines

`Option-Shift-Command-Up or down arrow` : Extend a selection one line up or down.

`Option-Shift-Command-Left or right arrow` : Extend a selection one character to the left or right.

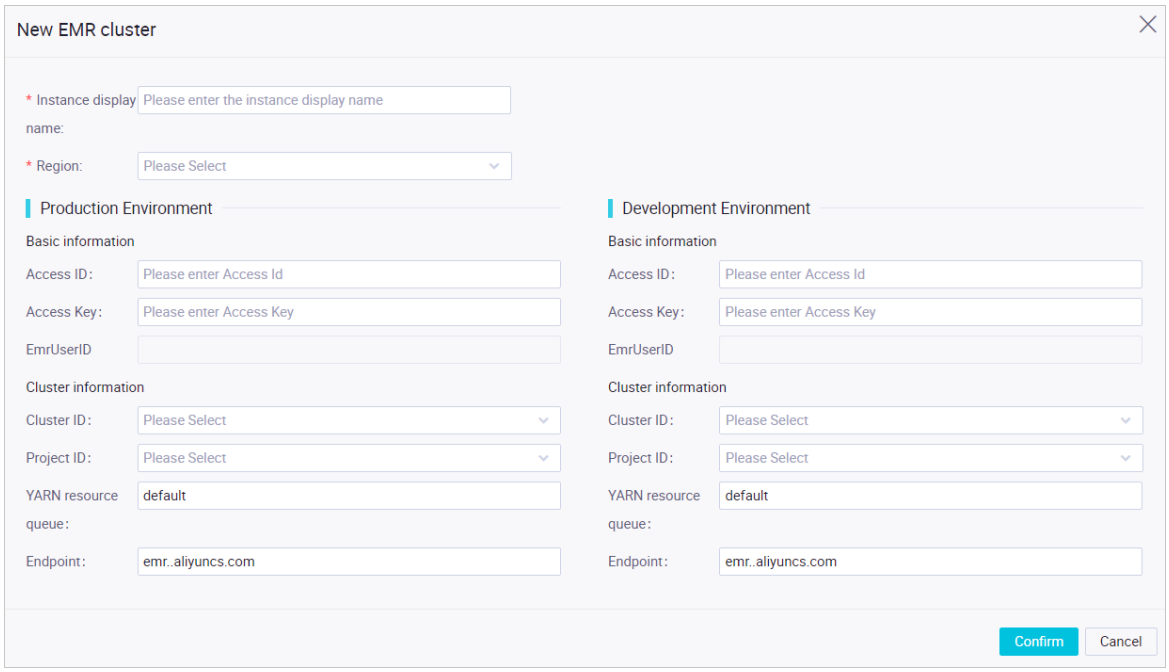
4.19. Use E-MapReduce in DataWorks

This topic describes how to use E-MapReduce in DataWorks.

Bind an E-MapReduce project to a DataWorks workspace

> **Note** Before you bind an E-MapReduce project to a DataWorks workspace, you must obtain the information about the E-MapReduce project.

1. Log on to the DataWorks console.
2. On the DataStudio page, click in the upper-right corner. The **Project Management** page appears.
3. Click the **E-MapReduce** tab in the **Computing Engine Information** section. On this tab, you can view the information about all available E-MapReduce compute engines in the current workspace.
4. Click **Add instances**.
5. In the **New EMR cluster** dialog box, set the parameters as required.



Parameter	Description
Instance display name	The name of the E-MapReduce compute engine instance.
Region	The region of the workspace.
Access ID	The AccessKey ID of the account authorized to access the E-MapReduce cluster.
Access Key	The AccessKey secret of the account authorized to access the E-MapReduce cluster.
EmrClusterID	The ID of the E-MapReduce cluster.
Cluster ID	The ID of the user who created the E-MapReduce cluster.

Parameter	Description
Project ID	The ID of the project in the E-MapReduce cluster.
YARN resource queue	The name of the resource queue in the E-MapReduce cluster. Unless otherwise specified, set the value to *default*.
Endpoint	The endpoint of the E-MapReduce cluster. You can obtain the endpoint in the Apsara Stack Operations console. For more information, see Obtain an endpoint.

6. Click **Confirm**. After the E-MapReduce cluster is bound to your workspace, you can create E-MapReduce nodes on the **DataStudio** page.

> **Note** If the binding fails, check whether the failure is caused by one of the following reasons:
>
> - The E-MapReduce user ID is bound to another tenant account.
> - The specified cluster name already exists.

Create an E-MapReduce node

E-MapReduce nodes are categorized into four types: EMR Hive, EMR Spark SQL, EMR Spark, and EMR MR. For more information, see Create an EMR MR node.

Reference resource files

E-MapReduce resource files are categorized into two resource types: EMR JAR and EMR File.

Reference E-MapReduce resource files by using the following methods:

- For EMR Hive and EMR MR nodes, add `--@resource_reference{"Resource name"}` at the first line of the code.
- For EMR Spark nodes, add `##@resource_reference{"Resource name"}` at the first line of the code.

Manage data

DataWorks allows you to query E-MapReduce metadata and synchronize the data for data development.

4.20. Migrate nodes in DataStudio

This topic describes how to migrate nodes in DataStudio that are created in a workflow of an earlier version to a workflow of a later version.

Migrate nodes in DataStudio

1. Log on to the DataWorks console. The DataStudio page appears.
2. Click **Business Flow** in the DataStudio pane to show all the created workflows. In the preceding figure, the workflow indicated by 1 is a workflow of an earlier version, and the workflow indicated by 2 is a workflow of a later version.
3. Click the desired workflow to open it and click Data Integration. Then, right-click a node that you

want to migrate in the Data Integration folder, and select **Move**.

4. In the **Move Node** dialog box, set **Location** to the path of the destination workflow of a later version.
5. Click **OK**.

5.HoloStudio

5.1. Overview

This topic describes what is HoloStudio and the features of HoloStudio.

HoloStudio, which is developed based on Hologres, is an all-in-one online analytical processing (OLAP) development platform deeply integrated with DataWorks. HoloStudio provides Hologres users with standardized and easy-to-use development services and one-stop real-time data warehouse building services by using a visualized and wizard-based user interface. In addition to standard management available in PostgreSQL, HoloStudio provides more interactive analytics features. HoloStudio supports the following features:

- Table management

  HoloStudio allows you to create a PostgreSQL table on a visualized user interface or by using SQL statements. HoloStudio also allows you to create a foreign table sourced from MaxCompute by synchronize the schema with one click and preview and analyze MaxCompute data.

- SQL Console

  The SQL Console module of HoloStudio allows you to use an SQL editor to obtain query results in seconds.

- Data analytics

  Based on the underlying capabilities of DataWorks, HoloStudio allows you to create multiple foreign tables at a time and synchronize data with one click and provides you with the one-stop, stable, and efficient extract-transform-load (ETL) service.

- Seamless connection to the Hologres console

  HoloStudio is seamlessly connected to the Hologres console. In the Hologres console, you can manage Hologres instances, users, and databases on a visualized user interface.

5.2. Bind a Hologres database to the current workspace

This topic describes how to bind a Hologres database to the current workspace.

If you need to use HoloStudio for data development, bind a Hologres database to the current workspace. Perform the following steps:

1. Log on to HoloStudio.

   On the left-side navigation submenu, click **PG management**. On the PG management tab, move the pointer over the Create icon and select **Database**.

2. In the Create Database dialog box, set relevant parameters.

   In the **Create Database** dialog box, set relevant parameters and click **Test connectivity**. If the system message indicating that the connectivity test is passed appears, the specified database is connected. Then, click **Complete**.

Parameter	Description	Remarks
Connect To	The type of the data store.	The value is automatically generated and cannot be changed.
Server	The endpoint of the Hologres instance.	You can view the endpoint on the Basic Information page of the Hologres instance in the Hologres console.
Port	The port number of the Hologres instance.	You can view the port number on the Basic Information page of the Hologres instance in the Hologres console.
Database Name	The name of the Hologres database to be bound to the current workspace.	In actual scenarios, create a Hologres database and set this parameter to the name of the database to bind the database to the current workspace.
User name	The AccessKey ID of your Apsara Stack tenant account.	You can click the username in the upper-right corner to view the AccessKey ID.
Password	The AccessKey secret of your Apsara Stack tenant account.	You can click the username in the upper-right corner to view the AccessKey secret.
JDBC Extension	The extension parameters used to establish a Java Database Connectivity (JDBC) connection to Hologres.	Example: `?preferQueryMode=simple&tcpKeepAlive=true`
Test connectivity	Tests whether the database is connected.	N/A

3. Use the Hologres database in HoloStudio.

   After the Hologres database is bound to the current workspace, click the Refresh icon on the **PG management** tab. After the database appears, you can use the database in HoloStudio.

5.3. SQL Console

The SQL Console in HoloStudio is an editor for executing SQL statements. In the SQL Console, you can execute SQL statements to analyze data in Hologres and obtain the query results. This topic describes the basic features and usage of the SQL Console in HoloStudio.

Folder

The Folder module stores new ad hoc queries, which helps you manage ad hoc queries.

In the left-side navigation pane, click **SQL Console**. Move the pointer over the Create icon and select **Folder**. In the Create Folder dialog box, enter a folder name and click Commit to create a folder. You can create an ad hoc query in the folder and execute standard SQL statements on tables. You can also right-click a table in the folder and select the relevant menu item to move, rename, or delete the table.

SQL Console

The SQL Console module allows you to create ad hoc queries and execute standard SQL statements.

1. Create an ad hoc query.

   In the left-side navigation pane, click **SQL Console**. Move the pointer over the Create icon and select **SQL Console**. In the Create Node dialog box, set relevant parameters.

   The following table describes the parameters for creating an ad hoc query.

Parameter	Description
Node Name	The name of the ad hoc query. The name can contain letters, digits, underscores (_), and periods (.).
Location	The folder where the ad hoc query is stored.
Database	The target database in which the ad hoc query is run.

2. Run the ad hoc query.

   Write the SQL statements used in the ad hoc query and click the **Run** icon to run the query. Then, you can check the query result. The following example shows how to create a table, import data to the table, and then query the table:

```
CREATE TABLE supplier (
 s_suppkey bigint NOT NULL,
 s_name text NOT NULL,
 s_address text NOT NULL,
 s_nationkey bigint NOT NULL,
 s_phone text NOT NULL,
 s_acctbal bigint NOT NULL,
 s_comment text NOT NULL,
PRIMARY KEY (s_suppkey)
);
INSERT INTO supplier VALUES
(1,    'Supplier#000000001', 'gf0JBoQDd7tgrzrddZ',    17,    '27-918-335-1736',    5755
94,    'each slyly above the careful'),
(6,    'Supplier#000000006',    'tQxuVm7s7CnK',    14,    '24-696-997-4969',    136579,
'final accounts. regular dolphins use against the furiously ironic decoys. '),
(10, 'Supplier#000000010',    'Saygah3gYWMp72i PY', 24, '34-852-489-8585', 389191, 'ing
waters. regular requests ar'),
(18, 'Supplier#00000001',    'PGGVE5PWAMwKDZw', 16, '26-729-551-1115', 704082, 'account
s snooze slyly furiously bold'),
(39, 'Supplier#000000039', 'SYpEPWr1yAFHaC91qjFcijjeU5eH',    8, '18-851-856-5633    61
1565', 88990, 'le slyly requests. special packages shall are blithely. slyly unusual pa
ckages sleep'),
(48, 'Supplier#000000048',    'FNPMQDuyuKvTnLXXaLf3Wl6OtONA6mQlWJ',    14,    '24-722-5
51-9498',563062, 'xpress instructions affix. fluffily even requests boos');
SELECT * FROM supplier;
```

The following table describes the GUI elements on the node editing tab.

GUI element	Description
SQL editor	You can write SQL statements in the SQL editor.
Save icon	You can click this icon to save all statements in the SQL editor.
Run icon	You can click this icon to execute all statements in the SQL editor. The result appears on the Result tab. You can also select an SQL statement to be executed. In this case, the system only executes this statement.
Refresh icon	You can click this icon to refresh the content in the SQL editor. The system retains only the saved content after the refresh.
Stop icon	You can click this icon to stop executing SQL statements.
Runtime Log	You can check the execution results and, if any, error messages.
Result	You can check the table content after the SQL statements are executed.

GUI element	Description
Steal Lock icon	You can click this icon to unlock the locked SQL Console.

HoloStudio also allows you to directly manage the queried data. For example, you can hide columns, copy data, and search for data.

> **Note** For statements without results returned, such as the `CREATE TABLE` statement, only operational logs are generated after the statements are executed.

5.4. PostgreSQL management

5.4.1. Manage databases

The PG management module of HoloStudio allows you to manage databases and tables with one click in a visualized manner. In addition, HoloStudio supports interactive queries with a response time within seconds. This topic describes how to use the PG management module to manage databases.

Create a database

The PG management module allows you to connect databases to HoloStudio and manage the connected databases in a visualized manner. Before you connect a database to HoloStudio, create the database in the Hologres console or by executing SQL statements in the SQL Console. The following example demonstrates how to create a database by executing SQL statements in the SQL Console and connect the database to HoloStudio:

1. Create an ad hoc query.

   Log on to the DataWorks console as a superuser and go to the HoloStudio page. In the SQL Console, execute the following SQL statement:

   ```
   create database dbname;
   create database testdb; // Create a database named testdb.
   ```

2. Bind the Hologres database to the current workspace.

   On the HoloStudio page, click **PG management** on the left-side navigation submenu. On the PG management tab, move the pointer over the Create icon and select **Database**.

3. In the Create Database dialog box, set relevant parameters.

   In the **Create Database** dialog box, set relevant parameters and click **Test connectivity**. If the system message indicating that the connectivity test is passed appears, the specified database is connected. Then, click **Complete**.

Parameter	Description	Remarks
Connect To	The type of the data store. Default value: Hologres.	The value is automatically generated and cannot be changed.

Parameter	Description	Remarks
Server	The endpoint of the Hologres instance.	You can view the endpoint on the Basic Information page of the Hologres instance in the Hologres console.
Port	The port number of the Hologres instance.	You can view the port number on the Basic Information page of the Hologres instance in the Hologres console.
Database Name	The name of the database to be bound to the current workspace.	The value must be the same as the database name in the CREATE DATABASE statement, for example, **testdb**.
User name	The AccessKey ID of your Apsara Stack tenant account.	You can click the username in the upper-right corner to view the AccessKey ID.
Password	The AccessKey secret of your Apsara Stack tenant account.	You can click the username in the upper-right corner to view the AccessKey secret.
JDBC Extension	The extension parameters used to establish a JDBC connection to Hologres.	Example: `?preferQueryMode=simple&tcpKeepAlive=true`
Test connectivity	Tests whether the database is connected.	If the system message indicating that the connectivity test is passed appears, the specified database is connected.

Delete a database

The PG management module of HoloStudio allows you to delete databases. On the left-side navigation submenu, click **PG management**. On the PG management tab, right-click the database you want to delete and select **Delete Database**.

In the Delete Database message, click **Ok** to delete the database.

> **Note** Only a superuser of a database or the database owner configured by a superuser can delete the database.

View database details

On the left-side navigation submenu, click **PG management**. On the PG management tab, right-click the database for which you want to view details and select **Database details**.

5.4.2. Manage tables

Similar to PostgreSQL, Hologres manages data by using tables. The PG management module of HoloStudio allows you to manage tables in a visualized manner. You can create, check, or delete a table with one click. This topic describes how to use the PG management module of HoloStudio to manage tables.

Create a table

1. Create a table.

   On the homepage of HoloStudio, click **Create Table**. Alternatively, click **PG management** on the left-side navigation submenu. On the PG management tab, move the pointer over the Create icon and select **Table**.

2. Edit the table content and attributes.

   On the tab that appears, edit the table content and attributes, and click **Commit**. The following figure shows an example of a column-oriented table with primary keys.

Section or tab	Parameter	Description
General	Interactive Analytics Database	The database where the table resides.
	Table Name	The name of the table.
	Description	The description of the table.
Field	Field Name	The name of the field in the table.
	Data Type	The data type of the field.
	Primary Key Field	Specifies whether to use the field as the primary key for the table.
	Optional	Specifies whether the field can be null.
	Array	Specifies whether the field is an ordered array of elements.
	Description	The description of the field.
	Actions	The actions that you can perform on the field. For example, you can delete the field from the table, or move up or down the position of the field in the table.
	Storage Mode	The storage mode of the table. Valid values: Row Store and Column Store. Default value: Column Store.

Section or tab	Parameter	Description
Properties	Lifecycle (Seconds)	The lifecycle of the table. Default value: Permanent.
	Clustered Index	The index used for sorting.
	Dictionary Code Columns	The column based on whose values a dictionary mapping is built.
	Bitmap Column	The column on which bit code is built.
Partitioned Table	PARTITION BY LIST	The partition field.

Check a table

1. View the DDL statement used to create the table.

   On the left-side navigation submenu, click **PG management**. Double-click the table you want to check and click **Generate DDL Statement** to check the SQL statement used to create the table.

2. Preview data.

   On the left-side navigation submenu, click **PG management**. Double-click the table you want to check and click **Data Preview** to check the content of the table. If the table contains no data, you can only view the fields of the table.

Delete a table

On the left-side navigation submenu, click **PG management**. On the PG management tab, right-click the table you want to delete and select **Delete Table**.

In the Delete message, click **Ok**.

5.4.3. Manage foreign tables

In Hologres, a foreign table does not store data but maps the table from the external data source. The PG management module of HoloStudio allows you to create, query, or delete foreign tables. You can only analyze foreign tables sourced from MaxCompute. This helps you obtain the query results.

This topic describes how to use the PG management module of HoloStudio to manage foreign tables.

Create a foreign table

On the left-side navigation submenu, click **PG management**. On the PG management tab, move the pointer over the Create icon and select **External Table**. On the tab that appears, set parameters for creating a foreign table and click **Commit**. An existing MaxCompute table is used in the following example. After you search for a MaxCompute table by entering its name, HoloStudio automatically generates a foreign table based on the fields of the MaxCompute table after you click Commit.

> **Note**
>
> 1. Before you create a foreign table in Hologres, make sure that its source table exists in a MaxCompute project.
> 2. The fields of a foreign table in Hologres have a one-to-one mapping with those of the source table in MaxCompute. You can query specific fields or all fields.

Section or icon	Parameter	Description
General	Interactive Analytics Database	The database where the foreign table to be created resides.
	Table Name	The name of the foreign table.
External Service	Types	The service type of the external table. You can only set this parameter to MaxCompute.
Table	Table	The source table in MaxCompute to be mapped.
Commit	Commit	You can click this button to commit the foreign table that you create.

Check a foreign table

1. **Preview data.**

   On the left-side navigation submenu, click **PG management**. On the PG management tab, double-click the foreign table you want to check, and click **Data Preview** to check the content of the foreign table.

2. **View the DDL statement used to create the table.**

   On the left-side navigation submenu, click **PG management**. On the PG management tab, double-click the foreign table you want to check, and click **Generate DDL Statement** to check the SQL statement used to create the foreign table.

Delete a foreign table

On the left-side navigation submenu, click **PG management**. On the PG management tab, right-click the foreign table you want to delete and select **Delete Table**. In the Delete message, click **Ok** to delete the foreign table.

5.5. Data analytics

5.5.1. Overview

The Data Analytics module of HoloStudio is seamlessly integrated with DataWorks for node scheduling and provides all-in-one, stable, and efficient extract, transform, load (ETL) services. It can also synchronize MaxCompute table schemas and data and allows you to upload local files for data analytics.

The Data Analytics module consists of the following submodules:

1. Folder: stores data analytics nodes, helping you manage data analytics nodes of each database.
2. Interactive Analytics Development: is integrated with DataWorks to schedule ETL nodes.
3. One-click MaxCompute table structure synchronization: allows you to create multiple foreign tables sourced from MaxCompute at a time.
4. One-click MaxCompute data synchronization: provides a visualized user interface for you to synchronize MaxCompute data to Hologres.
5. One-click local file Upload: allows you to upload local files to Hologres.

Folder

Folders store data analytics nodes, helping you manage data analytics nodes of each database.

On the left-side navigation submenu, click **Data Analytics**. On the Data Analytics tab, move the pointer over the Create icon and select **Folder**. In the Create Folder dialog box, enter a folder name and click **Commit**.

5.5.2. Use the Interactive Analytics Development submodule

The Interactive Analytics Development submodule is seamlessly integrated with DataWorks. You can use HoloStudio to import data from MaxCompute to Hologres. You can also use DataWorks to schedule nodes to periodically import data to Hologres. This topic describes how to use HoloStudio to map the source data stored in a MaxCompute table to Hologres for periodic scheduling.

1. **Prepare a MaxCompute table.**

   Create a table in MaxCompute and import data to the table. You can also select a table with data from Data Map. In this example, an existing table in Data Map is used. The following Data Definition Language (DDL) statement is used to create the table:

```
CREATE TABLE IF NOT EXISTS bank_data_odps
(
 age             BIGINT COMMENT 'age',
 job             STRING COMMENT 'job type',
 marital         STRING COMMENT 'marital status',
 education       STRING COMMENT 'education level',
 card         STRING COMMENT 'credit card available or not',
 housing         STRING COMMENT 'mortgage',
 loan            STRING COMMENT 'loan',
 contact         STRING COMMENT 'contact',
 month           STRING COMMENT 'month',
 day_of_week     STRING COMMENT 'day in a week',
 duration        STRING COMMENT 'duration',
 campaign        BIGINT COMMENT 'number of contacts during the campaign',
 pdays           DOUBLE COMMENT 'interval from the last contact',
 previous        DOUBLE COMMENT 'number of contacts with the customer',
 poutcome        STRING COMMENT 'result of the previous marketing campaign',
 emp_var_rate    DOUBLE COMMENT 'employment change rate',
 cons_price_idx  DOUBLE COMMENT 'consumer price index',
 cons_conf_idx   DOUBLE COMMENT 'consumer confidence index',
 euribor3m       DOUBLE COMMENT 'euro deposit rate',
 nr_employed     DOUBLE COMMENT 'number of employees',
 y               BIGINT COMMENT 'fixed time deposit available or not'
);
```

2. **Create a foreign table.**

   Go to the HoloStudio page. On the left-side navigation submenu, click **PG management** or **SQL Console**. On the tab that appears, create a foreign table for mapping data in the MaxCompute source table. In this example, use the following SQL statements to create a foreign table:

```
BEGIN;
CREATE FOREIGN TABLE if not EXISTS bank_data_foreign_holo (
 age int8,
 job text,
 marital text,
 education text,
 card text,
 housing text,
 loan text,
 contact text,
 month text,
 day_of_week text,
 duration text,
 campaign int8,
 pdays float8,
 previous float8,
 poutcome text,
 emp_var_rate float8,
 cons_price_idx float8,
 cons_conf_idx float8,
 euribor3m float8,
 nr_employed float8,
 y int8
)
SERVER odps_server
OPTIONS (project_name 'projectname', table_name 'bank_data_odps');
GRANT SELECT ON bank_data_foreign_holo TO PUBLIC;
COMMIT;
```

Note The OPTIONS parameter contains two fields: project_name, which is the name of the MaxCompute project, and table_name, which is the name of the MaxCompute table.

3. **Create a data storage table.**

   Create a table in HoloStudio to receive and store data. The fields in this table must be of the same data types as those in the foreign table. In this example, use the following SQL statements to create the storage table:

```
BEGIN;
CREATE TABLE if not EXISTS bank_data_holo (
 age int8,
 job text,
 marital text,
 education text,
 card text,
 housing text,
 loan text,
 contact text,
 month text,
 day_of_week text,
 duration text,
 campaign int8,
 pdays float8,
 previous float8,
 poutcome text,
 emp_var_rate float8,
 cons_price_idx float8,
 cons_conf_idx float8,
 euribor3m float8,
 nr_employed float8,
 y int8,
 ds text NOT NULL
)
PARTITION  BY LIST(ds);
CALL SET_TABLE_PROPERTY('bank_data_holo', 'orientation', 'column');
CALL SET_TABLE_PROPERTY('bank_data_holo', 'time_to_live_in_seconds', '700000');
COMMIT;
```

4. **Create a partitioned table.**

   On the HoloStudio page, click **Data Analytics** on the left-side navigation submenu. On the Data Analytics tab, move the pointer over the Create icon and select **Interactive Analytics Development** to create a Hologres development node. Then, go to the SQL editor of the node and enter SQL statements to create a partitioned table for obtaining the required data. After you enter the SQL statements, click the **Run** icon. In the Field dialog box, set a value for the ${bizdate} parameter. After the SQL statements are executed, click the **Save** icon and then **Go to DataStudio for Scheduling** to schedule the node. You can enter the following sample SQL statements:

```
create table if not exists bank_data_holo_1_${bizdate} partition of bank_data_holo
  for values in ('${bizdate}');
insert into bank_data_holo_1_${bizdate}
select
    age as age,
    job as job,
    marital as marital,
    education as education,
    card as card,
     housing as housing,
    loan as loan,
    contact as contact,
    month as month,
    day_of_week as day_of_week,
     duration as duration,
    campaign as campaign,
     pdays as pdays,
    previous as previous,
    poutcome as poutcome,
     emp_var_rate as emp_var_rate,
    cons_price_idx as cons_price_idx,
    cons_conf_idx as cons_conf_idx,
    euribor3m as euribor3m,
    nr_employed as nr_employed,
    y as y,
    '${bizdate}' as ds
from bank_data_foreign_holo;
```

5. **Schedule the partitioned table.**

   Go to the DataStudio page and create a Hologres development node. In the SQL editor of the node, enter SQL statements to synchronize the partitioned table information to the node and click **Update Code**. Before you create the node, make sure that a workflow is created.

6. **Set parameters for scheduling the data analytics node.**

   On the editing tab of the Hologres development node, click the **Properties** tab in the right-side navigation pane to set parameters for scheduling the node.

   i. **Set parameters in the General section.**

      In the **Arguments** field, specify a value for the ${bizdate} variable.

   ii. **Set parameters in the Schedule section.**

      Select Normal for **Execution Mode** and set other parameters as required.

   iii. **Set parameters in the Dependencies section.**

      Select **Yes** for **Auto Parse** and click **Use Root Node**. After DataStudio automatically parses and displays the root node as a parent node, change the value of Auto Parse to **No**. You can also select a table that is scheduled as a parent node.

7. **Save and deploy the node for scheduling.**

   After you set the scheduling parameters for the node, click the **Save** icon and then the **Submit** icon. After that, click **Deploy** in the upper-right corner.

8. **Deploy the node in Operation Center.**

On the **Create Package** page, find the target node and click Publish in the Actions column. After the node is deployed, click **Operation Center** in the top navigation bar to generate retroactive data for the node.

In **Operation Center**, right-click the published node and choose **Run > Current Node Retroactively**. Configure the node based on your business requirements.

9. **Check the content of the table in HoloStudio.**

   After the retroactive data generation node is run, go back to HoloStudio. On the left-side navigation submenu, click **PG management**. On the PG management tab, click a database and choose Mode > public > **Table**. Double-click the partitioned table that is scheduled and click **Data Preview** to check whether the data is imported to the table.

5.5.3. Create multiple foreign tables at a time

Seamlessly integrated with MaxCompute at the underlying layer, Hologres allows you to create foreign tables to query MaxCompute data in an accelerated manner. You can create multiple foreign tables at a time by using the IMPORT FOREIGN TABLE statement. To free you from SQL operations, HoloStudio provides the following submodule for you to create foreign tables in a visualized manner: One-click MaxCompute table structure synchronization.

1. **Create a schema sync node.**

   On the HoloStudio page, click **Data Analytics** on the left-side navigation submenu. On the Data Analytics tab, move the pointer over the Create icon and select **One-click MaxCompute table structure synchronization**. In the Create Node dialog box, set relevant parameters and click Commit. The schema sync node is created.

2. **Set parameters for synchronizing the table schema.**

   After the schema sync node is created, you must set parameters for synchronizing the table schema based on your needs.

Parameter	Description	Remarks
Target Library	The name of the Hologres database where the foreign tables are to be created.	N/A
Target Schema	The name of the schema in the specified Hologres database.	The default value is public. If you have created a schema, you can select the created schema.
Remote Service type	The type of the external service. You can create only foreign tables sourced from MaxCompute.	The default value is odps.
Remote server	The external server. The default value is odps_server.	After you purchase a Hologres instance, the system automatically creates a server named odps_server. You can directly use it.

Parameter	Description	Remarks
Remote library	The name of the MaxCompute project where the tables mapping the foreign tables to be created reside.	N/A
Table name rules	The regular expression for specifying the tables whose schema is to be synchronized. By default, the schema of all tables in the specified MaxCompute project will be synchronized.	◦ If a foreign table to be created is named the same as an existing foreign table in Hologres, the foreign table is not created. ◦ If a MaxCompute table whose schema is to be synchronized contains data types that Hologres does not support, an error is thrown. In this case, exclude this MaxCompute table in the regular expression. ◦ For more information, see IMPORT FOREIGN SCHEMA.
Regular preview	The execution result of the regular expression.	N/A

3. **Run the schema sync node.**

   Click the **Save** icon and then click the **Run** icon to run the schema sync node. After the schema sync node is run, click PG management on the left-side navigation submenu. The created foreign tables appear. You can query the table data.

5.5.4. Import MaxCompute data

To improve the efficiency of querying MaxCompute data, Hologres allows you to import MaxCompute data to Hologres for queries. HoloStudio provides the following submodule for you to directly import MaxCompute data in a visualized manner: One-click MaxCompute data synchronization.

1. **Create a data sync node.**

   On the HoloStudio page, click Data Analytics on the left-side navigation submenu. On the Data Analytics tab, move the pointer over the Create icon and select **One-click MaxCompute data synchronization**. In the Create Node dialog box, enter the node information and click Commit. The data sync node is created.

2. **Set parameters for synchronizing data.**

   After the data sync node is created, you must set parameters for synchronizing data.

Section	Parameter	Description	Remarks

Section	Parameter	Description	Remarks
MaxCompute Source table selection	External table source	The source of the foreign table. Valid values: External table already exists and New external table.	◦ If you select External table already exists, the existing foreign table mapping the MaxCompute table will be used. ◦ If you select New external table, you must create a foreign table mapping the MaxCompute table.
	External table table name	The name of the existing foreign table.	The foreign table must map the MaxCompute table whose data will be synchronized.
Target table settings	Target Library	The name of the Hologres database to which the MaxCompute data will be synchronized.	N/A
	Target schema	The name of the schema in the specified Hologres database.	The default value is public. If you have created a schema, you can select the created schema.
	Destination Table Name	The name of the target table to which the MaxCompute data will be synchronized.	The table name can be customized.
	Target table description	The description of the target table.	N/A
Synchronization settings	Synchronization field	The fields to be synchronized from the specified MaxCompute table.	You can select specific or all fields in the MaxCompute table.
	Partition configuration	The partition fields to be synchronized.	Hologres supports a maximum of one level of partitions.
	Index configuration	The index to be built for the target table.	N/A

Section	Parameter	Description	Remarks
SQL Script	SQL Script	The SQL statements that are executed when the data sync node is run.	N/A

3. **Run the data sync node.**

   Click the **Save** icon and then click the **Run** icon to run the data sync node. After the node is run, you can query the imported data in SQL Console or PG management.

5.5.5. Upload local files

This topic describes how to upload local files in HoloStudio in a visualized manner.

Hologres allows you to use the COPY statement to import data from the standard input of a client to a specified table. For more information, see COPY. HoloStudio allows you to import data in a local file to a specified table by uploading the local file in a visualized manner. To upload a local file in HoloStudio, perform the following steps:

1. **Create a table.**

   In SQL Console or PG management, create a table to which data in the local file will be imported. In this example, use the following SQL statements to create a table:

```
BEGIN;
CREATE TABLE if not EXISTS holo_bank (
 age int8,
 job text,
 marital text,
 education text,
 card text,
 housing text,
 loan text,
 contact text,
 month text,
 day_of_week text,
 duration text,
 campaign int8,
 pdays float8,
 previous float8,
 poutcome text,
 emp_var_rate float8,
 cons_price_idx float8,
 cons_conf_idx float8,
 euribor3m float8,
 nr_employed float8,
 y int8
);
COMMIT;
```

2. **Create a node for uploading the local file.**

Go to the HoloStudio page. On the left-side navigation submenu, click Data Analytics. On the Data Analytics tab, move the pointer over the Create icon and select **Upload files locally with one click**.

3. **Enter the node information.**

   In the **One-click local file Upload** dialog box, set the parameters based on your business needs and click Next Step.

Parameter	Description	Remarks
Target Library	The name of the Hologres database where the target table resides.	N/A
Target Schema	The name of the schema where the target table resides.	The default value is public. If you have created a schema, you can select the created schema.
Select the data table to import	The name of the target table to which data in the local file will be imported.	N/A

4. **Select the local file to upload and set other required parameters.**

   After you click Next Step, select the local file to upload, set other required parameters, and then click Commit.

Parameter	Description	Remarks
Select File	The local file to upload.	You can select a local file only in the .txt, .csv, or .log format.
Select separator	The delimiter of fields in the file. Select Comma (,) or Space ( ).	N/A
Original character set	The character set of the file.	◦ GBK ◦ UTF-8 ◦ CP936 ◦ ISO-8859
First behavior title	Specifies whether to use the first line as the header line.	N/A

5. **View the imported data.**

   After you click **Commit**, data in the selected local file is imported to the specified table. You can go to SQL Console or PG management to view the imported data.

5.6. Hologres console

5.6.1. Overview

Hologres is a real-time interactive analytics service that is fully compatible with PostgreSQL and seamlessly integrated with the big data ecosystem. Hologres delivers high-concurrency and low-latency performance in analyzing terabyte-scale data. Hologres allows you to use mainstream Business Intelligence (BI) tools to get an analytical insight into data from multiple dimensions and explore business data in an efficient and cost-effective manner.

For convenience of business, Apsara Stack provides the Hologres console independent from the DataWorks console for different users to managing Hologres instances, users, and databases.

Log on to the Apsara Stack console. In the top navigation bar, choose **Products > Interactive Analytics** to go to the Hologres console. The following figure shows the Overview page in the Hologres console.

5.6.2. View the instance list

The Instances page lists all Hologres instances purchased by your Apsara Stack tenant account. On this page, you can view the instance status, change instance configurations, and create instances. You can also click an instance name to go to the instance details page where you can manage objects in the instance, including databases and users.

Instances

Log on to the Apsara Stack console. In the top navigation bar, choose **Products > Interactive Analytics** to go to the Hologres console. In the Hologres console, click **Instance List** in the left-side navigation pane. The following figure shows the Instance List page.

1. **New engine instance button**

   On the Instance List page, click **New engine instance**. In the New engine instance dialog box, enter an instance name and select the instance specifications to create a Hologres instance.

2. **Search box**

   If you have purchased multiple Hologres instances, you can enter a keyword of an instance name in the search box to find the target instance.

3. **Running status column**

   The Running status column displays the running status of each Hologres instance. An instance can be in one of the following states:

   - Normal operation: The instance is running as expected.
   - Creating: The payment is successful, and Hologres is creating the instance. You must wait for 3 to 5 minutes.
   - Shutdown: The instance has been suspended and you cannot connect to it.

Operation column

The Operation column provides the following buttons for you to manage a Hologres instance:

1. **Management**

   Find the target instance and click **Management** in the Operation column. On the page that appears, you can view and manage objects in the instance, including databases and users.

2. **Change configuration**

   If your instance cannot meet your business needs or your instance has a large amount of surplus resources, you can click **Change configuration** in the Operation column. In the **Change configuration** dialog box, upgrade or downgrade your instance configurations based on your business needs.

3. **Shutdown**

   Find the target instance and click **Shutdown** in the **Operation** column to suspend the instance. You cannot connect to the suspended instance.

5.6.3. Manage instances

This topic describes how to view and change instance configurations, select a network type, and select a connection method on the Basic information page in the Hologres console.

View and change instance configurations

In the Hologres console, click **Instance List** in the left-side navigation pane. In the instance list, find the target instance and click **Management** in the **Operation** column. The Basic information page displays basic information about a Hologres instance, including the instance name, instance ID, region, instance version, billing method, instance specification, and creation time.

If you need to change the instance specifications, click **Change configuration**. In the Change configuration dialog box, upgrade or downgrade the instance specifications based on your business needs.

Select a network type

The following table lists the supported network type.

Network type	Domain name	Scenario
Internal network	`<instancename>-cn-<region>-internal.hologres.aliyuncs.com:80`	Select this network type when you want to connect to the Hologres instance by using the classic network, without charges on the Internet traffic.

Select a connection method

Hologres is compatible with PostgreSQL. You can connect to a Hologres instance from the PostgreSQL client or over JDBC interfaces by using ETL or BI tools.

The Connection Methods section offers methods for you to use common development tools to connect to a Hologres instance. You can select a development tool and connection method based on your business needs and preference.

1. **Connect from the PostgreSQL client**

   To connect to a Hologres instance from the PostgreSQL client, use the following connection string:

   ```
   PGUSER=<AccessId> PGPASSWORD=<AccessKey> psql -p <Port> -h <Endpoint> -d <Database>
   ```

2. **Connect over JDBC**

   To connect to a Hologres instance over JDBC, use the following connection string:

```
postgres://<AccessId>:<AccessKey>@<Endpoint>:<Port>/<database>? preferQueryMode=simple&
tcpKeepAlive=true
```

5.6.4. Manage users

This topic describes how to manage users on the User Management page in the Hologres console.

Overview

In the Hologres console, click **Instance List** in the left-side navigation pane. In the instance list, find the target instance and click **Management** in the **Operation** column. On the page that appears, click **User Management**. On the User Management page, you can manage users on a Hologres instance without executing cumbersome SQL statements. For example, you can add and delete users and grant permissions to users on this page.

After you create a Hologres instance with your Apsara Stack tenant account, this account becomes a superuser of the instance. A superuser has all permissions on the Hologres instance. By default, the User Management page displays only the information of the Apsara Stack tenant account that creates the Hologres instance. The information of a Resource Access Management (RAM) user appears on this page only after you use the Apsara Stack tenant account to add it to the instance.

Column	Description	Remarks
Members	Displays the usernames of the Apsara Stack tenant account and RAM users on the Hologres instance.	Generally, a username appears in the xxx format.
Cloud account	Displays the account IDs of users on the Hologres instance.	N/A
Type	Displays the roles assigned to users on the Hologres instance.	The user can be a superuser or normal user.

Add a user

On the User Management page, you can create RAM users on a Hologres instance without executing the SQL CREATE statement.

Click **Add new user**. In the Add new user dialog box, select existing RAM users under your Apsara Stack tenant account to add them to the Hologres instance. If no RAM user exists under your Apsara Stack tenant account, create a RAM user first.

When you add a RAM user, you can assign the superuser or normal user role to the user.

- Superuser: A superuser has all permissions on the Hologres instance without the need for additional authorization.
- Normal user: A normal user cannot view or manage any objects on the Hologres instance, including databases, schemas, and tables. A normal user must be authorized before it can view and manage objects in the instance. We recommended that you go to the DB management page to grant permissions to RAM users as required. Alternatively, you can use SQL statements to grant permissions

to RAM users.

Delete a user

Find the target user on the User Management page and click **Delete** in the Operation column to delete the user from the Hologres instance. A deleted user has no access to the Hologres instance.

5.6.5. Manage databases

This topic describes how to manage databases on the DB management page in the Hologres console.

Overview

In the Hologres console, click **Instance List** in the left-side navigation pane. In the instance list, find the target instance and click **Management** in the **Operation** column. On the page that appears, click **DB management**. On the DB management page, you can manage all databases on the current Hologres instance. You can create databases, select a permission management mode for the databases, and view database information.

> **Note** A default database named postgres is automatically created after you create a Hologres instance. This database is provided for management purposes only and does not appear on the DB management page. This database is allocated with limited resources. Create databases on this page based on your business needs.

Create a database

Hologres allows you to create a database with one click on the graphical user interface (GUI), eliminating the need for SQL operations.

Click **New Database**. In the New Database dialog box, enter a name for the database and set the Simple permissions model parameter to Open or Close. To simplify authorization, we recommend that you set the Simple permissions model parameter to Open.

Hologres provides two permission models for you to authorize users in a convenient way.

- Standard PostgreSQL authorization: Compatible with PostgreSQL, Hologres provides a permission model that is exactly the same as the standard PostgreSQL authorization model. You can authorize RAM users by using the standard PostgreSQL GRANT statement.
- SPM: Backed by the understanding of customers' business and its practical experience, Alibaba Cloud introduced a simple permission model (SPM) to Hologres to simplify the management of user permissions. The SPM is a coarse-grained model that authorizes users by user group.

After a database is created, you can use a development tool to connect to the database to analyze data.

Authorize a user

After the SPM is enabled for a new database, you can authorize RAM users with one click in the Hologres console. Perform the following steps:

1. **Open the Permission management right-side pane.**

   Find the target RAM user and click **User authorization** in the Operation column. You can grant permissions to a RAM user by adding the user to the desired user group.

2. **Add a RAM user to a user group.**

In the Permission management right-side pane, click **Add authorization**. In the Add authorization dialog box, select the account to which you want to grant permissions, select the desired user group below Permissions policy, and then click **OK**.

Revoke permissions

If the SPM is enabled for your database, you can revoke the permissions of a RAM user with one click in the Hologres console.

On the instance details page, click **DB management**. On the DB management page, find the target database and click **User authorization** in the Operation column. In the Permission management right-side pane, find the target RAM user and click **Delete authorization** in the Operation column.

Delete a database

On the DB management page, find the database no longer required and click Delete in the Operation column to delete the database. After a database is deleted, data in the database is also deleted and cannot be recovered.

6.Realtime Analysis

6.1. Overview

Integrated with DataWorks, DataAnalysis supports creating MaxCompute tables in tabular mode, collaboratively editing workbooks and performing statistical analysis, and generating and sharing visual reports. These features enable data developers and business staff to quickly analyze data.

Go to DataAnalysis

1. Log on to the DataWorks console.
2. Click the DataWorks icon in the upper-left corner and choose **All Products > DataAnalysis**.
3. On the **Home** page of DataAnalysis, click **Experience now** to go to the Web Excel page.

Features

- **Workbook**

  You can create and edit workbooks. Workbooks support basic operations such as addition, subtraction, multiplication, and division, and multiple data processing methods, including functions, classification, and aggregation. In addition, you can edit workbooks collaboratively with other users online and create pivot tables for further analysis.

- **Visual report**

  You can create and design visual reports by dragging, dropping, and configuring controls without running SQL statements.

- **Dimension table**

  You can create MaxCompute tables in tabular mode by one click without running SQL statements and edit MaxCompute tables collaboratively with other users online.

  - You can create a MaxCompute table in tabular mode.
  - You can import data into a MaxCompute table by one click.

6.2. SQL queries

DataWorks allows you to import data from a data source and perform queries and analysis of data by using the SQL query feature. This topic describes how to use the SQL query feature.

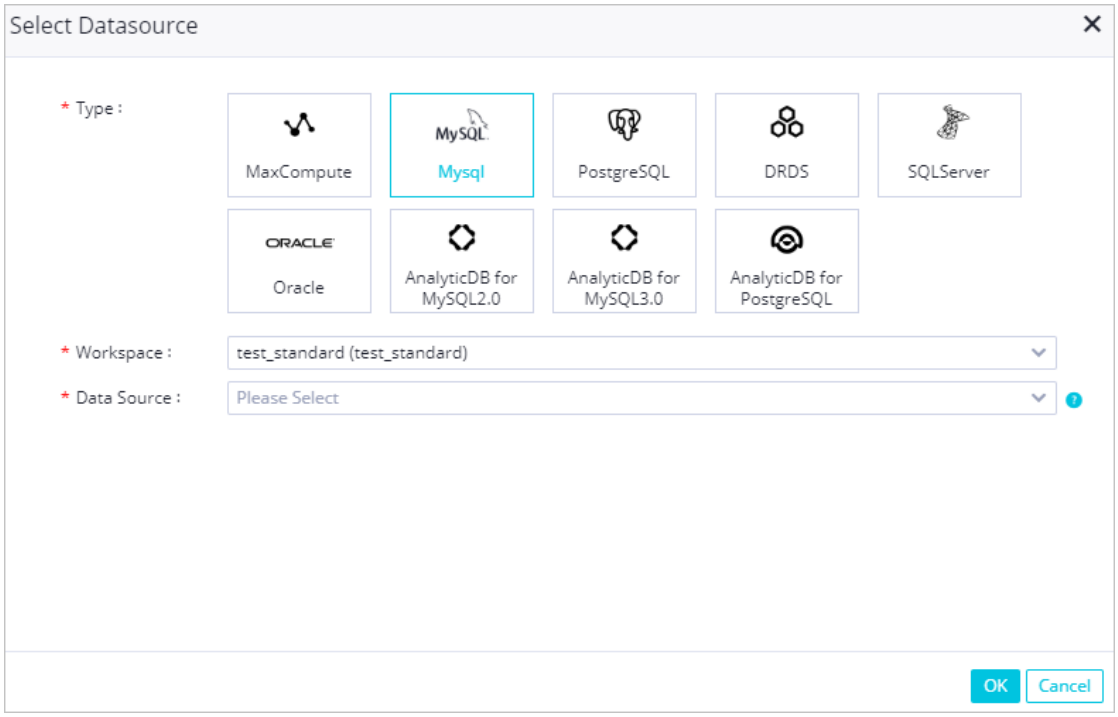
Create an SQL query task

1. Go to the DataAnalysis page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click the ☰ icon and choose **All Products > Data Development > DataAnalysis**. The **DataAnalysis** page appears.
2. On the **DataAnalysis** page, click **SQL Query** in the top navigation bar to go to the **SQL Query** page.
3. Specify the data source to be queried.

i. On the left side of the **SQL Query** page, click the + icon in the **Data source** section.

The first time you go to the **SQL Query** page, you must click **Add Now** first.

ii. In the **Select Datasource** dialog box, set the parameters as required.

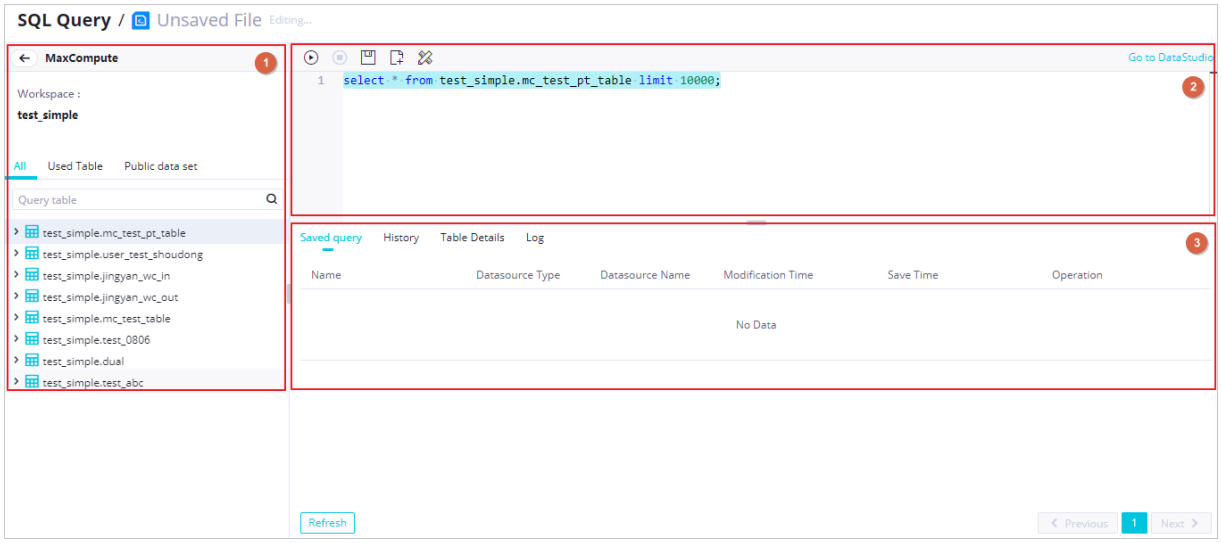


Parameter	Description
Type	Specify the type of the data source to be queried.
Workspace	The workspace in which the specified data source resides.
Data Source	Select the data source to be queried from the drop-down list. **Note** If you set the **Type** parameter to MaxCompute, this parameter is not displayed.

iii. Click **OK** to create an SQL query task. On the SQL Query page, you can use SQL statements to query and analyze data for the created SQL query task.

Perform and manage SQL queries

On the SQL Query page, you can use SQL statements to query and analyze data from the specified data source. You can also view saved queries, execution history, and logs of the SQL queries.



Area No.	Description
1	This section displays the tables that are contained in the specified data source for which you perform the SQL queries in the current workspace. You can also search for a table by entering keywords in the search box.
2	This section allows you to run SQL statements to query data. After you enter the SQL statement to be executed, click the icon in the toolbar to run the statement. You can also click **Go to DataStudio** to copy the SQL statement and go to the DataStudio page. On the DataStudio page, you can develop data and schedule ETL tasks.
3	In this section, you can view the details of the saved queries, execution history, and logs of SQL queries. You can also preview, load, rename, and delete historical SQL queries on the Saved query tab.

6.3. Workbook

6.3.1. Create a workbook

This topic describes how to create a workbook. After a workbook is created, you can rename, clone, delete, or change the owner of the workbook.

Procedure

1. Log on to the DataWorks console.
2. On the DataStudio page, click in the upper-left corner and choose **All Products >**

DataAnalysis.

3. On the DataAnalysis homepage, click **Experience Now**.
4. On the **Web Excel** page, click + in the **New Spreadsheet** section.

> **Note** If you have created workbooks, you can search for a workbook by entering its name in the search box in the **All Spreadsheets** section. Then, click the workbook name in the **File Name** column to go to the workbook editing page.

5. In the **New spreadsheet** dialog box, enter a name in the **File Name** field.
6. Click **OK**.

Result

After the workbook is created, it appears in the **All Spreadsheets** section. In this section, you can view all created workbooks. In addition, you can rename, clone, delete, or change the owner of a workbook.

- Find the target workbook and click **Rename** in the Operation column. In the **Rename** dialog box, enter the new name in the **File Name** field and click **OK**.
- Find the target workbook and click **Change Owner** in the Operation column. In the **Change Owner** dialog box, select an owner from the New Owner drop-down list and click **OK**.
- Find the target workbook and click **Clone** in the Operation column. The cloned workbook appears in the workbook list. The name of the cloned workbook contains the _copy suffix.
- Find the target workbook and click **Delete** in the Operation column. In the **Delete** message, click **OK**.

6.3.2. Edit a workbook

This topic describes how to edit a workbook. For example, you can import data to, export data from, and share a workbook, create a pivot table in a workbook, and use the data profiling feature.

Go to the workbook editing page

1. Log on to the DataWorks console.
2. On the DataStudio page, click ≡ in the upper-left corner and choose **All Products** > **DataAnalysis**.
3. On the DataAnalysis homepage, click **Experience Now**. The **Web Excel** page appears.
4. In the **All Spreadsheets** section of the **Web Excel** page, click the name of the target workbook in the **File Name** column.

   After you create a workbook, the workbook editing page appears.

Apply a template to a workbook or save a workbook as a template

You can apply an existing template to the current workbook by performing the following steps:

1. In the upper-right corner of the workbook editing page, choose **Template** > **Import Template**.
2. In the **Import Template** dialog box, select a file to be used as a template for the current workbook.

> **Note** The data of the selected template will overwrite that of the current workbook.

3. Click **OK**.

You can save the current workbook as a template by performing the following steps:

1. In the upper-right corner of the workbook editing page, choose **Template** > **Save as Template**.
2. In the **Template settings** dialog box, set the **Type**, **Name**, and **Description** parameters.

> **Notice** The template name can be up to 256 characters in length and the template description can be up to 1,024 characters in length.

3. Click **OK**.

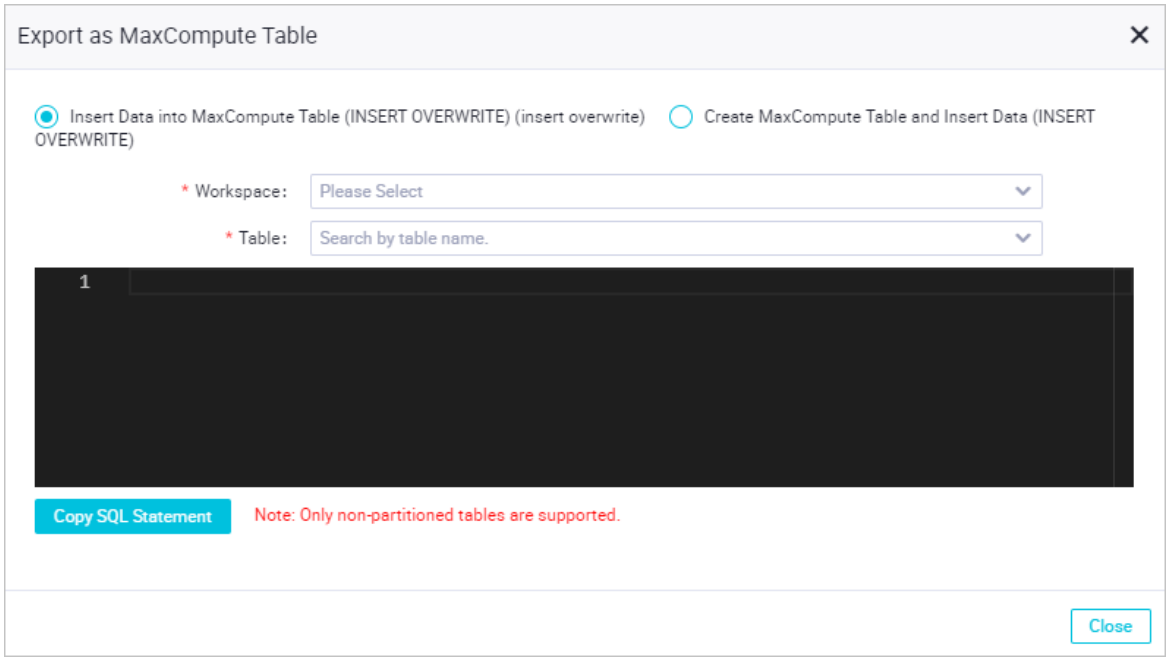
Import data to a workbook

1. In the upper-right corner of the workbook editing page, click **Import**.
2. In the Open dialog box, find and select a local file to be imported and click **Open**. The data in the local file is imported to the current workbook.

> **Note** You can only import data from Excel files.

Export data from a workbook to a MaxCompute table

1. In the upper-right corner of the workbook editing page, choose **Export** > **Generate MaxCompute Build Table Statement**.
2. In the **Export as MaxCompute Table** dialog box, set relevant parameters.



Insert mode	Parameter	Description
Insert Data into MaxCompute Table (INSERT	**Workspace**	The workspace to which the MaxCompute table belongs.

OVERWRITE) (Insert overwrite)	Parameter	Description
	Table	The MaxCompute table to which you want to insert data.
Create MaxCompute Table and Insert Data (INSERT OVERWRITE)	Workspace	The workspace to which the MaxCompute table belongs.
	Table Name	The name of the MaxCompute table. Make sure that the table name has not been used. You can click **Check Duplicate Names** to check whether the table name exists.

3. After the parameters are set, click **Copy SQL Statement**.

> **Notice** Only non-partitioned tables are supported.

Create a pivot table

1. On the workbook editing page, select the data for which you want to create a pivot table and click **Pivot** in the upper-right corner.
2. In the **Create Pivot Table** dialog box, set relevant parameters.

   Specify the range of the data to be analyzed. You can set the **Choose Data** parameter to **Select Range** or **Use External Data Store** as needed.

   - If you select **Select Range**, select the cells in the workbook for which you want to create a pivot table. The value of the **Range** field changes based on the selected cells.

     > **Note** This parameter is available only when you select **Select Range**.

   - If you select **Use External Data Store**, set the **Type** parameter first. You can set the Type parameter to **Mysql** or **Data Services**.

- If you select **Mysql**, set the parameters described in the following table.

Parameter	Description
Choose Data	The range of the data to be analyzed. Select **Use External Data Store**.
Type	The type of the data source. Select **Mysql**.
Workspace	The workspace where the MySQL data store resides.
Data Store	The name of the connection to the data store. To create a connection to a data store, perform the following steps: Click the **Workspace Manage** icon in the upper-right corner. On the page that appears, click **Data Source** in the left-side navigation pane. On the Data Source page, create a connection to a data store.
Table	The table for which you want to create a pivot table.

- If you select **Data Services**, set the parameters described in the following table.

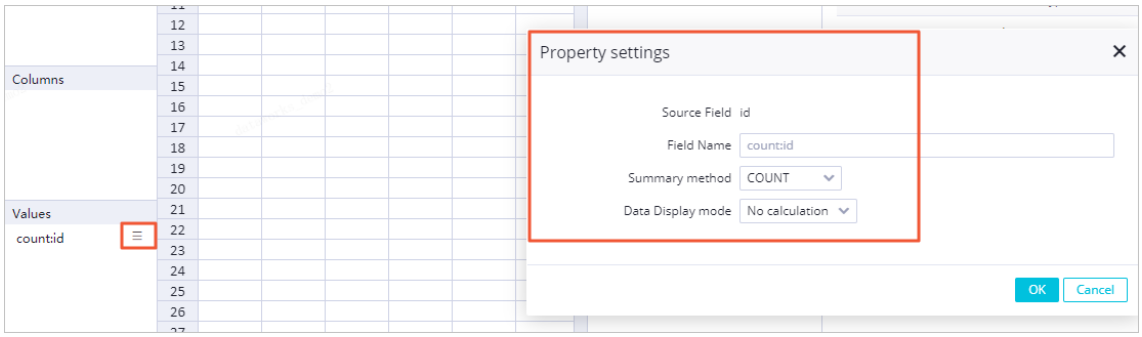
Parameter	Description
Choose Data	The range of the data to be analyzed. Select **Use External Data Store**.
Type	The type of the data source. Select **Data Services**.
Workspace	The workspace where the API of DataService Studio resides.
API Group	The API group to which the API belongs.
API	The API to be used as the data source.

3. Click **OK**. The pivot table editing page appears.

   This topic takes the pivot table for a selected range of data as an example.

   - **Data Source**: the range that you specified in the previous step.
   - **Pivot Table Fields**: the names of the fields that you selected in the previous step.
   - **Rows**: Drag fields from the Pivot Table Fields section to the **Rows** section. Each value of the field added to the **Rows** section occupies a row in the pivot table.
   - **Columns**: Drag fields from the Pivot Table Fields section to the **Columns** section. Each value of the field added to the **Columns** section occupies a column in the pivot table.
   - **Values**: Click the property setting icon for a field in the **Values** section. In the Property settings dialog box, set the **Summary method** and **Data Display mode** parameters. By default, the

Field Name parameter cannot be modified.



Parameter	Description
Source Field	The name of the selected source field.
Field Name	The name of the field that appears in the pivot table. The name is in the format of **Aggregation method:Source field name**.
Summary method	The aggregation method. Valid values: **SUM**, **COUNT**, **MAX**, **MIN**, and **AVG**.
Data Display mode	The mode for displaying the data. Valid values: **No calculation** and **Percentage of Total**.

- **Filters**: Drag fields from the Pivot Table Fields section to the Filters section. In the right-side pivot table display area, you can select the fields to filter data.

Download a workbook

In the upper-right corner of the workbook editing page, click **Download** to download the workbook to a local directory.

Share a workbook

In the upper-right corner of the workbook editing page, click **Share**. In the dialog box that appears, set the sharing mode.

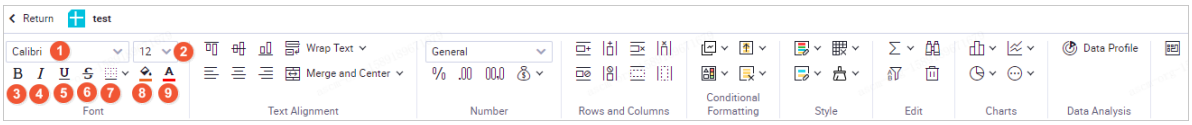
You can share the workbook in the following ways:

- **Link**: Click **Copy Link** and send the copied URL to other users as needed.
- **Users with Edit Access**: Click **Add** in the **Users with Edit Access** section. In the dialog box that appears, select the users to whom you want to grant the edit permission and click **OK**.
- **Visible to All**: To allow all users to view the workbook, turn on the **Visible to All** switch.
- **Users with Read Access**: To allow only specific users to view the workbook, turn off the **Visible to All** switch and click **Add** in the **Users with Read Access** section. In the dialog box that appears, select the users to whom you want to grant the read-only permission and click **OK**.

> **Note** If the system notifies you that the number of users to whom you want to grant the read-only permission reaches the upper limit, you can upgrade the DataWorks edition.

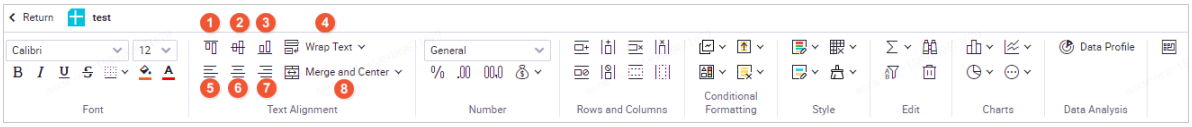
Menu bar

- Font



No.	Feature	Description
①	**Font**	Select a font type from the drop-down list as needed.
②	**Font Size**	Select a font size from the drop-down list as needed.
③	**Bold**	Set text in bold.
④	**Italic**	Set text in italic.
⑤	**Underline**	Underline text.
⑥	**Strikethrough**	Add a strikethrough to text.
⑦	**Borders**	Add borders to cells.
⑧	**Background Color**	Specify the background color of cells.
⑨	**Text Color**	Change the text color.

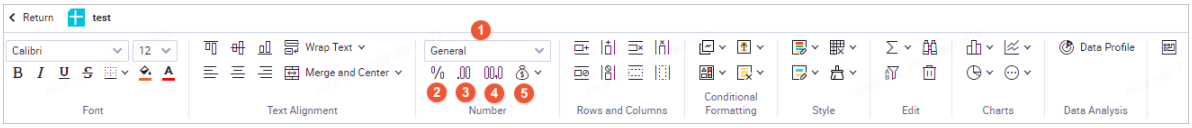
- Text Alignment



No.	Feature	Description
①	**Top Align**	Align text vertically to the top.
②	**Middle Align**	Align text vertically to the center.
③	**Bottom Align**	Align text vertically to the bottom.
④	**Wrap Text**	Display long text in multiple lines in a cell.
⑤	**Align Left**	Align text horizontally to the left.
⑥	**Center**	Align text horizontally to the center.

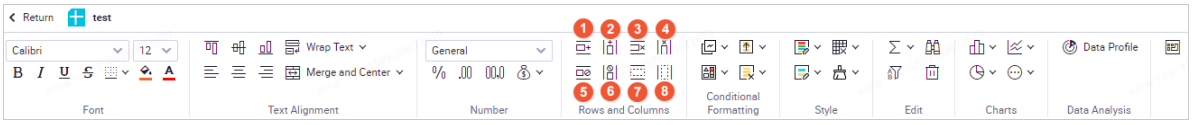
No.	Feature	Description
⑦	**Align Right**	Align text horizontally to the right.
⑧	**Merge and Center**	Merge multiple cells to one cell and center the content in the cell.

- **Number**



No.	Feature	Description
①	**Data Type**	Specify the type of data held in cells. You can select General, Number, Currency, Short Date, Long Date, Time, Percentage, Fraction, Scientific, and Text from the drop-down list.
②	**Percentage**	Apply the percentage format to numbers.
③	**Two Decimal Places**	Round numbers to two decimal places.
④	**1000 Separator**	Display numbers with thousands separators, for example, **1,005**.
⑤	**Currency**	Add a currency sign to numbers. The following currency signs are supported: yuan sign (¥), dollar sign ($), pound sign (£), euro sign (€), and franc sign (Fr).

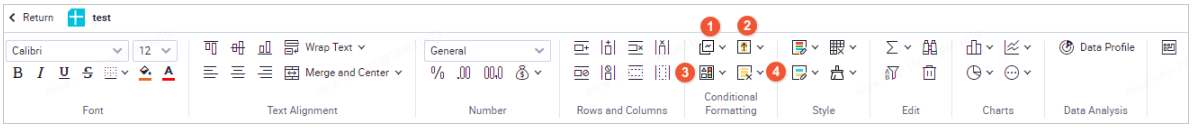
- **Rows and Columns**



No.	Feature	Description
①	**Insert Row**	Insert a row to the workbook.
②	**Insert Column**	Insert a column to the workbook.
③	**Delete Row**	Delete rows from the workbook.

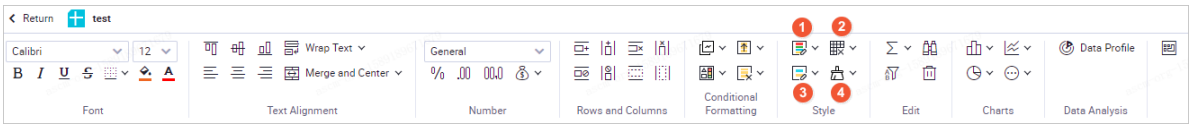
No.	Feature	Description
④	**Delete Column**	Delete columns from the workbook.
⑤	**Lock Row**	Lock the rows before the selected row in the workbook.
⑥	**Lock Column**	Lock the columns before the selected column in the workbook.
⑦	**Hide Row**	Hide rows in the workbook.
⑧	**Hide Column**	Hide columns in the workbook.

- **Conditional Formatting**



No.	Feature	Description
①	**Highlight cell rules**	Specify the rules for highlighting cells.
②	**Data Bar/Color Scale**	Format cells by using data bars and color scales.
③	**Icon Set**	Format cells by using icon sets. The icon sets include directional icons, shapes, indicators, and rating icons.
④	**Clear Rule**	Clear the formatting. You can select **Clear Rules from Selected Cells** or **Clear Rules from Entire Sheet** from the drop-down list.

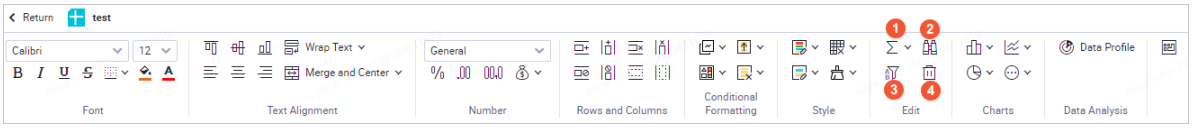
- **Style**



No.	Feature	Description
①	**Apply table style**	Apply a predefined table style to cells.
②	**Delete**	Remove the applied table style.
③	**Cell Style**	Apply a cell style to cells.

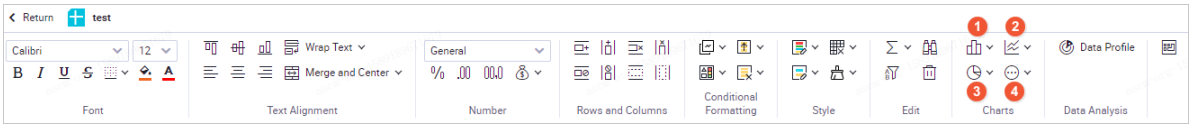
No.	Feature	Description
④	**Clear**	Clear the content or style in cells. You can select **Clear All**, **Clear Content**, or **Clear Style** from the drop-down list.

- **Edit**



No.	Feature	Description
①	**AutoSum**	Select an aggregation method. You can select **Sum**, **Average**, **Count Numbers**, **Max**, or **Min** from the drop-down list.
②	**Search**	Click **Search** or press **Ctrl+F** to open the search box.
③	**Sort and Filter**	Filter data and sort data in ascending or descending order.
④	**Clear**	Clear the content in cells.

- **Charts**

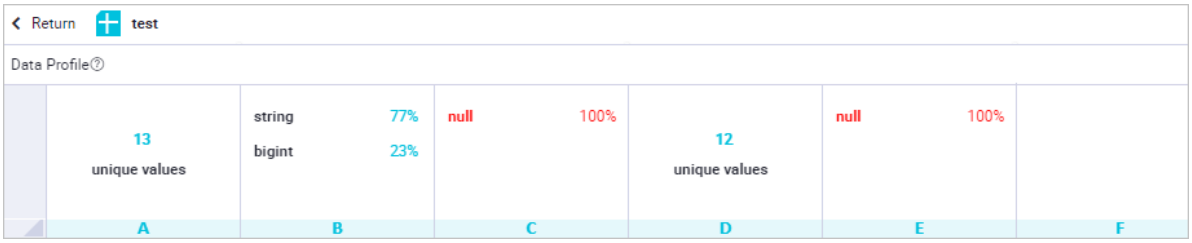


  - **Column Chart**: After you click the **Column Chart** icon, you can select **Column Chart**, **Stacked Column Chart**, or **100% Stacked Column Chart**.
  - **Line Chart**: After you click the **Line Chart** icon, you can select **Line Chart**, **Stacked Line Chart**, **100% Stacked Line Chart**, **Line Chart with Markers**, **Stacked Line Chart with Markers**, or **100% Stacked Line Chart with Markers**.
  - **Pie Chart**: After you click the **Pie Chart** icon, you can select **Pie Chart** or **Doughnut Chart**.
  - **More**: After you click the **More** icon, you can view more chart types, including area charts, bar charts, scatter charts, and stock charts.

- **Data Profile**

  The data profiling feature allows you to analyze the quality, structure, distribution, and statistics of the data. It also allows you to preview, profile, process, analyze, and visualize data. The data profiling feature analyzes data by column and allows you to view the distribution of data types and values of each column.

Select the data to be analyzed and click **Data Profile**. The data profiling feature displays the data type and value distribution of each column above the editing area in the form of charts and rich text.



Simple mode:

- For a column whose values are of the STRING or DATE type: The simple mode displays the values ranking top 2 based on frequency and their respective percentages, and the percentage of other values in the form of rich text. If the number of value types exceeds 50% of the total number of values, the simple mode displays the number of unique values.
- For a column whose values are of the INTEGER or FLOAT type: The simple mode displays the value distribution in the form of a histogram.
- For a column whose values are of the BOOLEAN type: The simple mode displays the proportions of different values in the form of pie charts.
- For a column whose values are of two or more data types: The simple mode displays the proportions of different data types in the form of pie charts. In addition, the system reminds you that the current column has dirty data. After the dirty data is cleared, the simple mode displays the data in one of the preceding forms based on the data type.
- For a column whose values are null values: The simple mode displays the percentage of null values in red.

Click **Detailed Mode** in the upper-right corner. In the **Data Profile** dialog box, you can view the profiling result of each column.

Detailed mode:

- For a column whose values are of the STRING or DATE type: The detailed mode displays the number of fields, the numbers and percentages of unique values, valid values, and null values, and the numbers of occurrences of the values ranking top 5 based on frequency.
- For a column whose values are of the INTEGER or FLOAT type: The detailed mode displays the number of fields, the numbers and percentages of unique values, valid values, zeros, and null values, the numbers of occurrences of the values ranking top 5 based on frequency, the statistics, and a histogram.
- For a column whose values are of the BOOLEAN type: The detailed mode displays the number of fields, the numbers and percentages of unique values, zeros, and null values, the numbers of occurrences of the values ranking top 5 based on frequency, and a pie chart.

> **Note** The system considers the true and false strings and the 0 and 1 integers as values of the BOOLEAN type.

- **List of Shortcut Keys**

  Click to view the shortcut keys for different features.

6.4. Dimension tables

6.4.1. Create and manage dimension tables

The dimension table feature allows you to create MaxCompute tables, import local data to MaxCompute tables, and edit MaxCompute tables in a visualized manner.

Prerequisites

1. A MaxCompute compute engine instance is bound to a DataWorks workspace.
2. A MaxCompute table is created. For more information about how to create a MaxCompute table, see Create a MaxCompute table.

Limits

- To create a dimension table in DataAnalysis, you must be an administrator, a project owner, or a developer of a DataWorks workspace.
- For a MaxCompute table that is created by using the dimension table feature, all fields in the MaxCompute table are of the STRING type. If you want to use fields of other data types, execute data definition language (DDL) statements to create a MaxCompute table on the **DataStudio** page.

Create a dimension table

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Development > DataAnalysis**.
3. On the DataAnalysis homepage, click **Experience Now**. The **Web Excel** page appears.
4. In the top navigation bar, click **Dimension**.
5. On the **Dimension** page, click the icon in the **New Dimension Table** section.
6. In the **New Dimension Table** dialog box, configure the parameters as required.

Parameter	Description
Target Workspace	The DataWorks workspace to which the MaxCompute table belongs.
Table Name	The name of the dimension table. A MaxCompute table will be used in the production environment. **Note** The table name can contain only letters, digits, and underscores (_), and must start with a letter.
Table Description	The description of the table, such as the purpose or features.
Field	The fields in the table. Only fields of the STRING type can be added.
Lifecycle	The lifecycle of the table. The table occupies storage resources in MaxCompute. To make sure that the resources can be recycled, select a proper lifecycle for the table from the drop-down list. If the specified lifecycle is exceeded, the table is deleted.

7. Select **I have known this risk and confirmed that as owner of this table, I am responsible for the subsequent changes to this table.** and click **OK** to go to the dimension table editing page to view and modify information about the table. For more information about how to edit a dimension table, see Edit a dimension table.

   The MaxCompute table created in DataAnalysis is maintained in the production environment. The creator of the table is responsible for the creation and maintenance of the table.

View and manage dimension tables

1. On the dimension table editing page, click **Return** in the upper-left corner or **Dimension** in the top navigation bar to go back to the Dimension page.
2. In the **All Dimension Tables** section, select **I created** or **Share it with me** from the drop-down list in the upper-right corner to view the tables in the corresponding category.

   You can also share dimension tables with specific members. For more information about how to share a dimension table, see Share a dimension table.
3. Click the file name of a required table or click **Edit** in the Operation column of the table to go to the dimension table editing page.

   On the Dimension page, you can also perform the following operations to manage a dimension table:

   - To change the owner of a dimension table, find the table and click **Change Owner** in the Operation column. In the **Change Owner** dialog box, select an owner from the New Owner drop-down list and click **OK**.
   - To delete a dimension table, find the table and click **Delete** in the Operation column. In the **Delete** message, click **OK**.

What's next

After you create a dimension table, go to the dimension table editing page and import data to this table. For more information about how to import data to a dimension table, see Import data to a dimension table.

6.4.2. Import data to a dimension table

After you create a dimension table, you can write data to the table for data analysis. You can also import data from a workbook, local CSV file, or local Excel file to the table for data analysis. This topic describes how to import data to a dimension table.

Prerequisites

A dimension table is created. For more information about how to create a dimension table, see Create a dimension table.

Procedure

1. Log on to the DataWorks console.
2. On the DataStudio page, click the ☰ icon in the upper-left corner and choose **All Products > Data Development > DataAnalysis**.
3. On the DataAnalysis homepage, click **Experience Now**. The **Web Excel** page appears.
4. In the top navigation bar, click **Dimension**.

5. In the **All Dimension Tables** section of the **Dimension** page, click the name of the table that you want to edit in the **File Name** column to go to the dimension table editing page.

   If no dimension table exists and you created one, the table editing page appears after the table is created. For more information about how to edit a dimension table, see Edit a dimension table.

6. On the dimension table editing page, click **Import** in the upper-right corner.
7. In the **Import** dialog box, select the type of the file that contains the data to be imported and configure parameters.

   > **Note** Only data of the STRING type can be imported to a dimension table. Data that is not of the STRING type will be automatically converted to the STRING type when it is imported.

   - **Workbook**

Parameter	Description
Spreadsheet	The workbook from which the data is to be imported. Select a workbook from the **Spreadsheet** drop-down list.
Sheet	The sheet from which the data is to be imported. Select a sheet from the **Sheet** drop-down list.
Data Preview	Displays the data in the selected sheet. When you preview the data in the selected sheet, you can determine whether to use the values in the first row as the column names by selecting or clearing **First Row as Field Names**.
Field Mapping	The mappings between the columns in the selected sheet and the fields in the dimension table.
Import Data Mode	The mode used to import data. Valid values: **Append** and **Overlay**.

   - **Local CSV file**

Parameter	Description
File	The CSV file from which the data is to be imported. Click **Select File(.csv)**, select a CSV file from the on-premises machine, and then click **Open**.
Original Character Set	The character set that is used by the selected CSV file. Valid values: **UTF-8** and **GBK**. If garbled characters appear, you can change the character set.
Separator	The row delimiter and column delimiter. ■ Valid values of row delimiters: \r\n, \n, and \r. ■ Valid values of column delimiters: commas (,), semicolons (;), and \t. If the cell data cannot be correctly separated, you can change the delimiters.

Parameter	Description
Data Preview	Displays the data in the selected sheet. When you preview the data in the selected sheet, you can determine whether to use the values in the first row as the column names by selecting or clearing **First Row as Field Names**.
Field Mapping	The mappings between the columns in the selected sheet and the fields in the dimension table.
Import Data Mode	The mode used to import data. Valid values: **Append** and **Overlay**.

- **Local Excel file**

Parameter	Description
File	The Excel file from which the data is to be imported. Click **Select File(.xlsx)**, select an Excel file from the on-premises machine, and then click **Open**.
Sheet	The sheet from which the data is to be imported. Select a sheet from the **Sheet** drop-down list.
Data Preview	Displays the data in the selected sheet. When you preview the data in the selected sheet, you can determine whether to use the values in the first row as the column names by selecting or clearing **First Row as Field Names**.
Field Mapping	The mappings between the columns in the selected sheet and the fields in the dimension table.
Import Data Mode	The mode used to import data. Valid values: **Append** and **Overlay**.

8. Click **OK**.
9. Click **Save** in the upper-right corner of the page.

   After you save the dimension table, you can click **Diff** in the upper-right corner of the page to check whether the changes meet expectations to prevent misoperations.

6.4.3. Edit a dimension table

This topic describes how to edit a dimension table in a visualized manner to modify the information of a MaxCompute table that you created. You do not need to write SQL code.

Prerequisites

A dimension table is created. For more information, see Create a dimension table.

Procedure

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Development > DataAnalysis**.

3. On the DataAnalysis homepage, click **Experience Now**. The **Web Excel** page appears.
4. In the top navigation bar, click **Dimension**.
5. In the **All Dimension Tables** section of the **Dimension** page, click the name of the table that you want to edit in the **File Name** column to go to the dimension table editing page.
6. On the dimension table editing page, view and modify the information about the dimension table.

   In the left-side pane of the dimension table editing page, you can view the table information, such as the workspace, table name, table description, lifecycle, and field description. To view the details of the dimension table, click the link below **Table Details** to go to the **Data Map** page. For more information, see View the details of a table.

   To modify the settings of the dimension table, perform the following steps: Click **Modify field settings**. In the **Modify the field settings dimension table** dialog box, change the values of **Table Description** and **Lifecycle** and click OK. You can also add fields to the table in this dialog box.

   The right side of the dimension table editing page displays all the data in the MaxCompute table as a workbook. The values in the first row are field names. You can double-click a cell to modify the content of a field in the corresponding row.
7. Click **Save** in the upper-right corner of the page to save the changes.

   After you save the dimension table, you can view all the data in the table. You can also click **Diff** in the upper-right corner of the page to view all the data in the **Diff Results** dialog box.

6.4.4. Share a dimension table

If you want to collaboratively edit a dimension table with multiple members, you can share the dimension table and grant the members the permissions to edit the table. This topic describes how to share a dimension table and grant edit or read permissions to specified members.

Prerequisites

The **Allow Sharing** switch is turned on on the **Management** page.

Procedure

1. Log on to the DataWorks console.
2. On the DataStudio page, click the ☰ icon in the upper-left corner and choose **All Products > Data Development > DataAnalysis**.
3. On the DataAnalysis homepage, click **Experience Now**. The **Web Excel** page appears.
4. In the top navigation bar, click **Dimension**.
5. In the **All Dimension Tables** section of the **Dimension** page, click the name of the table that you want to edit in the **File Name** column to go to the dimension table editing page.

   If no dimension table exists and you created one, the table editing page appears after the table is created. For more information about how to edit a dimension table, see Edit a dimension table.
6. In the upper-right corner of the dimension table editing page, click **Share**. In the dialog box that appears, configure the sharing method as required.

   You must configure the following information before you can share a dimension table with other members:

- **Link**: After you specify Users with Edit Access and Users with Read Access, click **Copy Link** and send the copied URL to specified members.
- **Users with Edit Access**: To specify members with permissions to edit the dimension table, click **Add** in the **Users with Edit Access** section. In the Share File with These Users dialog box, enter and select the names of the members to be granted the edit permissions, and click **OK**.

  > **Note** You can grant the edit permissions to a maximum of 10 members.

- **Users with Read Access**: To specify members with permissions to read the dimension table, click **Add** in the **Users with Read Access** section. In the Share File with These Users dialog box, enter and select the names of the members to be granted the read permissions, and click **OK**.

  > **Note** You can grant the read permissions to a maximum of 30 members.

After the sharing method is configured, you can send the URL to specified members. The members can use the URL to access the dimension table. You can go back to the **Dimension** page to view shared dimension tables.

6.5. Report

6.5.1. Create a report

This topic describes how to create a report. After a report is created, you can rename or delete the report.

Procedure

1. Log on to the DataWorks console.
2. On the DataStudio page, click  in the upper-left corner and choose **All Products > DataAnalysis**.
3. On the DataAnalysis homepage, click **Experience Now**.
4. On the **Web Excel** page, click **Report** in the top navigation bar.
5. Click in the **New Report** section.

   Alternatively, click a template in the **New Report** section to create a report.

   > **Note** If you have created reports, you can search for a report by entering its name in the search box in the **All Reports** section. Then, click the report name in the **File Name** column to go to the report editing page.

6. In the **New Report** dialog box, set the **Report Name** and **Report Description** parameters.
7. Click **OK**.

Result

After the report is created, it appears in the **All Reports** section. In this section, you can view all created reports. In addition, you can rename or delete a report.

- To rename a report, perform the following steps: Find the target report and click **Rename** in the

Operation column. In the **Rename** dialog box, enter the new name in the **File Name** field and click **OK**.

- To delete a report, perform the following steps: Find the target report and click **Delete** in the Operation column. In the **Delete** message, click **OK**.

6.5.2. Edit a report

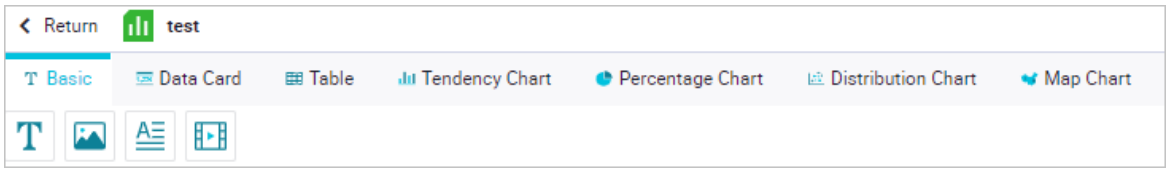
This topic describes how to edit, preview, save, share, and release a report, and save a report as a template.

Procedure

1. Go to the Report page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click in the upper-left corner and choose **All Products > DataAnalysis**.
   iii. On the DataAnalysis homepage, click **Experience Now**.
   iv. On the **Web Excel** page, click **Report** in the top navigation bar.
2. Go to the report editing page.

   Use one of the following methods to go to the report editing page:
   - After you create a report, the report editing page appears.
   - In the **All Reports** section of the Report page, click the name of the target report in the **File Name** column.
3. Drag controls from the menu bar to the canvas. In this topic, the **Bar Chart** control on the **Tendency Chart** tab is dragged to the current report.

   You can drag a control from the menu bar to the canvas to use the control as a component in the report.

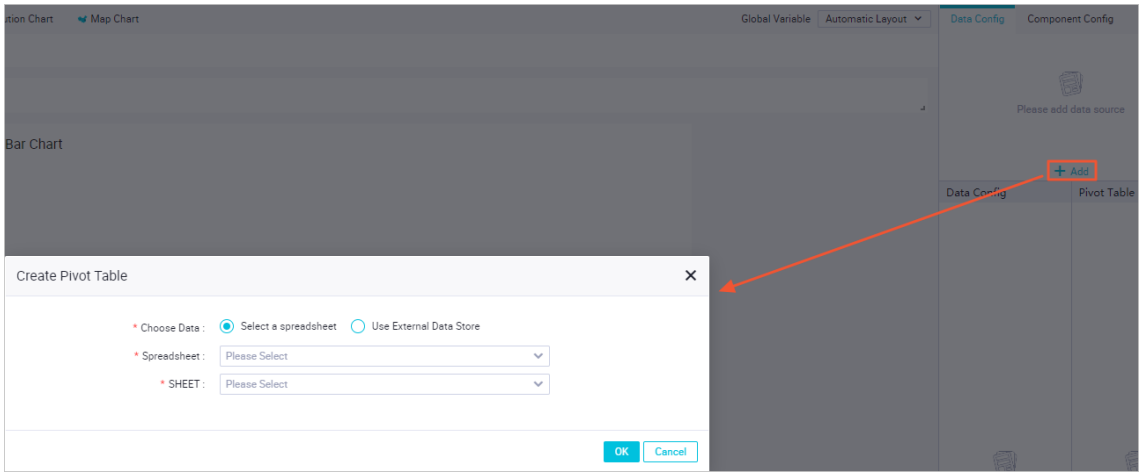


4. Click the **Bar Chart** component. Then, select a data store on the **Data Config** tab of the configuration section.

   If the required data store is added, click the name of the data store on the **Data Config** tab.

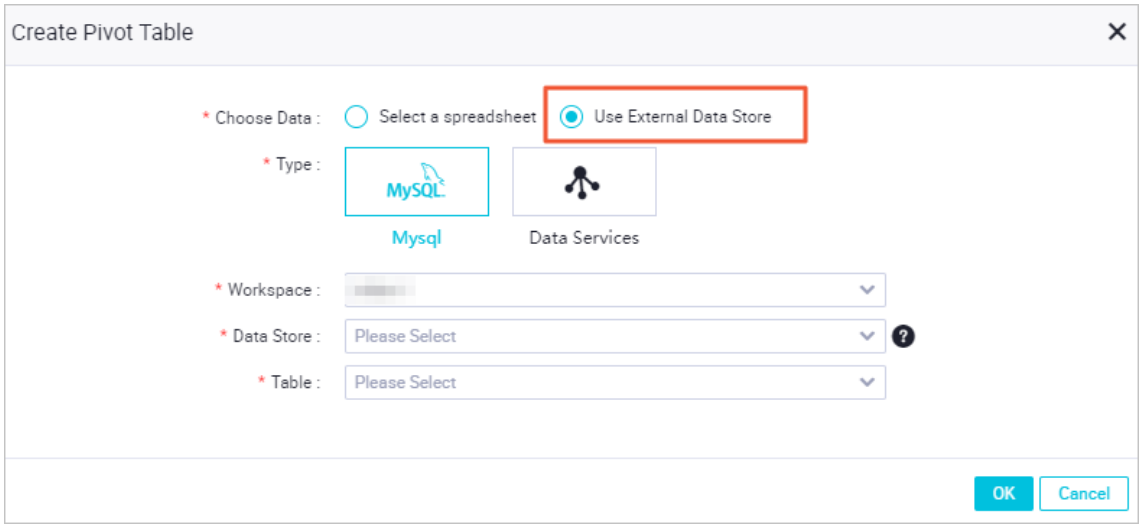
   If you need to add a data store, click **Add** on the **Data Config** tab. In the **Create Pivot Table** dialog box, set relevant parameters and click **OK**.

   You can set the Choose Data parameter to Select a spreadsheet or Use External Data Store.
   - **Select a spreadsheet**: You can select an editable worksheet in a workbook of the current user as the data store.



Parameter	Description
Choose Data	The range of the data to be analyzed. Select **Select a spreadsheet**.
Spreadsheet	The workbook from which the data is analyzed. Select a workbook from the **Spreadsheet** drop-down list.
SHEET	The worksheet of which the data is analyzed. Select a worksheet from the **SHEET** drop-down list.

- **Use External Data Store**: You can select a MySQL data store or the API of DataService Studio.

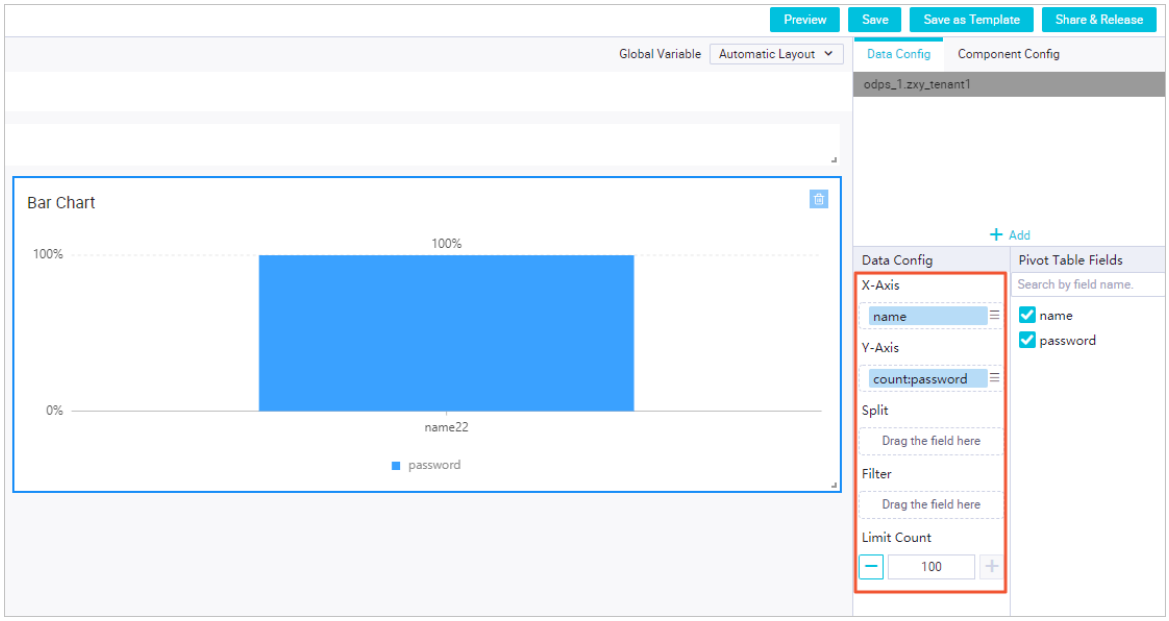


Parameter	Description
Choose Data	The range of the data to be analyzed. Select **Use External Data Store**.
Type	The type of the data source. Valid values: **Mysql** and **Data Services**.
Workspace	The workspace where the MySQL data store resides.

Parameter	Description
Data Store	The name of the connection to the data store. To create a connection to a data store, perform the following steps: Click the **Workspace Manage** icon in the upper-right corner. On the page that appears, click **Data Source** in the left-side navigation pane. On the Data Source page, create a connection to a data store. **Note** This parameter is available only when you set the Type parameter to **Mysql**.
Table	The table of which the data is analyzed. **Note** This parameter is available only when you set the Type parameter to **Mysql**.
API Group	The API group to which the API belongs. **Note** This parameter is available only when you set the Type parameter to **Data Services**.
API	The API to be used as the data source. **Note** This parameter is available only when you set the Type parameter to **Data Services**.

5. Select fields as statistical items.

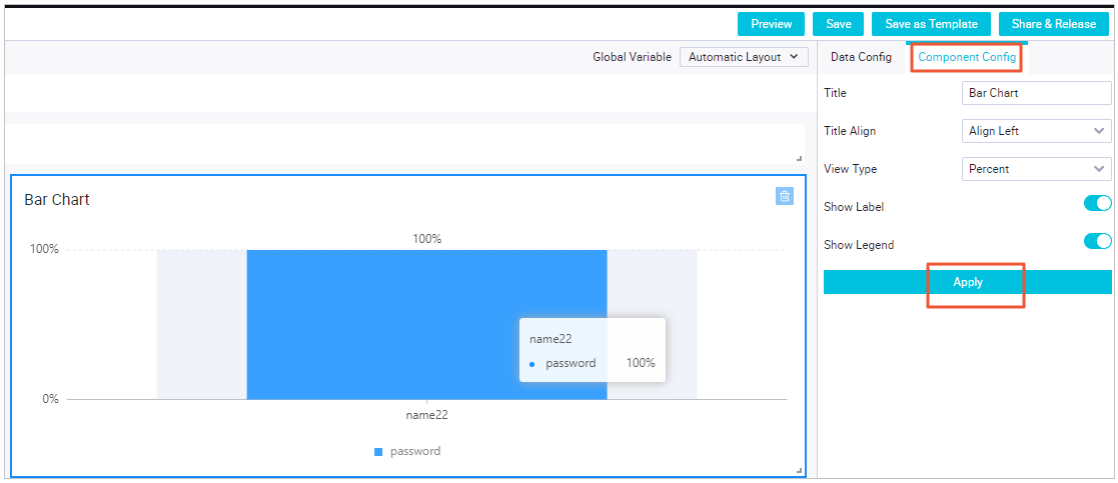
   The fields that are required vary with the chart type. For example, you must specify the X-axis, Y-axis, split, and filter fields for a column chart by dragging fields from the **Pivot Table Fields** section to the **Data Config** section. You can also specify the number of vertical columns that can appear in the column chart.



Multiple charts can use the same data store. They can use the data store in different ways without affecting each other. A chart can use only one data store. When you click a chart and drag fields from the Pivot Table Fields section to the Data Config section, the chart is associated with the data store.

6. Configure the information about the column chart.
   i. On the canvas, click the **Bar Chart** component.
   ii. Click the **Component Config** tab in the right-side configuration section.

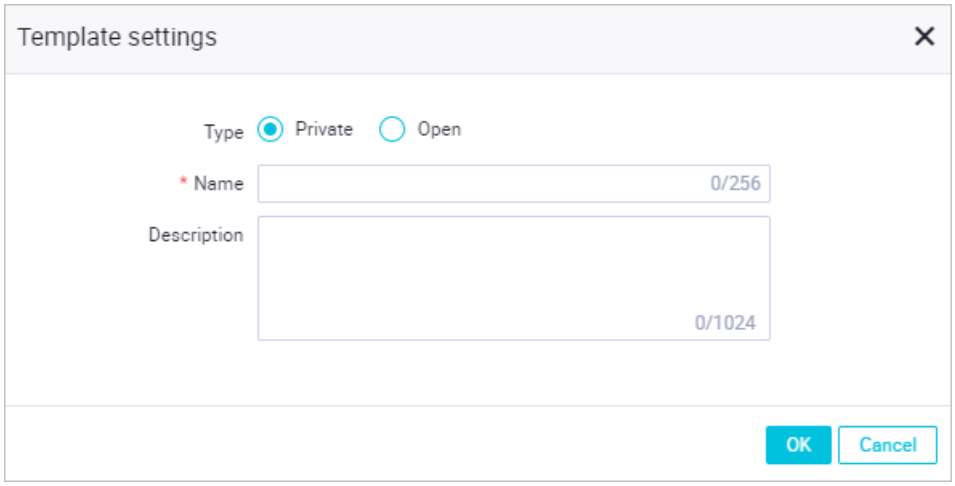
iii. On the **Component Config** tab, set relevant parameters.



Parameter	Description
Title	The title of the component.
Title Align	The alignment of the chart title. Valid values: **Align Left**, **Align Center**, and **Align Right**.
View Type	The display mode of vertical columns. Valid values: **Stack**, **Parallel**, and **Percent**.
Show Label	Specifies whether to display labels for the component.
Show Legend	Specifies whether to display legends for the component.

7. Click **Apply**.
8. Return to the **Report** page or preview, save, share, or release the report as needed.
   - **Return**: You can click **Return** in the upper-left corner to return to the **Report** page. On the Report page, you can view other reports and go to the report editing page of other reports.
   - **Preview**: You can click **Preview** in the upper-right corner to preview the report.
   - **Save**: You can click **Save** in the upper-right corner to save the report, so that you can open and edit the saved report next time.
   - **Save as Template**: You can save a report as a template, so that you can create a report based on the template. Perform the following steps to save a report as a template:
     a. On the report editing page, click **Save as Template** in the upper-right corner.
     b. On the **Preview** page, click **Next Step (Template setup)**.

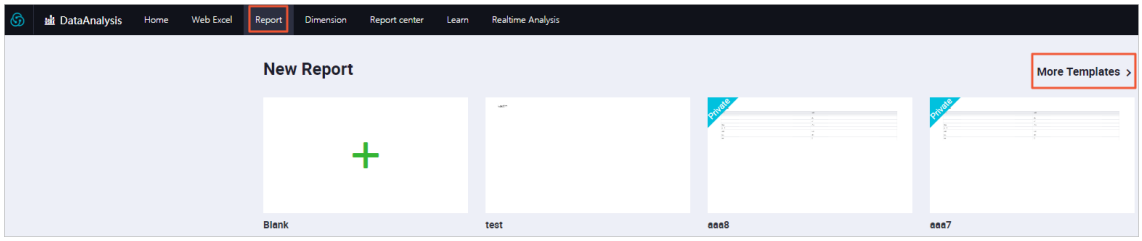
c. In the **Template settings** dialog box, set relevant parameters.



Parameter	Description
Type	Specifies whether to show or hide the template for other users. Valid values: **Private** and **Open**.
Name	The name of the template. The name can be up to 256 characters in length.
Description	The description of the template. The description can be up to 1,024 characters in length.

d. Click **OK**.

After you save the report as a template, you can click the template in the New Report section of the **Report** page to create a report.



- **Share & Release**: You can click **Share & Release** in the upper-right corner to share and release the report. You can share the report with specified users or all users. If you need to share the report to specific users, click **Add** to specify the users.

7.Administration

7.1. Overview

Generally, developers need to test workflows and nodes on the Operation Center page.

As a key tool for routine O&M, Operation Center enables you to manage and maintain the workflows and nodes that you have committed. The Operation Center service consists of four modules: Dashboard, Nodes, Node Instances, and Monitor.

- **Dashboard**: enables you to view and manage all global nodes of DataWorks. It displays various information, including **Instances**, **Instances Run Today**, **Node Runtime**, **Instances Run in the Last Month**, **Nodes with Errors in the Last Month**, and **Node Types** of the current workspace.
- **Nodes**: provides **Recurring** and **Manually Triggered**.
- **Node Instances**: provides **Recurring**, **Manually Triggered**, **Smoke Test** and **Retroactive**. You can manage them in a list view or DAG.
  - The list view displays the running status of nodes in a list. You can add multiple alerts at a time, change owners, and add nodes to baselines.
  - In the DAG, you can maintain and manage the running status of nodes and their dependencies on ancestor and descendant nodes. You can also perform operations, such as retroactive data generation and rerun, for a single node.
- **Monitor**: provides **Baseline Instances**, **Baselines**, **Events**, **Alert Triggers**, and **Alerts**.

7.2. Dashboard

The Dashboard page provides information about the node running status, the trend of the number of nodes that were run, the node running time, and the nodes with errors.

Go to the Dashboard page

1. Log on to the DataWorks console.
2. Click ☰ in the upper-left corner and choose **All Products > Operation Center**. By default, the **Dashboard** page appears.

The **Dashboard** page consists of the following sections: **Instances**, **Instances Run Today**, **Node Runtime**, **Nodes with Errors In the Last Month**, **Instances Run In the Last Month**, and **Node Types**.

> Note

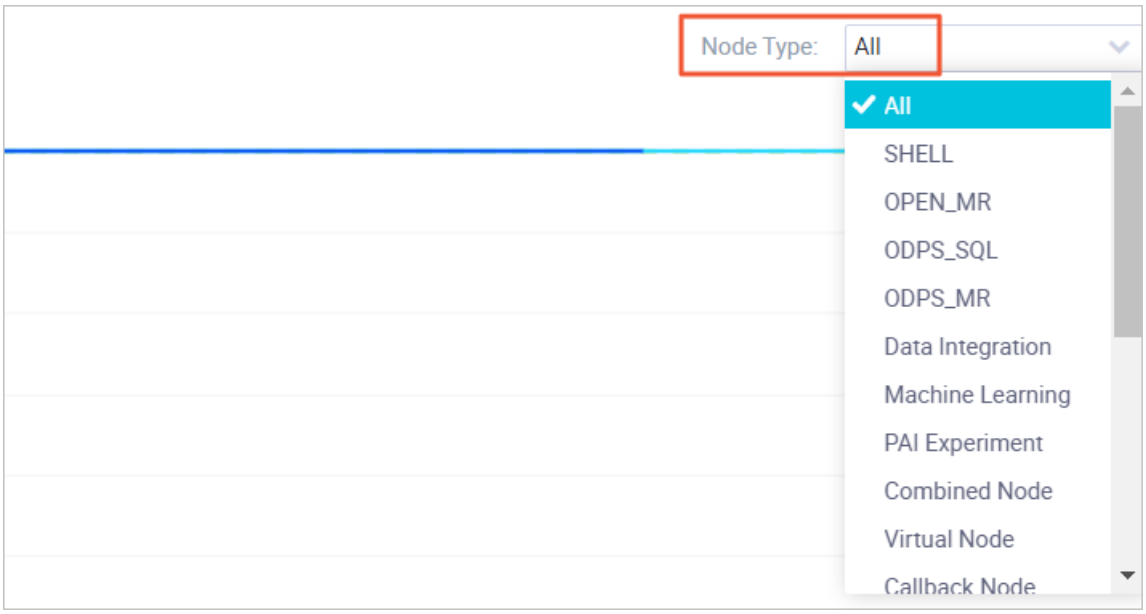
- Pending (Schedule): the instances whose scheduled time has not arrived. These instances will be automatically run when their scheduled time arrives.
- Pending (Resources): the instances whose scheduled time has arrived. These instances are waiting for scheduling resources.

View the summary of instance running

The **Instances** section displays the numbers of auto triggered node instances that were run today and yesterday, as well as the historical average. If the deviations among the three numbers are large, an exception occurred during a specific period of time. Further check and analysis are required.

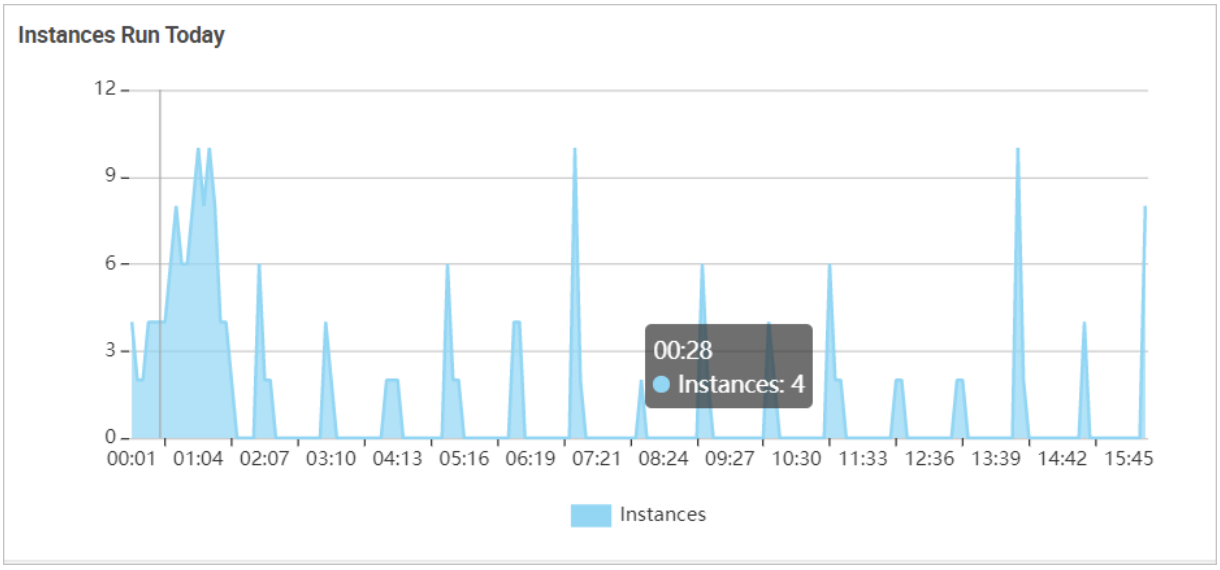
The statistical chart on the right of the Instances section shows three lines in different colors, representing the numbers of instances completed between 00:00 and 23:00 of today and yesterday, as well as the historical average. You can also view the number and proportion of instances in the pie chart on the left.

In the upper-right corner of the statistics chart, you can select a node type from the **Node Type** drop-down list to view statistics on the specified type of nodes.



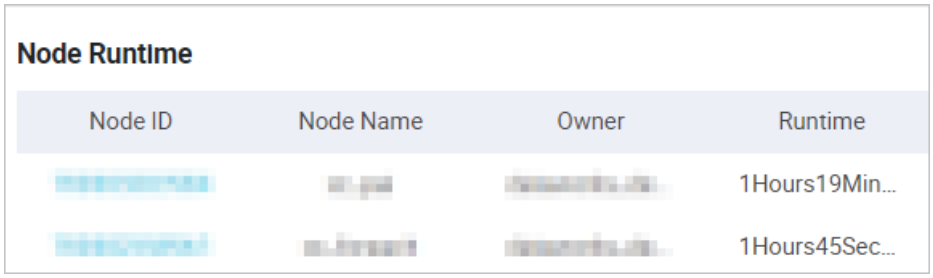
View the statistics on instances run today

The **Instances Run Today** section displays the numbers of node instances that were running at different time points of the current day. In the chart, you can find the time point when peak concurrency occurred and the maximum number of concurrent nodes. Based on the information, you can determine whether to avoid node scheduling at the peak hours and adjust the scheduling time of nodes.



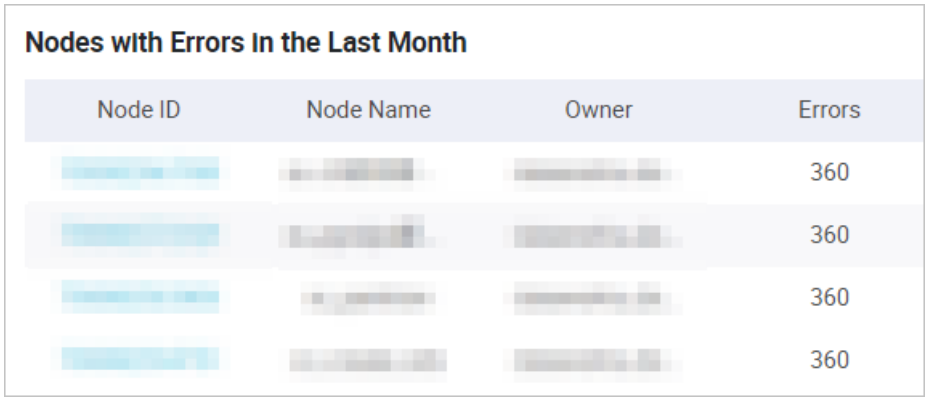
View the rankings of nodes based on their running time

The **Node Runtime** section displays nodes with a specified timestamp in the current workspace by their running time. By default, this section displays the top 10 nodes in descending order of their running time. You can view **Node ID**, **Node Name**, **Owner**, and **Runtime** of each node.



View the rankings of nodes with errors in the last month

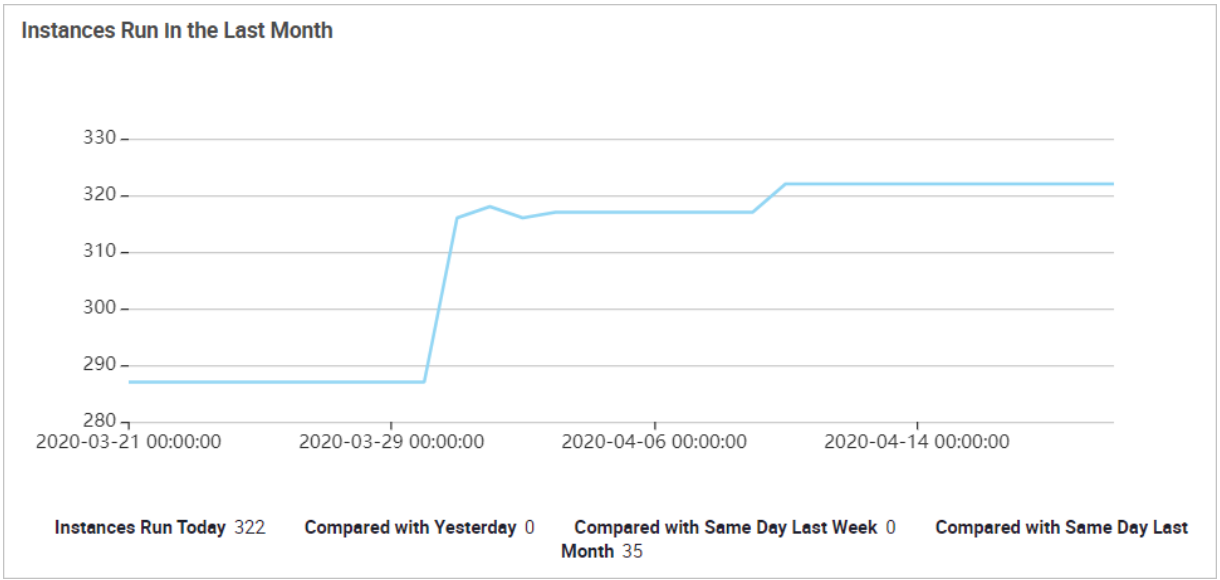
The **Nodes with Errors In the Last Month** section displays the top 10 nodes with the most errors in the last month. You can view **Node ID**, **Node Name**, **Owner**, and **Errors** of each node.



You can click a node ID to go to the node details page.

View the trend of the number of instances that were run

The **Instances Run In the Last Month** section displays the number of instances that were run today, comparison with yesterday, comparison with the same day last week, and comparison with the same day last month.



View the node distribution by node type

In the Node Types section, you can move the pointer over a sector of the pie chart to view the number and proportion of nodes of the specific type.

7.3. Real-time node O&M

7.3.1. Manage real-time computing nodes

The Stream Task page of the DataWorks console displays all real-time computing nodes. You can view the basic information and details of the nodes on this page. You can also configure alert rules for the nodes that you want to monitor. This way, you can identify and handle exceptions at the earliest opportunity if an error occurs on a node.

Limits

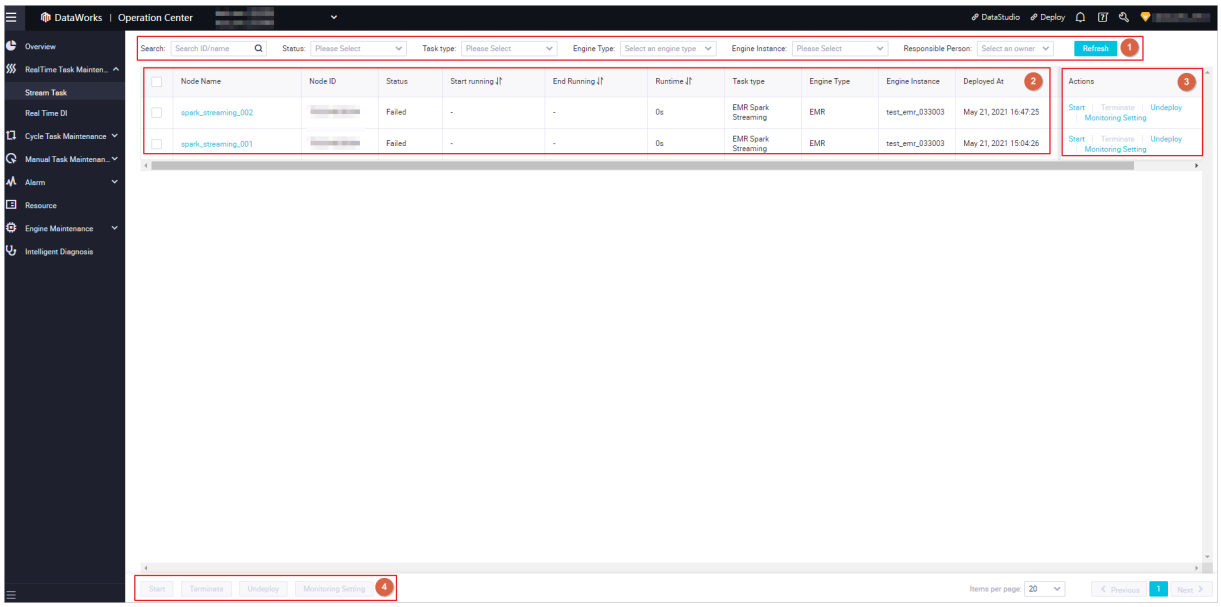
DataWorks supports O&M only for EMR Spark Streaming and EMR Streaming SQL nodes on the Stream Task page.

Go to the Stream Task page

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Task Operation > Operation Center**.
3. On the Operation Center page, choose **RealTime Task Maintenance > Stream Task** to go to the Stream Task page.

View real-time computing nodes in the node list

The Stream Task page displays all real-time computing nodes in the production environment. You can view the basic information of a real-time computing node and perform operations such as starting, stopping, or undeploying a real-time computing node. You can also configure alert rules for a real-time computing node.



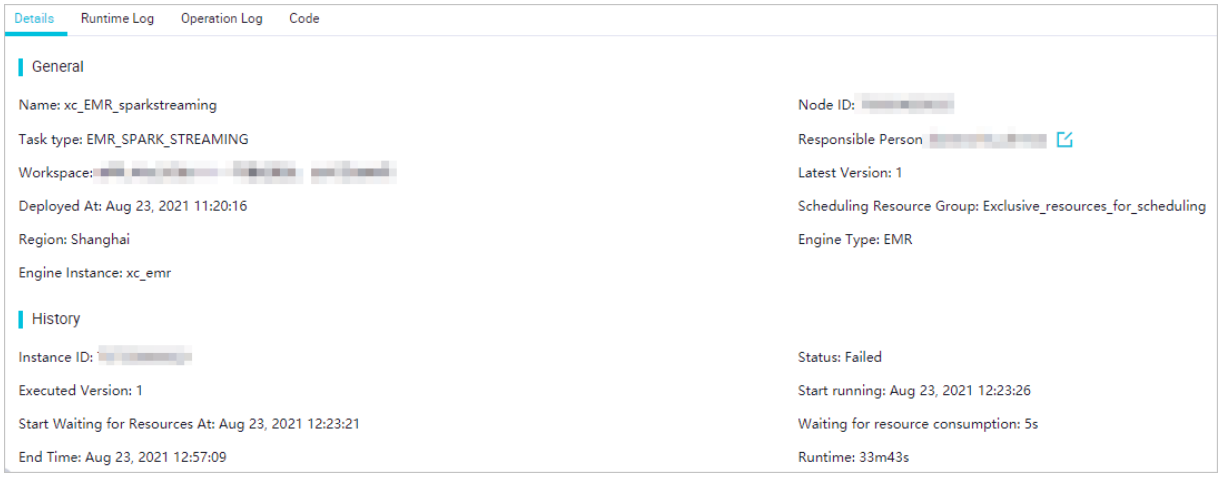
Section	Description

Section	Description
1	In this section, you can search for a real-time computing node by node ID or node name. You can also specify one of the following filter conditions to perform the operation: **Status**, **Task type**, **Engine Type**, **Engine Instance**, and **Responsible Person**. Note • If you search for nodes by node ID or node name, the search results are affected by other filter conditions that you specified. Only the nodes that meet all the filter conditions that you specified are displayed. • If you search for nodes by node name, fuzzy match is supported. After you enter a keyword, all real-time computing nodes whose names contain the keyword are displayed.
2	In this section, you can view the basic information and details of a real-time computing node. • **Node Name**: the name of the node. You can click the node name to open the details panel of the node. • **Node ID**: the ID of the node. • **Status**: the state of the node. The node can be in one of the following states: **Not Running**, **Generating**, **Pending (Resources)**, **Starting**, **Running**, **Stopped**, **Failed**, **Restarting**, and **Undeploying**. • **Start running**: the time when the node started to run. • **End Running**: the time when the running of the node was complete. • **Runtime**: the duration for which the node was running. Unit: seconds. • **Task type**: the type of the node. Note DataWorks supports O&M only for EMR Spark Streaming and EMR Streaming SQL nodes on the Stream Task page. • **Engine Type**: the type of the compute engine used to run the node. DataWorks supports O&M only for real-time computing nodes that are run by using the E-MapReduce (EMR) compute engine. • **Engine Instance**: the name of the EMR compute engine instance that is associated with the workspace to which the node belongs. • **Deployed At**: the time when the node was deployed. The time is in the yyyy-MM-dd HH:mm:ss format. • **Responsible Person**: the owner of the workspace to which the node belongs. Note You can rank all real-time computing nodes in ascending or descending order based on the values in the **Start running**, **End Running**, or **Runtime** column.

Section	Description
3	In this section, you can perform the following operations on a real-time computing node: • **Start**: Start the node. • **Terminate**: Terminate the node. • **Undeploy**: Undeploy the node. • **Monitoring Setting**: Configure alert rules for the node. If the node fails to run, the system sends an alert notification to the specified alert contact by text message, email, mobile phone, DingTalk chatbot, or webhook URL. For more information about how to configure an email address and a mobile number for the alert contact, see What can I do if I am unable to receive alert notifications after I configure an alert in Operation Center? For more information about how to configure a DingTalk chatbot and obtain a webhook URL, see Scenario practices: Send alert notifications to a DingTalk group.
4	In this section, you can perform an operation on multiple real-time computing nodes at a time. You can select multiple nodes and click **Start**, **Terminate**, **Undeploy**, or **Monitoring Setting** to perform the related operation on these nodes.

View the details of a real-time computing node

On the Stream Task page, find the real-time computing node whose details you want to view and click the node name to open the details panel of the node. In the node details panel, you can view the details, run logs, operation logs, and code of the node on the **Details**, **Runtime Log**, **Operation Log**, and **Code** tabs. This allows you to quickly find the operations performed on the node and operation records, obtain the details of the errors reported for the node, and fix the errors.



7.3.2. Manage real-time synchronization nodes

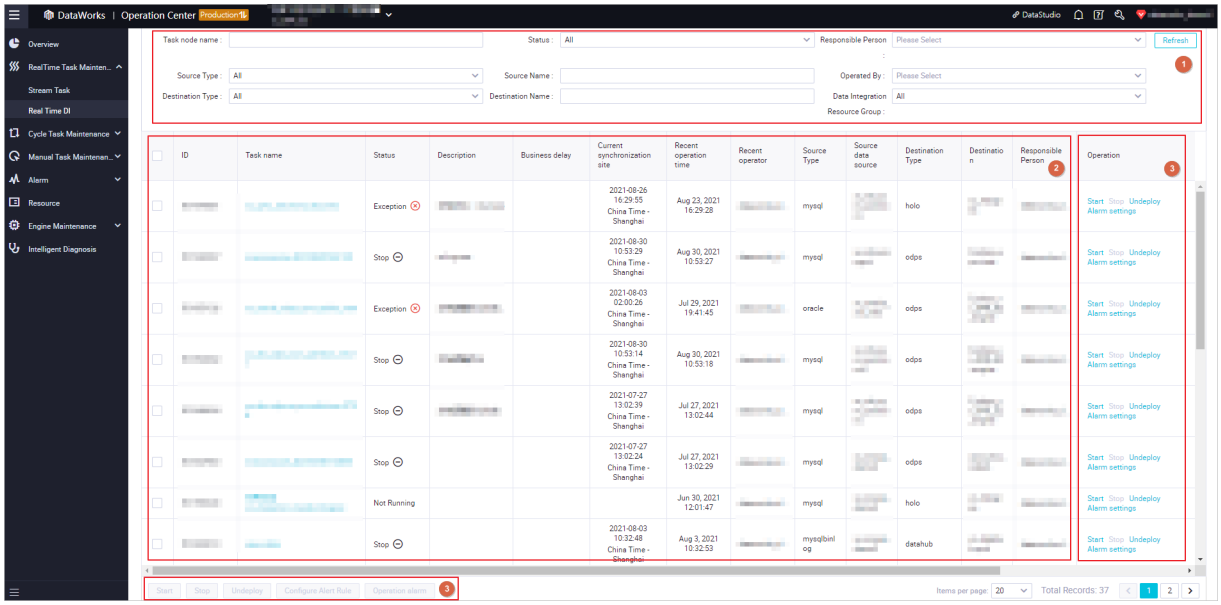
The Real Time DI page of the DataWorks console displays all real-time synchronization nodes that are committed to and run by the scheduling system.

Go to the Real Time DI page

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Task Operation > Operation Center**.
3. On the Operation Center page, choose **RealTime Task Maintenance > Real Time DI** to go to the Real Time DI page.

View real-time synchronization nodes in the node list

The node list on the Real Time DI page displays real-time synchronization nodes that are run by the scheduling system. You can manage these nodes and perform O&M operations on these nodes. For example, you can view the basic information, properties, and details of a real-time synchronization node. You can also start or undeploy a real-time synchronization node.

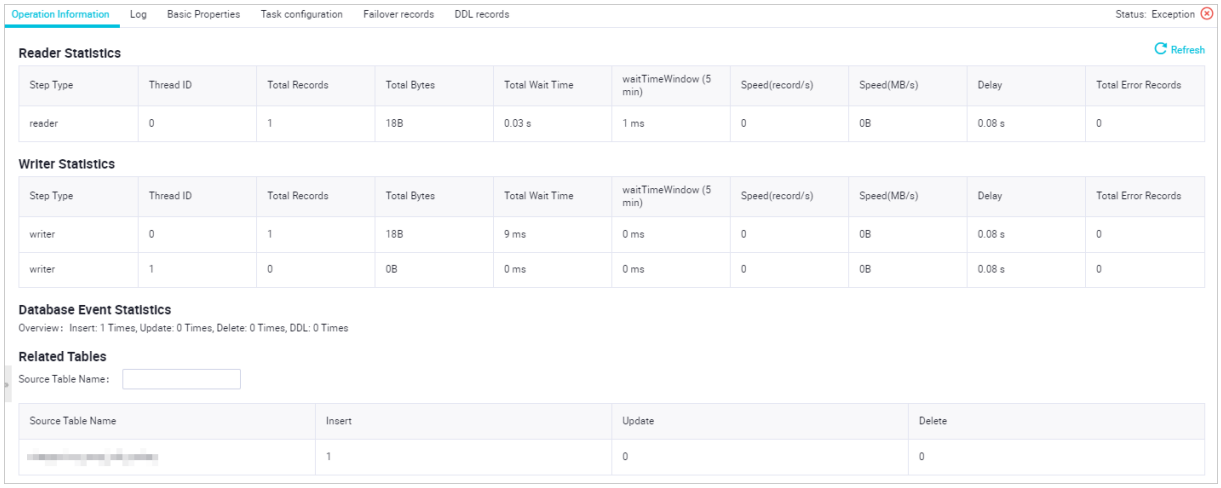


Section	Description
1	In this section, you can specify filter conditions to search for real-time synchronization nodes. You can specify the following filter conditions to search for nodes: **Task node name**, **Status**, **Responsible Person**, **Source Type**, **Source Name**, **Operated By**, **Destination Type**, **Destination Name**, and **Data Integration Resource Group**. **Note** When you search for nodes by node name, the search results are affected by other filter conditions that you specified. Only the nodes that meet all the filter conditions that you specified are displayed.

Section	Description
2	In this section, you can view the following information about a real-time synchronization node: • **ID**: the ID of the node. • **Task name**: the name of the node. You can click the node name to open the details panel of the node. • **Status**: the state of the node. The node can be in one of the following states: Running, Not Running, Waiting for Resources, Exception, and Stop. • **Description**: the description of the node. • **Business delay**: the period of time between the current time and the offset from which incremental data starts to be synchronized. • **Current synchronization site**: the offset at which incremental data is being synchronized. • **Recent operation time**: the time when the last operation was performed on the node. • **Recent operator**: the user who last performed an operation on the node. • **Source Type**: the type of the source. • **Source data source**: the name of the source. • **Destination Type**: the type of the destination. • **Destination**: the name of the destination. • **Responsible Person**: the owner of the workspace to which the node belongs. • **Current Start Site**: the offset from which incremental data starts to be synchronized. • **Data Read Speed (Bytes per Second)**: the speed at which the data is read. • **Recording Speed**: the speed at which the read data is written to logs. • **Data Integration Resource Group**: the resource group that is used to run the node.
3	In this section, you can perform the following operations on multiple nodes at a time: • **Start**: Start the nodes. • **Stop**: Stop the nodes that are running. • **Undeploy**: Undeploy the nodes that are not running, abnormal, or stopped. • **Configure Alert Rule**: Configure alert rules for the nodes. • **Operation alarm**: Delete, enable, or disable alert rules and modify alert rules based on metric types.

View the details of a real-time synchronization node

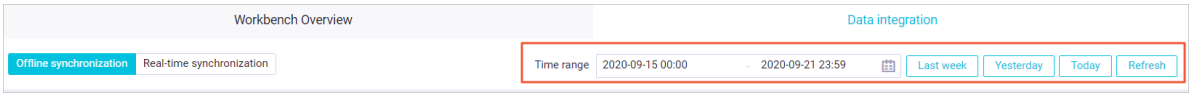
On the Real Time DI page, find the real-time synchronization node whose details you want to view and click the node name to open the details panel of the node. In the node details panel, you can view the details of the node on the **Details**, **Log**, **Basic Properties**, **Task configuration**, **Failover**, and **DDL Records** tabs.



Perform operations on a real-time synchronization node

- Start a real-time synchronization node

  Find the node that you want to start and click **Start** in the **Operation** column. The Start dialog box appears. After you set the parameters in the dialog box, click **Confirm**. Then, the system starts to run the node.

  If you select **Reset site**, you must set the **Start time point** and **Time zone** parameters.

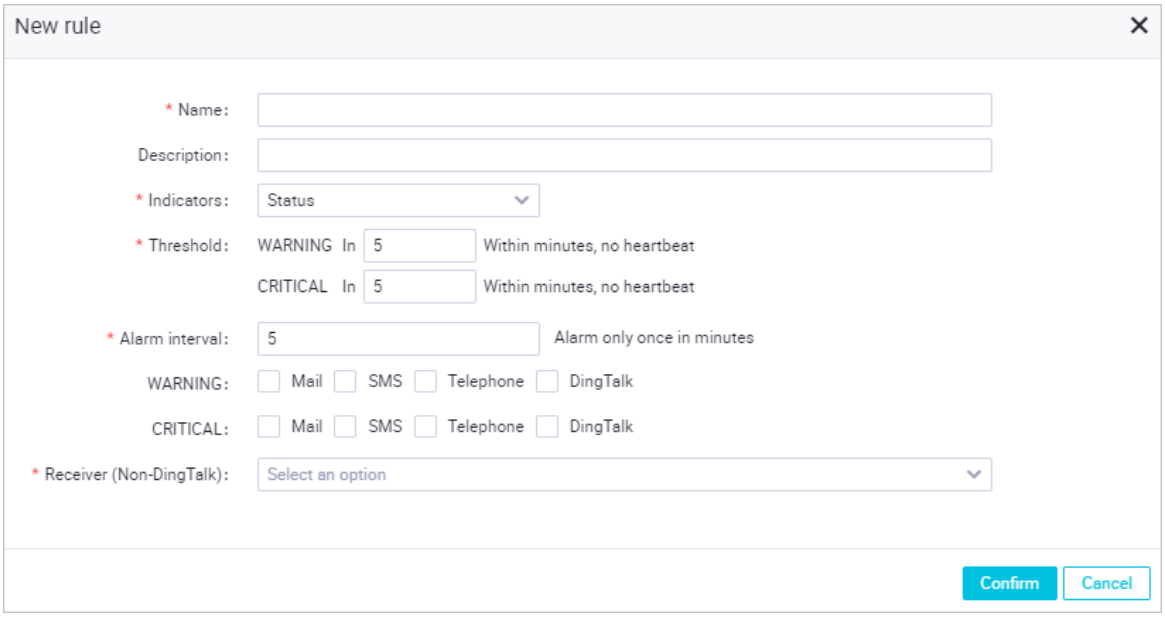
- Stop a real-time synchronization node

  Find the node that you want to stop and click **Stop** in the Operation column. In the message that appears, click **Stop**.

- Undeploy a real-time synchronization node

  Find the node that you want to undeploy and click **Undeploy** in the Operation column. In the message that appears, click **Undeploy**.

- View the alert settings of a real-time synchronization node
  - Find the node whose alert settings you want to view and click Alarm settings in the Operation column. On the **Alarm event** tab, you can view the alert events of the node and filter alert events by setting the **Occurrence time**, **Alarm Level**, and **Rules** parameters.
  - Click the **Alarm rules** tab. On this tab, you can view all the alert rules that are created for the node. You can view the metric type and state of an alert rule. You can also modify, disable, delete, or test an alert rule by clicking a button in the **Operation** column. If you want to create an alert rule, click **New rule** and set the parameters that are described in the following table.



Parameter	Description	Required
Name	The name of the alert rule.	Yes
Description	The description of the alert rule.	No
Indicators	The type of metric that triggers an alert. The Indicators parameter and the Threshold parameter must be used in pairs. The value of the Threshold parameter varies based on that of the Indicators parameter. Valid values: Status, Business delay, Failover, Dirty Data, and Not Supported by DDL Statements.	Yes

Parameter	Description	Required
Threshold	The thresholds for the metric type that is specified by the Indicators parameter. ■ If you set the Indicators parameter to **Status**, you must specify the interval at which alerts are triggered. ■ If you set the Indicators parameter to **Business delay**, you must specify a duration for service latency and another duration for which an alert can last after the duration for service latency elapses. ■ If you set the Indicators parameter to **Failover**, you must specify a duration and the maximum number of failovers that can be performed within the duration before an alert is triggered. ■ If you set the Indicators parameter to **Dirty Data**, you must specify a duration and the maximum number of dirty data records that are allowed within the duration before an alert is triggered. ■ If you set the Indicators parameter to **Not Supported by DDL Statements**, you need to only select an alert level.	Yes
Alarm interval	The minimum interval at which alerts are reported. Default value: 5. Unit: minutes. The minimum interval cannot be shorter than 1 minute.	Yes

Parameter	Description	Required
WARNING	The method used to receive a notification for a WARNING-level alert. Valid values: **Mail**, **SMS**, **Telephone**, and **DingTalk**. **Note** ■ Mail: If you want to use a RAM user to receive the notification, you must use an Apsara Stack tenant account to add the email address of the RAM user to user information. ■ SMS: If you want to use a RAM user to receive the notification, you must use an Apsara Stack tenant account to add the mobile number of the RAM user to user information. ■ Telephone: If you want to use a RAM user to receive the notification, you must use an Apsara Stack tenant account to add the mobile number of the RAM user to user information. ■ DingTalk: If you want to receive the notification by using a DingTalk chatbot, you must configure a DingTalk chatbot for your DingTalk group and add the keyword *DataWorks* for the DingTalk chatbot.	No

Parameter	Description	Required
CRITICAL	The method used to receive a notification for a CRITICAL-level alert. Valid values: **Mail**, **SMS**, **Telephone**, and **DingTalk**.	No
Receiver (Non-DingTalk User)	The alert contact to whom alert notifications are sent.	Yes

7.4. Auto triggered node O&M

7.4.1. Manage auto triggered nodes

Auto triggered nodes are automatically run as scheduled after they are committed to the scheduling system.

> **Note**
> - By default, the auto triggered node list displays nodes in all the workflows created by the current account.
> - After you commit a node, the scheduling system automatically generates and runs an instance of the node at 23:30 the next day. If you commit a node after 23:30, the scheduling system generates and runs an instance of the node on the third day.
> - Do not perform any operations on the **Workspace name_root** node, which is the root node of the workspace. All instances of auto triggered nodes depend on this node. If this node is frozen, instances of auto triggered nodes cannot be run.

Manage auto triggered nodes in the node list

The **Cycle Task** page displays auto triggered nodes that are committed to the scheduling system in a list.

Operation	Description
Filter	Find required nodes by setting parameters in the red box marked with 1 in the preceding figure. You can search for nodes by node name or node ID and set the **Node Type**, **Owner**, **My Nodes**, **Modified Today**, and **Frozen** parameters to filter nodes. **Note** When you search for nodes by node name, the search result is affected by other filter conditions you specified. Only the nodes that meet both the specified search condition and other filter conditions are returned in the search result.
DAG	Click **DAG** in the Actions column of a node to view the directed acyclic graph (DAG) of the node. You can view the node information, such as properties, operations logs, and code, in the DAG.

Operation	Description
Test	Click **Test** in the Actions column of a node to test the node. For more information, see Manage test instances.
Patch Data	Click **Patch Data** in the Actions column of a node and select an item from the drop-down list to generate retroactive data for the node. For more information, see Manage retroactive instances.
More operations	Click **More** in the Actions column of a node to perform more operations on the node. You can perform the following operations on the node: • Select **Freeze** to freeze the node. After the node is frozen, DataWorks can still generate instances of the node, but does not run the instances of the node and its descendant instances. • Select **Unfreeze** to unfreeze the node. After the node is unfrozen, DataWorks can normally run the instances of the node and its descendant instances. • Select **View Instances** to view the instances of the node. • Select **Configure Alert Trigger** to configure alert triggers for the node. • Select **Change Owner** to change the owner of the node. • Select **Add to Baseline** to add the node to a baseline. • Select **Change Resource Group** to change the resource group used to run the node if multiple resource groups exist in the workspace. • Select **Configure Data Quality Rules** to configure rules for monitoring the data quality of the node. • Select **View Lineage** to view the lineage of the node. • Select **View Ancestor and Descendant Nodes** to go to the **Node Information** page, where you can view node information on the **Ancestor Nodes** and **Descendant Nodes** tabs.
Batch operations	Select multiple nodes and click a button in the red box marked with 3 in the preceding figure to perform batch operations. The buttons include **Configure Alert Trigger**, **Change Owner**, **Change Resource Group**, **Add to Baseline**, **Freeze**, **Unfreeze**, and **Delete**.

Manage auto triggered nodes in a DAG

Click the name of a node or **DAG** in the Actions column to view the DAG of the node. In the DAG, you can right-click the node to perform related operations.

Operation	Description
Show Ancestor Nodes or **Show Descendant Nodes**	View ancestor or descendant nodes at one or more levels. If the workflow contains three or more nodes, the DAG displays only the current node and hides its ancestor and descendant nodes.
View Node Details	Go to the **Node Information** page to view the node information, including the input table, output table, ancestor nodes, and descendant nodes.
View Code	View the code of the node.
Edit Node	Go to the DataStudio page to modify the node.

Operation	Description
View Instances	View the instances of the node.
View Lineage	View the lineage of the node.
Test	Go to the **Smoke Test** dialog box. Set the **Smoke Test Instance Name** and **Data Timestamp** parameters and click **OK**. Then, the Test Instance page appears.
Run	Select **Current Node Retroactively**, **Current and Descendant Nodes Retroactively**, or **Mass Nodes Retroactively**.
Freeze	Pause the scheduling of the node.
Unfreeze	Resume the scheduling of the node after it is frozen.
Configure Data Quality Rules	Configure rules for monitoring the data quality of the node.

7.4.2. Manage auto triggered node instances

Auto triggered node instances are snapshots taken for auto triggered nodes that are run.

An instance is generated each time when an auto triggered node is run as scheduled. You can manage auto triggered node instances. For example, you can view the running status of instances, and stop, rerun, and unfreeze instances.

> **Note** DataWorks regularly generates instances for auto triggered nodes. Each generated instance runs the latest code. If you modify and recommit the node code after instances are generated, the instances that have not been run will run the latest code.

Manage auto triggered node instances in the instance list

You can manage auto triggered node instances in the instance list. For example, you can check operational logs, rerun instances, and stop running instances.

Operation	Description
Filter	Find required instances by setting parameters in the red box marked with 1 in the preceding figure. You can search for instances by node name or node ID and set the following parameters to filter instances: **Data Timestamp**, **Node Type**, **Run At**, **Solution**, **Workflow**, **Region**, **Engine type**, **Engine instance**, **Baseline**, **Owner**, **Recurrence**, **Status**, **My Nodes**, **My Nodes with Errors**, **My Incomplete Nodes**, and **Re-run node**. **Note** By default, the data timestamp is set to the previous day of the current day.

Operation	Description
Stop	Stop the instance. You can stop instances only when they are in the **Pending (Resource)** or **Running** state. After you perform this operation, the instance enters the **Failed** state.
Rerun	Rerun the instance. After the instance is rerun, its descendant instances will be run as scheduled. Perform this operation if an instance fails or it is not run as scheduled. **Note** This operation applies only to instances in the **Pending (Ancestor)**, **Successful**, or **Failed** state.
Rerun Descendant Nodes	Rerun the instance and its descendant instances. You must specify the instances to rerun. After they are run, their descendant instances will be run as scheduled. This operation applies to data recovery. **Note** You can select instances only in the **Pending (Ancestor)**, **Successful**, or **Failed** state. The value No appears in the **Meet Rerun Condition** column of instances in other states, and you cannot select the instances.
Set Status to Successful	Set the status of the instance to **Successful** and run its descendant instances as scheduled. You can perform this operation if an instance fails. **Note** This operation applies only to failed instances but not workflows.
Freeze	Freeze the instance if it is in the Running state.
Unfreeze	Unfreeze the instance after it is frozen. • If the instance is not run, DataWorks automatically runs this instance after its ancestor instances are run. • If all the ancestor instances have been run, the instance is directly set to Failed. You must manually rerun the instance.
Batch operations	Select multiple instances and click a button in the red box marked with 3 in the preceding figure to perform batch operations. The buttons include **Stop**, **Rerun**, **Set Status to Successful**, **Freeze**, and **Unfreeze**.

Manage auto triggered node instances in a DAG

Click the name of an instance or **DAG** in the Actions column to view the directed acyclic graph (DAG) of the instance. In the DAG, you can right-click the instance to perform related operations.

Operation	Description
Show Ancestor Nodes or **Show Descendant Nodes**	View ancestor or descendant instances at one or more levels. If a workflow contains three or more instances, the DAG displays only the current instance and hides its ancestor and descendant instances.

Operation	Description
View Runtime Log	View the operational logs of the instance if it is in the Running, Successful, or Failed state.
View Code	View the code of the instance.
Edit Node	Go to the DataStudio page to modify the node to which the instance belongs.
View Lineage	View the lineage of the instance.
More operations	View more instance information on the **General**, **Context**, **Runtime Log**, **Operation Log**, and **Code** tabs.
Stop	Stop the instance if it is in the Pending (Resource) or Running state. After you perform this operation, the instance enters the Failed state.
Rerun	Rerun the instance. After the instance is rerun, its descendant instances will be run as scheduled. Perform this operation if an instance fails or it is not run as scheduled. **Note** This operation applies only to instances in the **Pending (Ancestor)**, **Successful**, or **Failed** state.
Rerun Descendant Nodes	Rerun the instance and its descendant instances. You must specify the instances to rerun. After they are run, their descendant instances will be run as scheduled. This operation applies to data recovery. **Note** You can select instances only in the **Pending (Ancestor)**, **Successful**, or **Failed** state. The value No appears in the **Meet Rerun Condition** column of instances in other states, and you cannot select the instances.
Set Status to Successful	Set the status of the instance to Successful and run its descendant instances as scheduled. You can perform this operation if an instance fails. **Note** This operation applies only to failed instances but not workflows.
Resume	Continue to run the instance after it fails.
Emergency Operations	Perform emergency operations, including **Delete Dependencies**, **Change Priority**, and **Force Rerun**. Perform these operations in emergency only. These operations take effect only on the current instance for one time. Select **Delete Dependencies** to delete dependencies of the current instance. You can perform this operation so that you can start the current instance when the ancestor instances fail and the current instance does not depend on the data of the ancestor instances.
Freeze	Freeze the instance if it is in the Running state.

Operation	Description
Unfreeze	Unfreeze the instance after it is frozen. • If the instance is not run, DataWorks automatically runs this instance after its ancestor instances are run. • If all the ancestor instances have been run, the instance is directly set to Failed. You must manually rerun the instance.

7.4.3. Manage retroactive instances

DataWorks runs retroactive instances to generate retroactive data for auto triggered nodes. You can manage retroactive instances. For example, you can view the running status of instances, and stop, rerun, or unfreeze instances.

Limits

- When DataWorks generates retroactive data in a period of time for a node, if one instance of the node fails on a day, the retroactive instance for that day is also set to Failed. DataWorks will not run instances of this node for the next day. To sum up, DataWorks runs instances of a node on a day only when all its instances of the previous day are successful.
- For a self-dependent auto triggered node, if the first instance for which retroactive data needs to be generated has a last-cycle instance on the previous day but the last-cycle instance is not run, the retroactive instance cannot be triggered. If the first instance for which retroactive data needs to be generated does not have a last-cycle instance on the previous day, the retroactive instance is directly triggered.
- DataWorks generates alerts only for auto triggered node instances.
- If an auto triggered node has an instance in the Running state, its retroactive or test instance can start to run only after this auto triggered node instance is run.
- If both an auto triggered node instance and a retroactive instance are running for a node at the same time, you must stop the retroactive instance to ensure proper running of the auto triggered node instance.

Go to the page for configuring retroactive instances

1. Log on to the DataWorks console.
2. Click ☰ in the upper-left corner and choose **All Products > Operation Center**.
3. In the left-side navigation pane, choose **Cycle Task Maintenance > Cycle Task**.
4. On the page that appears, click the rightward arrow in the middle to show the Actions column in the node list. Find the target node, click **Patch Data** in the Actions column, and then select a mode for generating retroactive data.

   You can also right-click the target node in the directed acyclic graph (DAG), move the pointer over **Run**, and then select a mode for generating retroactive data.

Generate retroactive data for the current node

1. Find the target node, click **Patch Data** in the Actions column, and then select **Current Node Retroactively**.
2. In the **Patch Data** dialog box, set the parameters.

Parameter	Description
Retroactive Instance Name	The name of the retroactive instance. DataWorks automatically generates a name, which can be changed.
Data Timestamp	The data timestamp of the retroactive instance.
Node	The name of the node for which you want to generate retroactive data. You cannot change the name.
Parallelism	Specifies whether to generate multiple retroactive instances at a time. ◦ If you select **Disable**, only one retroactive instance is generated. The retroactive instance is run multiple times based on the data timestamp in sequence. ◦ If you select **2 Parallel Groups**, **3 Parallel Groups**, **4 Parallel Groups**, or **5 Parallel Groups**, multiple retroactive instances are generated. The retroactive instances are run based on the data timestamp in parallel or in sequence. ■ If the number of days in the data timestamp is smaller than the number of parallel groups, the retroactive instances are run in parallel. For example, the data timestamp is from January 11 to January 13, and you select 4 Parallel Groups. In this case, three retroactive instances are generated for each day in the data timestamp, and the three retroactive instances are run in parallel. ■ If the number of days in the data timestamp is greater than the number of parallel groups, some instances must be run multiple times in sequence while others are run in parallel. For example, the data timestamp is from January 11 to January 13, and you select 2 Parallel Groups. In this case, two retroactive instances are generated. They are run in parallel for once, and one of them must be run for a second time.

3. Click **OK**.

Generate retroactive data for the current and descendant nodes

1. Find the target node, click **Patch Data** in the Actions column, and then select **Current and Descendant Nodes Retroactively**.
2. In the **Patch Data** dialog box, set the parameters, including **Nodes**.

Parameter	Description
Retroactive Instance Name	The name of the retroactive instance. DataWorks automatically generates a name, which can be changed.
Data Timestamp	The data timestamp of the retroactive instance.

Parameter	Description
Parallelism	Specifies whether to generate multiple retroactive instances at a time. ○ If you select **Disable**, only one retroactive instance is generated. ○ If you select **2 Parallel Groups**, **3 Parallel Groups**, **4 Parallel Groups**, or **5 Parallel Groups**, multiple retroactive instances are generated.
Nodes	The nodes for which you want to generate retroactive data. You can set **Node Name** and **Node Type** to filter nodes.

3. Click **OK**.

Generate retroactive data for a large number of nodes

1. Find the target node, click **Patch Data** in the Actions column, and then select **Mass Nodes Retroactively**.
2. In the **Patch Data** dialog box, set the parameters. The following table describes the parameters for generating retroactive data for a large number of nodes.

Parameter	Description
Retroactive Instance Name	The name of the retroactive instance. DataWorks automatically generates a name, which can be changed.
Data Timestamp	The data timestamp of the retroactive instance. **Note** We recommend that you do not set this parameter to a long range. Otherwise, the retroactive instance may be delayed due to insufficient resources.
Parallelism	Specifies whether to generate multiple retroactive instances at a time. ○ If you select **Disable**, only one retroactive instance is generated. ○ If you select **2 Parallel Groups**, **3 Parallel Groups**, **4 Parallel Groups**, or **5 Parallel Groups**, multiple retroactive instances are generated.
Nodes	○ If you select the **Current Node** check box, retroactive instances are generated for the current node and its descendant nodes. ○ If you clear the **Current Node** check box, the current node is dry-run and retroactive instances are generated for its descendant nodes.
Workspaces	The workspaces of the nodes for which DataWorks needs to generate retroactive data. Select workspaces under **Available Workspaces** and add them to **Selected Workspaces**. Fuzzy match is supported when you search for workspaces under Available Workspaces.

Parameter	Description
Node Whitelist	The nodes outside the selected workspaces, for which DataWorks needs to generate retroactive data. **Note** You can search for nodes by node ID.
Node Blacklist	The nodes inside the selected workspaces, for which DataWorks does not need to generate retroactive data. **Note** You can search for nodes by node ID.

3. Click **OK**.

Manage retroactive instances in the instance list

Operation	Description
Filter	Find required instances by specifying filter conditions. You can search for instances by node name or node ID and set the following parameters to filter instances: **Retroactive Instance Name**, **Node Type**, **Owner**, **Run At**, **Data Timestamp**, **Engine type**, **Engine instance**, **Baseline**, and **My Nodes**. **Note** By default, the data timestamp is set to the previous day of the current day.
DAG	View the DAG of the instance. You can view the running result of the instance in the DAG.
Stop	Stop the instance. You can stop instances only when they are in the **Pending (Resource)** or **Running** state. After you perform this operation, the instance enters the **Failed** state.
Rerun	Rerun the instance.
Rerun Descendant Nodes	Rerun the descendant instances of the instance.
Freeze	Pause the scheduling of the instance.
Recover	Resume the scheduling of the instance after it is paused.
View Lineage	View the lineage of the instance.

Manage retroactive instances in a DAG

Click the name of an instance or **DAG** in the Actions column to view the DAG of the instance. In the DAG, you can right-click the instance to perform related operations.

> **Note** After you click **Refresh** in the upper-right corner, the DAG of the instance is refreshed, but the operational logs are not.

Operation	Description
Show Ancestor Nodes or Show Descendant Nodes	Show ancestor or descendant instances at one or more levels. If a workflow contains three or more instances, the DAG displays only the current instance and hides its ancestor and descendant instances.
View Runtime Log	View the operational logs of the instance if it is in the Running, Successful, or Failed state.
View Code	View the code of the instance.
Edit Node	Go to the **DataStudio** page to modify the node to which the instance belongs.
View Lineage	View the lineage of the instance.
Stop	Stop the instance. You can stop instances only when they are in the **Pending (Resource)** or **Running** state. After you perform this operation, the instance enters the **Failed** state.
Rerun	Rerun the instance if it is in the Failed or an abnormal state.
Rerun Descendant Nodes	Rerun all the descendant instances of the instance.
Set Status to Successful	Set the status of the instance to Successful and run its descendant instances as scheduled. You can perform this operation if an instance fails. **Note** This operation applies only to failed instances but not workflows.
Emergency Operations	Perform emergency operations in emergency only. Emergency operations take effect only on the current instance for one time. Select **Delete Dependencies** to delete dependencies of the current instance. You can perform this operation so that you can start the current instance when the ancestor instances fail and the current instance does not depend on the data of the ancestor instances.
Freeze	Pause the scheduling of the instance.
Unfreeze	Resume the scheduling of the instance after it is paused.

7.4.4. Manage test instances

Test instances are generated when you test auto triggered nodes. You can manage test instances.

Go to the Test Instance page

1. Log on to the DataWorks console.
2. Click ☰ in the upper-left corner and choose **All Products > Operation Center**.
3. In the left-side navigation pane, choose **Cycle Task > Test Instance**. The Test Instance page appears, where you can view the list and directed acyclic graphs (DAGs) of test instances.

Manage test instances in the instance list

You can manage test instances in the instance list. For example, you can rerun, freeze, or unfreeze instances, set the status of instances to Successful, view the lineage of instances, and check operational logs.

Operation	Description
Filter	Find required instances by specifying filter conditions. You can search for instances by node name or node ID and set the following parameters to filter instances: **Node Type**, **Owner**, **Run At**, **Data Timestamp**, **Status**, **Region**, **Engine type**, **Engine instance**, **Baseline**, **My Nodes**, **Tested by Me Today**, and **Frozen**.
Stop	Stop the instance. You can stop instances only when they are in the **Pending (Resource)** or **Running** state. After you perform this operation, the instance enters the **Failed** state.
Rerun	Rerun the instance. After the instance is rerun, its descendant instances will be run as scheduled. Perform this operation if an instance fails or it is not run as scheduled. **Note** This operation applies only to instances in the **Pending (Ancestor)**, **Successful**, or **Failed** state.
More operations	Click **More** in the Actions column and then select an operation. The operations include **Set Status to Successful**, **Freeze**, **Unfreeze**, **View Lineage**, and **View Runtime Log**.
Batch operations	Select multiple instances and click a button in the lower part of the instance list to perform batch operations. The buttons include **Stop**, **Rerun**, **Set Status to Successful**, **Freeze**, and **Unfreeze**.

Manage test instances in a DAG

Click the name of an instance or **DAG** in the Actions column to view the DAG of the instance. In the DAG, you can right-click the instance to perform related operations.

Operation	Description
View Runtime Log	View the operational logs of the instance if it is in the Running, Successful, or Failed state.
View Code	View the code of the instance.
Edit Node	Go to the **DataStudio** page to modify the node to which the instance belongs.

Operation	Description
View Lineage	View the lineage of the instance.
Stop	Stop the instance. You can stop instances only when they are in the **Pending (Resource)** or **Running** state. After you perform this operation, the instance enters the **Failed** state.
Rerun	Rerun the instance. After the instance is rerun, its descendant instances will be run as scheduled. Perform this operation if an instance fails or it is not run as scheduled. **Note** This operation applies only to instances in the **Pending (Ancestor)**, **Successful**, or **Failed** state.
Set Status to Successful	Set the status of the instance to **Successful** and run its descendant instances as scheduled. You can perform this operation if an instance fails. **Note** This operation applies only to failed instances but not workflows.
Freeze	Pause the scheduling of the instance.
Unfreeze	Resume the scheduling of the instance after it is frozen. • If the instance is not run, DataWorks automatically runs this instance after its ancestor instances are run. • If all the ancestor instances have been run, the instance is directly set to Failed. You must manually rerun the instance.

7.5. Manually triggered node O&M

7.5.1. Manage manually triggered nodes

Manually triggered nodes are nodes whose scheduling type is set to manual before they are committed to the scheduling system.

Note After a manually triggered node is committed to the scheduling system, it will run only after it is manually triggered.

Manage manually triggered nodes in the node list

The manually triggered node list displays manually triggered nodes that are committed.

Operation	Description

Operation	Description
Filter	Find required nodes by specifying filter conditions. You can search for nodes by node name and set the **Owner**, **My Nodes**, and **Modified Today** parameters to filter nodes. **Note** When you search for nodes by node name, the search result is affected by other filter conditions you specified. Only nodes that meet both the specified search condition and other filter conditions are returned in the search result.
DAG	Click **DAG** in the Actions column of a node to view the directed acyclic graph (DAG) of the node. You can view the node information, such as the code and lineage, in the DAG.
Run	Click **Run** in the Actions column of a node to run the node. Manually triggered node instances are generated.
View Instances	Click **View Instances** in the Actions column of a node to go to the **Manual Instance** page, where you can view the instances of the node.
More operations	Choose **More > Change Owner** in the Actions column of a node to change the owner of the node.
Batch operations	Select multiple nodes and click **Change Owner** in the lower part of the page to change the owner of the nodes.

Manage manually triggered nodes in a DAG

Click the name of a node or **DAG** in the Actions column to view the DAG of the node. In the DAG, you can right-click the node to perform related operations.

Operation	Description
View Code	View the code of the node.
Edit Node	Go to the **DataStudio** page to modify the node.
View Instances	View the instances of the node.
View Lineage	View the lineage of the node.
Run	Run the node. After you click Run, manually triggered node instances are generated.
Change Resource Group	Change the resource group where the node is run.

7.5.2. Manage manually triggered node instances

DataWorks generates manually triggered node instances from manually triggered nodes. Manually triggered nodes do not have node dependencies. They must be run manually.

> **Notice** DataWorks generates alerts only for auto triggered node instances when they fail to be run.

Go to the page for managing manually triggered node instances

1. Log on to the DataWorks console.
2. Click ☰ in the upper-left corner and choose **All Products > Operation Center**.
3. In the left-side navigation pane, choose **Trigger Task Maintenance > Trigger Instance**. The Trigger Instance page appears, where you can view the list and directed acyclic graphs (DAGs) of manually triggered instances.

Manage manually triggered node instances in the instance list

Operation	Description
Filter	Find required instances by specifying filter conditions. You can search for instances by instance name and set the **Owner**, **Data Timestamp**, and **Run At** parameters to filter instances.
DAG	Click **DAG** in the Actions column to view the DAG of the instance. You can view the running result of the instance in the DAG.
Stop	Click **Stop** to stop the instance if it is in the Running state.
Rerun	Rerun the instance.
Batch stop	Select multiple instances and click **Stop** to stop the selected instances at a time.

Manage manually triggered node instances in a DAG

Click the name of an instance or **DAG** in the Actions column to view the DAG of the instance. In the DAG, you can right-click the instance to perform related operations.

> **Note** A manually triggered node instance does not have dependencies, so only the current instance appears in the DAG.

Operation	Description
View Runtime Log	View the operational logs of the instance if it is in the Running, Successful, or Failed state.
View Code	View the code of the instance.
Edit Node	Go to the **DataStudio** page to modify the node to which the instance belongs.
View Lineage	View the lineage of the instance.
Stop	Stop the instance.

Operation	Description
Rerun	Rerun the instance if it is in the Failed or an abnormal state.

7.6. MaxCompute engine O&M

DataWorks Operation Center allows you to view the jobs, quota groups, and projects of MaxCompute.

Prerequisites

A MaxCompute engine is added on the **Project Management** page.

Go to the MaxCompute page

1. Log on to the DataWorks console.
2. Click ☰ in the upper-left corner and choose **All Products > Operation Center**.
3. In the left-side navigation pane, choose **Engine Maintenance > MaxCompute**. The **Job Queue** tab appears by default.

   You can view the jobs, quota groups, and projects of MaxCompute.

View jobs

On the **Job Queue** tab, you can view the total number of jobs and numbers of jobs in the **Running**, **Waiting for Resources**, and **Initializing** states.

You can set the **QuotaGroup** and **Project** parameters to filter jobs. You can view the following parameters of each job: **Instance**, **Execution Time**, **CPU Usage (Minimum/Maximum)**, **Memory Usage (Minimum/Maximum)**, **Priority**, **Node**, **Submitted By**, **Project**, **Type**, **Quota Group**, **Cluster**, **Status**, and **Start Time**.

View quota groups

On the **MaxCompute** page, click the **Quotas** tab.

On the **Quotas** tab, you can view the following parameters of each quota group: **Project**, **Quota Group**, **Default**, **Cluster**, **Minimum CPU (cores)**, **Maximum CPU (cores)**, **Minimum Memory (bytes)**, **Maximum Memory (bytes)**, and **Projects**.

You can click the name of a quota group to view the resource usage information about the quota group.

View projects

On the **MaxCompute** page, click the **Projects** tab.

On the **Projects** tab, you can view the following parameters of each project: **Project**, **Owner**, **Cluster**, **Quota Group**, **Storage Used**, **Storage Quota**, **Storage Usage**, and **Files**.

7.7. Monitor

7.7.1. Overview

The Monitor module is a node monitoring and analysis system of DataWorks. Based on monitoring rules and node running status, the Monitor module determines whether, when, and how to trigger an alert, and whom an alert is sent to. It automatically selects the most appropriate alerting time, notification methods, and recipients.

The Monitor module provides you with the following benefits:

- Improves your efficiency on configuring monitoring rules.
- Prevents invalid alerts from bothering you.
- Automatically covers all important nodes for you.

General monitoring systems cannot meet the requirement of DataWorks. The reasons are as follows:

- DataWorks has numerous nodes, so it is difficult for you to find out the nodes to be monitored. Some DataWorks businesses have a large number of nodes, and dependencies between the nodes are complex. Even if you know the most important node, it is difficult to find all ancestor nodes of the node and monitor them all. In this case, if you simply monitor all nodes, a large number of invalid alerts may be generated. In consequence, you may miss those useful alerts.
- The alerting method varies with nodes. For example, some monitoring tasks require the relevant nodes to run for more than one hour before triggering alerts, while other monitoring tasks require the relevant nodes to run for more than two hours. It is extremely complex to set a monitoring node for each node, and it is difficult to predict the alert threshold value for each node.
- The alerting time varies with nodes. For example, an alert for an unimportant node can be sent after you start working in the morning. An alert for an important node needs to be sent immediately when an error occurs. General monitoring systems cannot tell the importance of each node.
- Different alerts require different operations to turn off.

The Monitor module provides comprehensive monitoring and alerting logic. You only need to provide the node name of your business. Then, the Monitor module automatically monitors the entire process of your node and creates standard alert triggers for the node. In addition, you can customize alerting triggers by completing basic settings.

Currently, the Monitor module has been used for monitoring all important businesses of Alibaba Group. Its full-path monitoring function guarantees the overall data output of all important businesses of Alibaba Group. In addition, it supports analyzing ancestor and descendant node paths to promptly detect risks and provide O&M advice for business departments. These functions of the Monitor module have guaranteed the long-term high stability of businesses in Alibaba Group.

7.7.2. Feature description

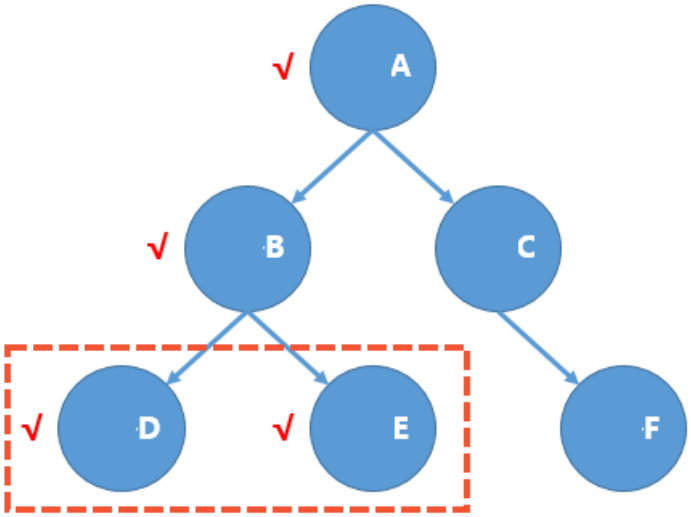
7.7.2.1. Baseline alert and event alert

This topic describes the functional logic of baseline alerts and event alerts from the aspects of monitoring scope, node capturing, alert object judgment, alerting time judgment, notification methods, and alert escalation.

Monitoring scope

A baseline is a management unit of a group of nodes, that is, a node group. You can specify nodes to monitor in a baseline.

After a baseline is monitored, all nodes of the baseline and its ancestor nodes are monitored. The Monitor module does not monitor all nodes by default. A node is monitored only when it has descendant nodes that are added to a monitoring baseline. If no descendant nodes are added to a monitoring baseline, the Monitor module does not report any alert even if the node fails.



As shown in the preceding figure, assume that DataWorks has only six nodes, and nodes D and E belong to a monitoring baseline. Nodes D and E and all their ancestor nodes are monitored by the Monitor module. That is, any error or slowdown on node A, B, D, or E will be detected by the Monitor module. However, nodes C and F are not monitored by the Monitor module.

Node capturing

After the nodes to be monitored are specified, if a monitored node incurs an exception, the Monitor module generates an event. All alert decisions are based on the analysis of this event. Two types of node exceptions are available. You can choose **Events > Event Type** to view them.

- **Error**: indicates that a node fails to run.
- **Slow**: indicates that the running time of a node is significantly longer than the average running time of the node in the past periods.

> **Note** If a node times out and then encounters an error, two events are generated.

Alerting time judgment

Buffer, an important concept in the Monitor module, refers to the maximum time period that a node can be delayed. The latest start time of a node is obtained by subtracting the average uptime from the baseline time.

The baseline time of baseline A is 05:00, you must set the latest start time of node E to 04:10. This time is calculated by subtracting the average uptime of node F (20 minutes) and node E (30 minutes) from the baseline time 05:00. This time is also the latest completion time of node B in baseline A.

To ensure that the baseline time of baseline B is 06:00, you must set the latest completion time of node B to 04:00. This time, which is earlier than 04:10, is calculated by subtracting the average uptime of node D (2 hours) from the baseline time 06:00. To meet the baseline time of both baseline A and baseline B, you must set the latest completion time of node B to 04:00.

The latest completion time of node A is 02:00, which is calculated by subtracting the average uptime of node B (2 hours) from 04:00. The latest start time of node A is 01:50, which is calculated by subtracting the average uptime of node A (10 minutes) from 02:00. If node A fails to run before 01:50, it is probable that baseline A is broken.

If node A fails to run at 01:00, its buffer is 50 minutes, which is the difference between 01:00 and 01:50. As demonstrated in this example, buffer reflects the degree of caution for a node exception.

Baseline alert

Baseline alerting is an additional feature developed for baselines that are enabled. Each baseline must provide an alert buffer and committed time. Baseline alerting is the action of notifying the preset alert recipient three times at the interval of 30 minutes when the baseline completion time estimated by the Monitor module exceeds the alert buffer.

Notification method

Currently, baseline alerts are sent to the baseline owner by default. On the **Alert Triggers** page, you can find **Global Baseline Alert Trigger**, click **View Details**, and change the alert trigger method and the alerting action.

Gantt chart function

The Gantt chart function reflects the key path of a node. The function is provided by the **Baseline Instances** module of Monitor.

> **Note** The key path is the slowest upstream link that causes the node to be completed at this time point.

7.7.2.2. Custom alert trigger

Alert trigger customization is a lightweight monitoring function of the Monitor module.

You can customize all monitoring alert triggers by setting the following parameters:

- Objects: You can specify nodes, baselines, and workspaces as objects.
- Trigger Condition: Valid values include Completed, Uncompleted, Error, Uncompleted Cycle, and Overtime.
- Notification Method: Valid values include SMS and Email.
- Maximum Alerts: This parameter indicates the maximum number of alert reporting times. If the number of alerting times exceeds the preset threshold, no alerts are generated.
- Minimum Alert Interval: This parameter indicates the minimum time interval at which DataWorks reports alerts.
- Quiet Hours: This parameter indicates the specified period during which no alerts are reported.
- Recipient: This parameter indicates the person who receives alerts. You can set this parameter to the node owner or another recipient.

A monitoring rule uses the following five alert trigger conditions: Completed, Uncompleted, Error, Uncompleted Cycle, and Overtime.

- **Completed**

A completion alert can be set for nodes, baselines, and workspaces. Once all nodes of the preset objects are completed, the completion alert is reported. If you set a completion alert for a baseline, the alert is reported when all nodes of the baseline are completed.

- **Uncompleted**

  You can set alerts for nodes, baselines, or workspaces that are not completed at a certain time point. For example, if you require that a baseline be completed at 10:00, an alert containing a list of uncompleted nodes is reported once a node in the baseline is not completed at the specified time.

- **Error**

  An error alert can be set for nodes, baselines, and workspaces. Once a node has an error, an alert containing detailed node error information is sent to the recipient.

- **Uncompleted Cycle**

  For the monitoring rules of hourly scheduled nodes, you can separately specify the uncompleted time points in different periods.

- **Overtime**

  An overtime alert can be set for nodes, baselines, and workspaces. Once a monitored node of the preset object is not completed within the specified time, an alert is reported.

7.7.3. Instructions

7.7.3.1. Baseline instances

On the Baseline Instances page, you can view the information about a baseline.

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Operation Center**. The **Operation Center** page appears.
3. On the Operation Center page, choose **Monitor > Baselines** in the left-side navigation pane. On the **Baselines** page, create a baseline. For more information abut how to create a baseline, see Create a baseline.

   > **Note** If you have created a baseline, skip this step and directly go to the **Baseline Instances** page to perform subsequent operations.

4. In the left-side navigation pane, choose **Monitor > Baseline Instances**. The **Baseline Instances** page appears. On this page, you can search for baseline instances by condition, such as the data timestamp, owner name or ID, event ID, workspace, and baseline name. You can also click **View Details**, **Handle**, and **View Gantt Chart** in the Actions column to perform corresponding operations on a baseline.

   > **Note** After creating a baseline, you must enable the baseline so that a baseline instance can be generated.

   A baseline can be in the one of the following four states:

   - **Normal**: All nodes in the baseline are completed before the alerting time.
   - **Alerting**: One or more nodes in the baseline are not completed at the alerting time but the committed completion time has not arrived.

- **Overtime**: One or more nodes in the baseline are not completed at the committed completion time.
- **Others**: All nodes in the baseline are paused or the baseline is not associated with any node.

You can click **View Details**, **Handle**, and **View Gantt Chart** in the Actions column to perform corresponding operations on a baseline.

- **View Details**: Click **View Details** to go to the **Baseline Instance Details** page.

  On the **Baseline Instance Details** page, you can view the general information, critical path, baseline instance information, history graph, and relevant events.

  > **Note**
  > - In the preceding figure, the data timestamp is `one day before the system time`.
  > - When you create a baseline, you can select **By the Day Interval** or **By the Hour Interval**. The **Cycle** parameter appears as the advanced settings of the **Committed Time** parameter only when you select By the Hour Interval.

- **Handle**: Click Handle to pause the alert generated by the baseline when the baseline is being handled.
- **View Gantt Chart**: Click **View Gantt Chart** to view the critical paths of nodes.

7.7.3.2. Baselines

You can create and define baselines on the Baselines page.

Create a baseline

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Operation Center**. The **Operation Center** page appears.
3. On the Operation Center page, choose **Monitor > Baselines** in the left-side navigation pane.
4. On the **Baselines** page, click **Create Baseline** in the upper-right corner.

   > **Note** Currently, only workspace administrators can create baselines.

5. In the **Create Baseline** dialog box that appears, set the parameters and click **OK**.

Parameter	Description
Baseline Name	The name of the baseline.
Workspace	The workspace of the node associated with the baseline.
Owner	The name or ID of the owner.

Parameter	Description
Recurrence	Specifies whether the baseline detects nodes by day or hour. ○ **By the Day Interval**: Select this option for nodes scheduled by day. ○ **By the Hour Interval**: Select this option for nodes scheduled by hour.
Nodes	○ **Node**: the node associated with the baseline. Enter the name or ID of a node, and then click the icon on the right to add the node. You can add multiple nodes. ○ **Workflow**: the workflow associated with the baseline. Enter the name or ID of a workflow, and then click the icon on the right to add the workflow. We recommend that you only add the last node of a workflow instead of all nodes.
Priority	The priority of the baseline. A baseline with a higher priority is scheduled first. Currently, the only available priority value is 1.
Estimated Completion Time	The completion time of the node estimated based on the average running time of the node during previous scheduling. If no historical data is available, the message **The completion time cannot be estimated due to a lack of historical data** appears.
Committed Time	The time point when a node should be completed. An alert is triggered if the node is not completed until the time point obtained by subtracting the alert margin threshold from the committed completion time.
Margin Threshold	The interval before an alert is triggered. For example, set Committed Time to 3:30 and Margin Threshold to 10 minutes. An alert is triggered if the node is not completed at 3:20. Assume that the average running time of the node is 30 minutes. If the node is not started at 2:50, an alert is triggered. **Note** The average running time of a node can be calculated based on the data of the last 15 days.

6. After a baseline is created, click **Enable** in the Actions column to enable the baseline.

   You can click **View Details**, **Change**, **Enable**, Disable, or **Delete** in the Actions column to perform the corresponding operation on a baseline.

   - **View Details**: Click **View Details** to view the basic information about the baseline.
   - **Change**: Click **Change** to modify the baseline.
   - **Enable** or Disable: Click Enable or Disable to enable or disable the baseline. A baseline instance can be generated only when the corresponding baseline is enabled.
   - **Delete**: Click **Delete** to delete the baseline.

Add a node to a baseline

By default, all nodes in the production environment are in the default baseline of each workspace. When you create a baseline, you actually move nodes from the default baseline to the baseline that you create.

> **Note** A node must belong to a baseline, so you cannot directly remove nodes from the default baseline. Instead, you can create a baseline to move nodes from the default baseline to the new baseline. When you delete a baseline that you create, you actually move the nodes in the baseline to the default baseline.

To change the baseline of a node, perform one of the following operations:

- On the **Baselines** page, click **Create Baseline** in the upper-right corner. Then, create a baseline by following the instructions in the Create a baseline section.
- In the left-side navigation pane, choose Nodes > **Recurring**. On the page that appears, find the node and choose **More > Add to Baseline** in the Actions column.

7.7.3.3. Events

On the Events page, you can view all events related to slowdown or errors.

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Operation Center**.
3. On the Operation Center page, choose **Monitor > Events** in the left-side navigation pane. The **Events** page appears.

   You can search for events by condition, such as the event owner, time when an event was detected, event status, event type, and name or ID of a node or node instance.

   In the search results, each event is displayed in a row and associated with a node that encounters errors. The worst baseline indicates a baseline with the minimum margin among the baselines affected by an event.

   - Click **View Details** in the Actions column of an event. You can view the event occurrence time, alert time, clearance time, historical runtime logs of the node, and detailed node logs.

     You can assign an alert recipient. After you click **View Alerts**, the alert details page corresponding to the event appears. Affected baselines are all descendant baselines affected by the node associated with the event. You can observe descendant baselines and the impact on these baselines and analyze node logs to identify the causes of the event.
   - If you click **Handle**, DataWorks records the event handling operation and pauses the alert for the event when the event is being handled.
   - If you click **Ignore**, DataWorks keeps the event ignorance record and permanently stops the alert for the event.

7.7.3.4. Alert triggers

This topic describes how to customize alert triggers on the Alert Triggers page.

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Operation Center**.

3. On the Operation Center page, choose **Monitor > Alert Triggers** in the left-side navigation pane. The **Alert Triggers** page appears.
4. On the Alert Triggers page, click **Create Custom Trigger** in the upper-right corner.
5. In the **Create Custom Trigger** dialog box that appears, set relevant parameters.

Parameter	Description
Trigger Name	The name of the custom alert trigger.
Object Type	The granularity of monitored objects. Valid values: **Node** and **Workflow**.
Objects	The monitored object. Enter the name or ID of a node or workflow and click the icon on the right to add the object.
Trigger Condition	The condition for triggering an alert. Valid values: **Completed**, **Uncompleted**, **Error**, **Uncompleted Cycle**, and **Overtime**.
Maximum Alerts	The maximum number of alert reporting times. If the number of alert reporting times exceeds the preset threshold, no alerts are reported.
Minimum Alert Interval	The minimum time interval at which DataWorks reports an alert.
Quiet Hours	The specified period during which no alerts are reported.
Notification Method	The method of reporting alerts. Valid values: **Email** and **SMS**.
Recipient	The person who receives alerts. You can set this parameter to the node owner or another recipient.
DingTalk Chatbot	The DingTalk chatbot to receive alerts.

6. Click **OK** to create the alert trigger.

   On the **Alert Triggers** page, you can click **View Details** in the Actions column of an alert trigger to view the details of the alert trigger.

7.7.3.5. Alert information

You can view all alerts on the Alerts page.

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Operation Center**.
3. On the Operation Center page, choose **Monitor > Alerts** in the left-side navigation pane. The **Alerts** page appears.

   You can search for alerts by condition, such as the alert trigger ID or name, recipient, alert time, notification method, and alert trigger type.

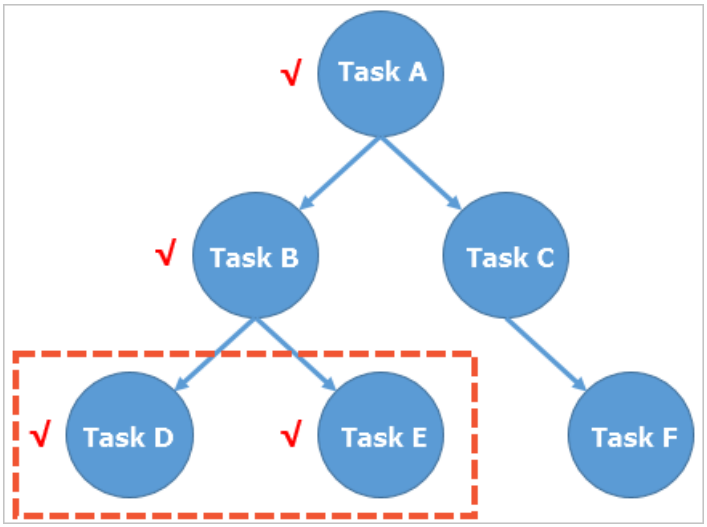
   You can also view alert details such as the notification method and status. To view more alert details, find the target alert and click **View Details** in the Actions column.

7.7.4. FAQ

This topic describes the FAQ about the Monitor service.

What can I do if I do not need to receive alerts for a node?

After you create and enable a monitoring baseline, the Monitor service monitors all nodes in the baseline and their ancestor nodes. If a node in the baseline or an ancestor node of the baseline affects data generation of the monitored nodes in the baseline, the Monitor service reports an alert to the node owner.



As shown in the preceding figure, assume that DataWorks has only six nodes, and Nodes D and E belong to a monitoring baseline. The Monitor service monitors Nodes D and E and all their ancestor nodes. Namely, the Monitor service will detect any error or slowdown on Node A, B, D, or E. Nodes C and F are not monitored by the Monitor service.

Nodes A and B are ancestor nodes of Nodes D and E and may affect data generation of the monitored nodes in the baseline. When an error or slowdown occurs on Node A or B, the Monitor service reports an alert to the node owner.

If you do not need to receive alerts for a node, use the following methods:

- If the owners of Nodes D and E do not need to receive alerts, contact the baseline owner to remove Nodes D and E from the baseline.
- If the owner of Node A or B does not need to receive alerts, contact the owners of Nodes D and E to delete the dependency of Nodes D and E on Node A or B.

Why is no alert reported for a baseline in the Overtime state?

Baseline monitoring is controlled by the baseline switch and enabled for nodes. If all nodes are running properly, no alert will be triggered even in the Overtime state. This is because all the nodes are running properly and the Monitor service cannot determine which node has an error. Overtime is a baseline state, indicating that a node is still not completed after the committed time.

If the baseline still enters the Overtime state when all nodes are running properly, consider the following reasons:

- The baseline time is not properly set.
- The node dependency is not properly configured.

Can I disable the Monitor service from reporting an alert for a node that slows down?

The Monitor service reports a node slowdown alert only when a node meets both of the following conditions:

- The node is an ancestor node of an important baseline.
- Compared with its historical performance, the node does slow down.

You can view the descendant baseline affected by the node on the **Event Management** page. Then, you can confirm the impact with the party whose monitoring baseline contains descendant nodes of your node.

- If the node slowdown has a minor impact, you can ignore the alert.
- If the node slowdown has a major impact, maintain your node properly.

Why do I fail to receive an alert for an error node?

The Monitor service reports an alert only for specified nodes when an error occurs. An alert is reported for an error node only when the node meets one of the following conditions:

- The node is an ancestor node of a baseline that has been enabled.
- An alert trigger has been customized.

What can I do if I receive an alert at night?

1. Log on to the DataWorks console.
2. On the **DataStudio** page, click ☰ in the upper-left corner and choose **All Products > Operation Center**.
3. In the left-side navigation pane, choose **Alarm > Event Management**.
4. On the **Event Management** page, disable the event alert. Disable the event alert in one of the following ways:
   - Handle the event that triggers an alert.
     a. Find the target event and click **Handle** in the Actions column.
     b. In the **Handle Event** dialog box, set the **Handling Time** parameter.
     c. Click **OK**.

     > **Note** DataWorks records the event handling operation and pauses alerting for the event when the event is being handled.

   - Ignore the event that triggers an alert.
     a. Find the target event and click **Ignore** in the Actions column.
     b. In the **Ignore Event** message, click **OK**.

     > **Note** DataWorks records the event ignoring operation and permanently stops alerting for the event.

8.Security Center

8.1. Overview

Security Center provides flexible permission management features. It allows you to request permissions and handle permission requests on the graphical user interface (GUI), and view and manage permissions. Security Center not only improves data security but also facilitates data permission management.

Log on to the DataWorks console. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Security Center**. The **Security Center** page appears.

Security Center consists of the following modules: **My Permissions**, **Authorizations**, and **Approval Center**.

Currently, Security Center provides the following features:

- Self-service permission request: Users can select the required tables to quickly initiate a permission request online. This online request mode is more efficient than the original mode in which users need to contact administrators offline.
- Permission management: Administrators can view the users who have permissions on database tables and revoke permissions as required. Users can also revoke unnecessary permissions themselves.
- Permission request approval: Before granting permissions to users, administrators approve permission requests initiated by users. This implements a visual and process-based permission management system, and supports reviewing the approval process.

In Security Center, you can view permissions on all the tables in an organization, request and revoke table permissions, and approve or reject permission requests.

Each operation in Security Center applies to all the workspaces of a tenant in standard mode and basic mode.

8.2. My Permissions

On the My Permissions page, you can view your table and field permissions in a workspace, and request or revoke table and field permissions.

View table and field permissions

1. Move the pointer over the DataWorks icon in the upper-left corner, and click **Security Center**. In the left-side navigation pane, click **My Permissions**. The **Table** tab appears.
2. On this tab, you can select a workspace and specify the environment (for a workspace in standard mode) to view all the tables of the workspace in the specified environment. You can also enter a table name in the search box to search for required tables in fuzzy match mode.

   You can view the names and owners of tables in the workspace, view your permissions on the tables, and request or revoke table and field permissions.

Request table and field permissions

1. Select the tables and fields on which you want to request permissions.
   - Request permissions on a table or some fields in the table

     Select the required fields on which you have no permissions in a table and choose **More > Request Permission** in the Actions column.

Alternatively, choose **More > Request Permission** in the Actions column for a table without selecting any fields to request permissions on all the fields in the table.

> **Note** You can request permissions on fields only in a workspace with LabelSecurity enabled. If LabelSecurity is disabled for a workspace, you can request permissions only on tables in this workspace.

- Request permissions on multiple tables and fields

  Select all the required tables and fields and click **Request Permission**.

> **Note** You can also click **Request Permission** without selecting any tables or fields, and then select the required tables and fields in the **Table Permission Request** dialog box.

2. Set the parameters in the **Table Permission Request** dialog box.

Parameter	Description
Workspace	The name of the workspace, which is automatically entered based on the information you specified on the My Permissions page. You can change the workspace as required.
Environment	The environment of the workspace.
MaxCompute Project	The name of the MaxCompute project.
Grant To	The account for which you request permissions. You can request permissions for the current account or a production account of another workspace you joined.
Reason for Request	The reason why you request permissions.
Objects Requested	The tables on which you request permissions. The tables that you select on the previous page are displayed. You can add tables or delete existing tables as required.

3. After the configuration is completed, click **Submit**. If you do not want to request the permissions, click **Cancel**.

Revoke permissions

You can revoke table and field permissions.

- Revoke field permissions

> **Note**
> - You can revoke permissions on fields only in a workspace with LabelSecurity enabled.
> - To revoke permissions on all the fields in a table, you can directly revoke the permissions on the table.

i. Choose **More > Revoke Field Permission** in the **Actions** column for the table on which you want to revoke permissions.

ii. In the **Revoke Field Permission** dialog box, select the fields on which you want to revoke permissions.

iii. Click **OK**.

- Revoke table permissions

i. Choose **More > Revoke Permission** in the **Actions** column for the table on which you want to revoke permissions.

ii. In the **Revoke Permission** dialog box, select the permissions you want to revoke.

iii. Click **OK**.

8.3. Authorizations

On the Authorizations page, a workspace administrator can view the accounts that have permissions on tables and fields in each workspace, and revoke unnecessary table and field permissions.

You can move the pointer over the DataWorks icon in the upper-left corner, and click **Security Center**. In the left-side navigation pane, click **Authorizations**. On the **Table** tab that appears, you can view and search for tables in workspaces of the current organization.

On the Table tab, you can select a workspace and specify the environment (for a workspace in standard mode) to view all the tables of the workspace in the specified environment. You can also enter a table name in the search box to search for required tables in fuzzy match mode.

View accounts that have permissions on a table

On the **Table** tab of the **Authorizations** page, click the plus sign (+) in front of a table to view all the accounts that have permissions on the table.

Revoke table permissions

Click **Revoke Permission** in the Actions column for an account to revoke the permissions of the account on the current table.

View field permissions

Click **View Field Permissions** in the Actions column for an account to view the permissions of the account on the fields in the current table.

Revoke field permissions

If LabelSecurity is enabled for the workspace, select fields on the Field Permissions page and click **Revoke Field Permissions** to revoke the permissions on the fields.

8.4. Approval Center

On the Approval Center page, you can view your requests and their status, view and handle the requests pending your approval, and view the requests that you have handled.

My Requests

1. Move the pointer over the DataWorks icon in the upper-left corner, and click **Security Center**. In the left-side navigation pane, click **Approval Center**. On the Approval Center page, click the **My**

Requests tab.

On this tab, you can view the information about each of your requests, including **Object Type**, **Workspace**, **Status**, **MaxCompute Project**, **Request Time**, and **Table**.

> **Note** If a request contains permission requests for tables that belong to different owners, Security Center automatically splits the request into multiple requests by table owner.

2. Click **View** in the Actions column to view the details about a request.

Pending My Approval

1. On the **Approval Center** page, click the **Pending My Approval** tab.

   On this tab, you can view the requests pending your approval. If a request is pending your approval, a red dot appears next to Approval Center and Pending My Approval to remind you.

   You can view the information about each of requests pending your approval, including **Object Type**, **Grant To**, **Request Time**, **Workspace**, **MaxCompute Project**, and **Table**.

2. Click **Handle** in the Actions column to view the details about a request and handle it on the Request Details page. The request details include the progress and objects requested.
3. Enter your comments and click Approve or Reject as required.

Handled by Me

1. On the **Approval Center** page, click the **Handled by Me** tab.

   On this tab, you can view the information about each request that you have handled, including **Object Type**, **Grant To**, **Result**, **Workspace**, **MaxCompute Project**, **Table**, and **Request Time**.

2. Click **View** in the Actions column to view the details about a request. The request details include the progress and objects requested.

8.5. FAQ

This topic describes the frequently asked questions (FAQs) about the Security Center service of DataWorks.

- Q: What permissions can I request in Security Center?

  A: In Security Center, you can request permissions on tables in DataWorks workspaces in the development environment and production environment.

- Q: What is the relationship between Data Management and Security Center?

  A: Security Center is a product that upgrades and replaces the permission and security features in Data Management. You can choose **Security Center > My Permissions** to view the permissions requested or granted by using the `odpscmd grant` command in **Data Management**.

  If you want to request other permissions and handle permission requests on the GUI, go to **Security Center** and perform operations as required. The **Data Management** service does not support permission request and approval any more.

- Q: Why cannot I select fields when I request permissions?

A: If LabelSecurity is enabled for a workspace, you can request permissions on fields in this workspace. If LabelSecurity is disabled for a workspace, you can request permissions only on tables in this workspace.

- Q: Who will handle my request?

  A: Your request is handled by a workspace administrator or a table owner. After either of them approves or rejects your request, the request is closed.

- Q: Why do I find two requests on the **My Requests** page after I submit only one request?

  A: The tables in your request belong to two owners. In this case, Security Center automatically splits your request into two by table owner.

- Q: I request permissions on a field for one month only. Why does the validity period of the permissions become permanent after my request is approved?

  A: The security level of this field is zero or not higher than the security level of your account.

- Q: Why do I obtain permissions on some tables and fields on which I have not requested any permissions?

  A: The possible causes are as follows:

  - An administrator has granted the permissions to you by running commands in the DataWorks console.
  - After your request is approved in Security Center, Security Center also grants you the permissions on fields whose security level is zero or not higher than the security level of your account, even though you have not requested the permissions.

- Q: Why does a request disappear from the **Pending My Approval** tab before I handle it?

  A: Another workspace administrator or the table owner has approved or rejected the request. The request is closed and no longer needs to be handled.

- Q: What can I do if the message "An error occurred in the MaxCompute project" appears when I specify the workspace and environment?

  A: Send the error message and error code to a workspace administrator for troubleshooting.

- Q: Why do I fail to revoke permissions on a field?

  A: You can revoke permissions only on the fields whose security level is higher than the security level of your account.

- Q: Why do I fail to request permissions by using my tenant account?

  A: By default, a tenant account has all permissions. Therefore, you do not need to request permissions for your tenant account. The tenant account hides unnecessary operations such as permission request. This does not affect the use of the tenant account.

- Q: In **Security Center**, can I view the permission request and approval records of **Data Management**?

  A: Security Center and Data Management have not synchronized permission request and approval records yet. You need to go to **Data Management** to view the permission request and approval records of Data Management.

- Q: Can I revoke permissions based on the request records in Security Center?

A: Currently, Security Center is not the only service that provides authorization. To facilitate permission revocation, the Authorizations page in Security Center provides an access control list (ACL) of all users, regardless of the authorization channel. You can revoke any granted permissions without using the request records.

- Q: A permission request submitted in **Data Management** has not been approved yet. Do I need to submit it again in Security Center?

  A: Security Center and Data Management have not synchronized permission request and approval records yet. You need to submit the permission request again in Security Center.

- Q: How do I specify the LabelSecurity parameter for fields?

  A: You need to go to **Data Map** to set the LabelSecurity parameter for fields.

9.Security Center (new version)

9.1. Overview

DataWorks Security Center helps you build a security system that can secure data and personal privacy. Security Center can meet various security requirements, such as auditing, in high-risk scenarios. You can use Security Center without the need to perform additional configurations.

Security Center provides security capabilities for big data systems in the cloud throughout the entire data security lifecycle. It also provides best practices for various security diagnostic scenarios based on security specifications. Security Center provides the following features:

- Data permission management

  Security Center supports fine-grained permission requesting, request processing, and permission auditing. This allows you to manage permissions based on the principle of least privilege. In addition, Security Center allows you to view the request processing progress and follow up request processing in real time. For more information, see Data access control.

- Security diagnosis

  Security Center provides features such as platform security diagnosis and data usage diagnosis. It also provides best practices for various security diagnosis scenarios based on security specifications. These features ensure that your business is run more effectively in a secure environment. For more information, see Platform security diagnosis.

9.2. Platform security diagnosis

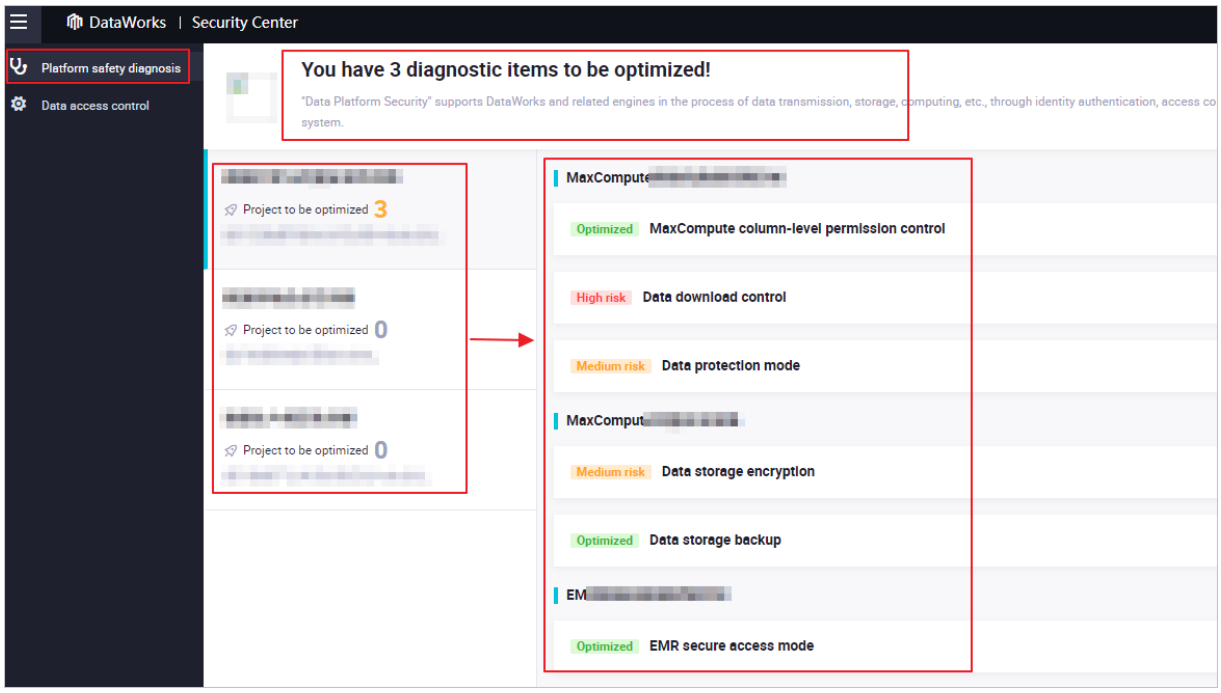
The platform security diagnosis feature of DataWorks provides security capabilities for features, such as identity authentication, access control, and development mode, during data transmission, storage, and computing on the nodes in the current DataWorks workspace and the associated compute engine. In addition, best practices are provided for security diagnosis. The platform security diagnosis feature helps you identify the security risks of your platform at the earliest opportunity and build a basic security system before you perform related transactions.

Go to the Platform safety diagnosis page

1. Log on to the DataWorks console.
2. In the upper-left corner of the page that appears, click the ☰ icon and choose **All Products > Data governance > Security Center**. The **Data access control** page appears.
3. In the left-side navigation pane, click **Platform safety diagnosis** to go to the **Platform safety diagnosis** page.

View diagnosis results

The **Platform safety diagnosis** page displays the security risks that are detected during business interactions between the current workspace and the associated compute engine instance based on the best practices for security risks. You can identify risk categories and levels based on the diagnosis results, view risk details, and process the items that can be optimized to ensure secure and reliable business interactions.



The following types of diagnostic items are provided:

- **Data calculation and storage**

  Diagnoses security issues for features such as data permission management, data storage encryption, and data storage backup, and identifies potential security risks at the earliest opportunity to improve security during data storage and access.

- **Data transmission security diagnosis**

  Diagnoses security issues for features such as the access control of data sources and the isolation of data sources in the production and development environments. In addition, this diagnostic item identifies security risks during data transmission so that you can manage these risks at the earliest opportunity. This diagnostic item ensures a secure and reliable environment for data transmission.

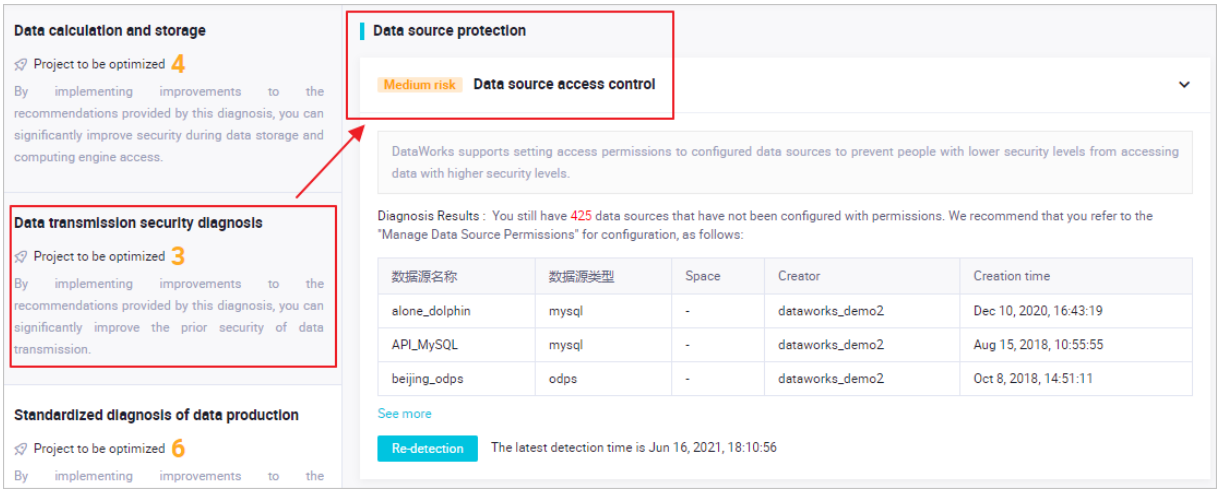
- **Standardized diagnosis of data production**

  Diagnoses security issues for production processes, such as the rationality of the roles, number of administrators, and deployment personnel within the current workspace, and allows you to identify and process security risks at the earliest opportunity. This diagnostic item improves the reliability and security of the data output system.

- **Platform security configuration diagnosis**

  Diagnoses security issues for features, such as auditing of operations on DataWorks, to improve the overall data security.

Potential security risks are classified as **medium** and **high** risks. You can click a security risk to view its details and manage the security risk at the earliest opportunity. The following figure shows the details about a **medium-level risk** of **data source access control**.



- Security risk

  Permissions are not configured on the data sources. This way, users of low security levels can access data of high security levels. This leads to insecure access to the data sources.

- Suggestion

  You can configure permissions for the data sources based on the provided suggestion to improve access security for the data sources.

9.3. Data access control

The data access control feature provides a visual interface that allows you to request permissions, process requests, view request processing progress, follow up request processing, and audit and manage permissions.

Limits

You can use the **data access control** feature to request only permissions on MaxCompute tables.

Usage note

The **Data access control** page displays the access control platform of the new version. If you want to use the access control platform of the earlier version, click **Return to old version** in the top navigation bar of the page. For more information about the access control platform of the earlier version, see Overview.

Go to the Data access control page

1. Log on to the DataWorks console.
2. In the upper-left corner of the page that appears, click the ☰ icon and choose **All Products > Data governance > Security Center**. The **Data access control** page appears.

Request permissions

1. Go to the **Permission application** tab.

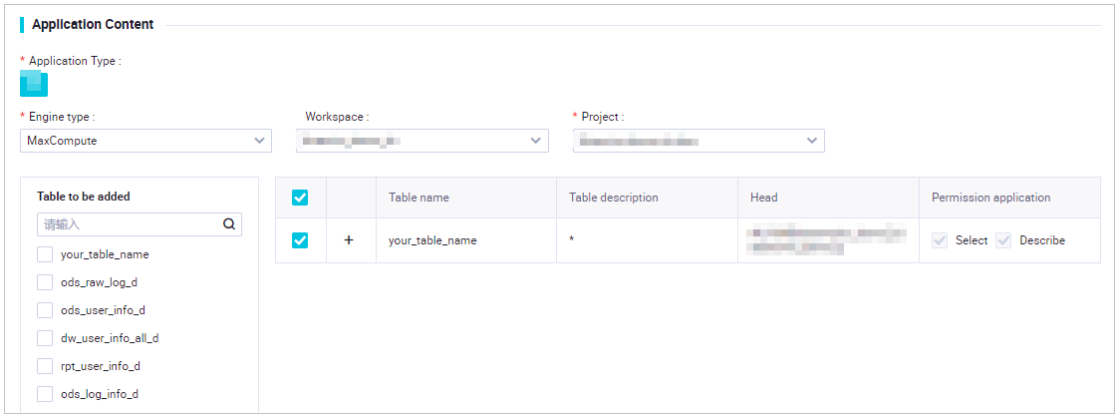
2. Select the tables on which you want to request permissions.
   i. In the **Application Content** section, set the **Workspace** and the **Project** parameters.

      You can use the **data access control** feature to request only permissions on MaxCompute tables.

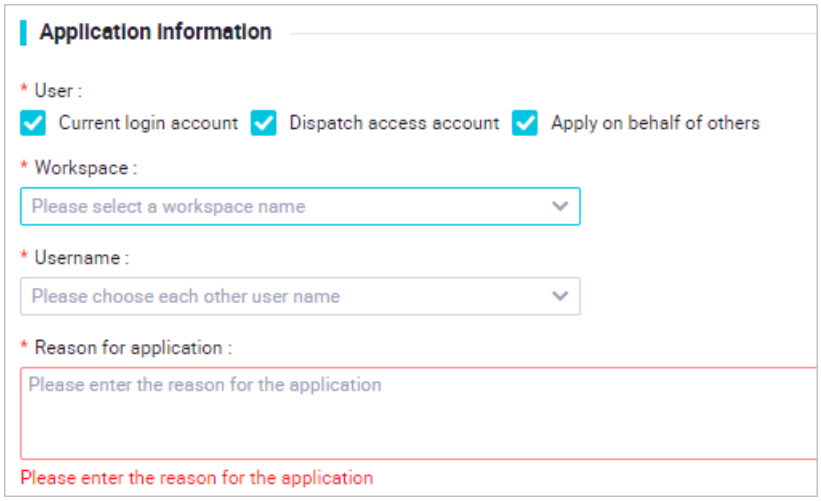
      The default value of the **Application Type** parameter is **Table** and that of the **Engine type** parameter is **MaxCompute**.

   ii. Select the tables on which you want to request permissions in the **Table to be added** section.

      After you select tables, the information of the tables is displayed on the right. You can click the + icon on the left side of a table name to view all the fields in the table. You can request the permissions on some or all fields. By default, the permissions on all fields are requested.



3. In the **Application information** section, set the parameters.



Parameter	Description

Parameter	Description
User	○ **Current login account**: Request the permissions on the tables for the account that is used to log on to the current workspace. ○ **Dispatch access account**: Request the permissions on the tables for the account that has a scheduling access identity. If you select this option, you must set the **Workspace** parameter. ○ **Apply on Behalf of others**: Request the permissions on the tables for an account that is not used to log on to the current workspace. If you select this option, you must set the **Username** parameter.
Workspace	The account that has a scheduling access identity.
Username	The username of the account that is not used to log on to the current workspace.
Reason for application	The reason why you want to request the permissions.

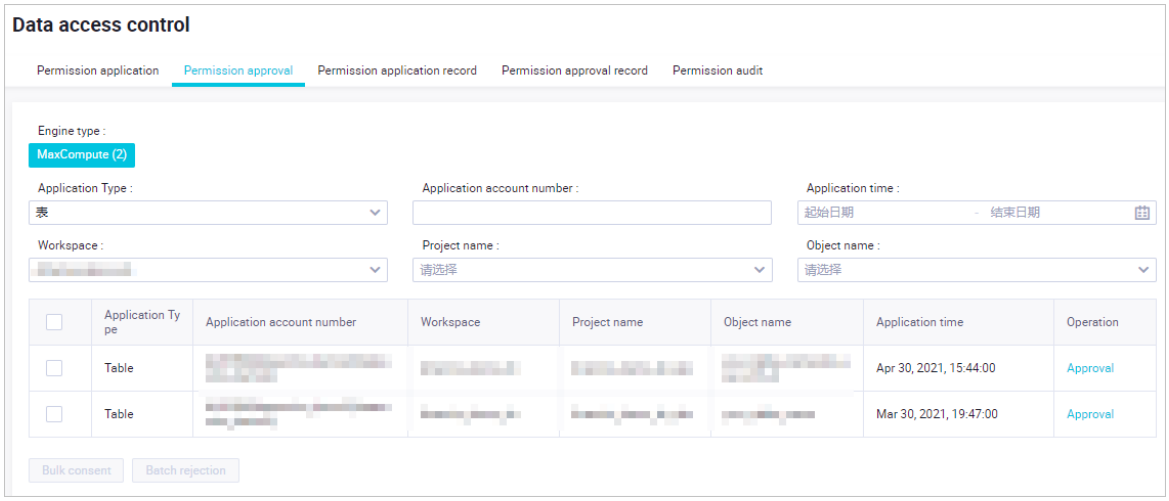
4. Click **Apply for permission** to submit the request.

   You can view the processing details and record of the current request on the **Permission application record** tab.

Process requests

1. View the information about the pending requests.

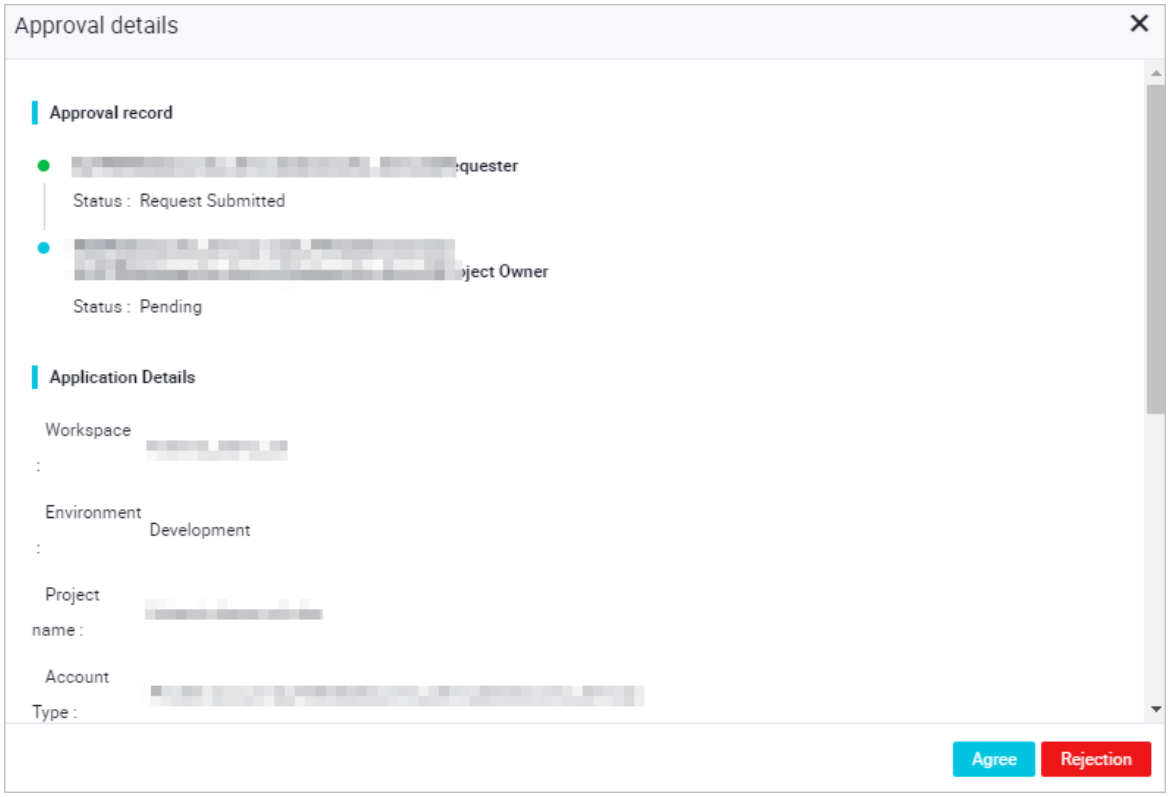
   Go to the **Permission approval** tab. You can use the following parameters to find the pending requests within the current Apsara Stack tenant account: **Application account number**, **Application time**, **Workspace**, **Project name**, and **Object name**.



2. View the details about a request.

   Find the request and click **Approval** in the **Operation** column. Then, you can view the details and

processing record of the request in the **Approval details** dialog box.



3. Process requests.

   To process a single request, enter your comments and click **Agree** or **Rejection** based on your business requirements.

   To process multiple requests at a time, you can select all the requests that you want to process on the **Permission approval** tab, click **Bulk consent** or **Batch rejection**, and then enter your comments.

View historical permission requests and their processing records

- Go to the **Permission application record** tab. Then, you can use the following parameters to find the historical permission requests within the current Apsara Stack tenant account: **Approval status**, **Application time**, **Workspace**, **Project name**, and **Table name**.

  You can click **View details** in the **Operation** column that corresponds to a request to view the details about the request. In addition, you can continue to process the requests whose approval states are **In approval**.

- Go to the **Permission approval record** tab. Then, you can use the following parameters to find the request processing records within the current Apsara Stack tenant account: **Application account number**, **Approval Results**, **Workspace**, **Project name**, **Object name**, and **Application time**.

  You can click **View details** in the **Operation** column that corresponds to a request to view the details about the request.

Audit permissions

Go to the **Permission audit** tab. Then, you can use the following parameters to find the permission requests that are processed for the workspace, project, or object in Security Center: **Workspace**, **Project name**, and **Object name**.

10.Approval Center

10.1. Overview

DataWorks Approval Center is a functional module that is used to manage approval processes for data permissions and sensitive behavior. You can define approval scopes and processes in Approval Center to meet different approval requirements in different scenarios.

Features

When you develop and manage data in DataWorks, you can manage permissions on table data and data service APIs with ease. You can use the default approval process provided by **Security Center** or customize an approval process in **Approval Center** to manage permissions.

When a permission application is submitted after custom approval processes are configured, DataWorks automatically checks whether the permissions to be applied for hit a custom approval process. If a custom approval process is hit, DataWorks forwards the application based on the custom approval process.

You can perform the following operations in DataWorks **Approval Center**:

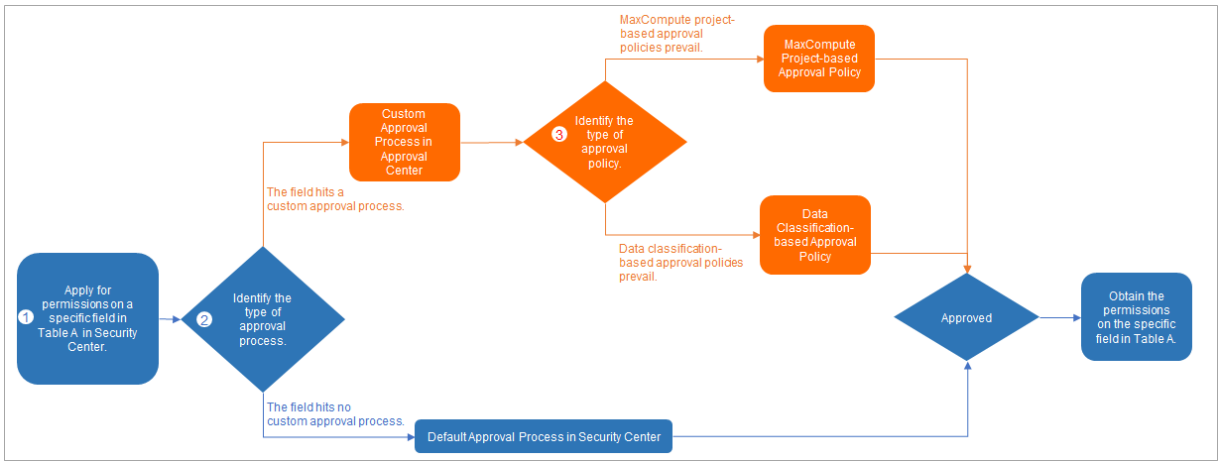
- Define an approval policy: You can specify the scope of approval objects and define an approval process to customize a process of managing key data resources and sensitive behavior. In addition, you can configure notification methods such as text messages, emails, or DingTalk chatbots.
- Process permission applications: You can process permission applications that you submit and process permission applications as an approver in approval processes in Approval Center.

For more information about custom approval policies, see Approval policies for MaxCompute data and Approval policies for data services.

After a custom approval policy is configured, you can process the applications for table permissions or data service permissions based on the approval policy. For more information, see Applications for table field permissions and approval processes and Applications for data service permissions and approval processes.

Applications for table field permissions and approval processes

After approval policies for MaxCompute data are configured in **Approval Center**, a user submits an application for the permissions on a specific table field in **Security Center**. Then, the application is processed based on the flowchart shown in the following figure.



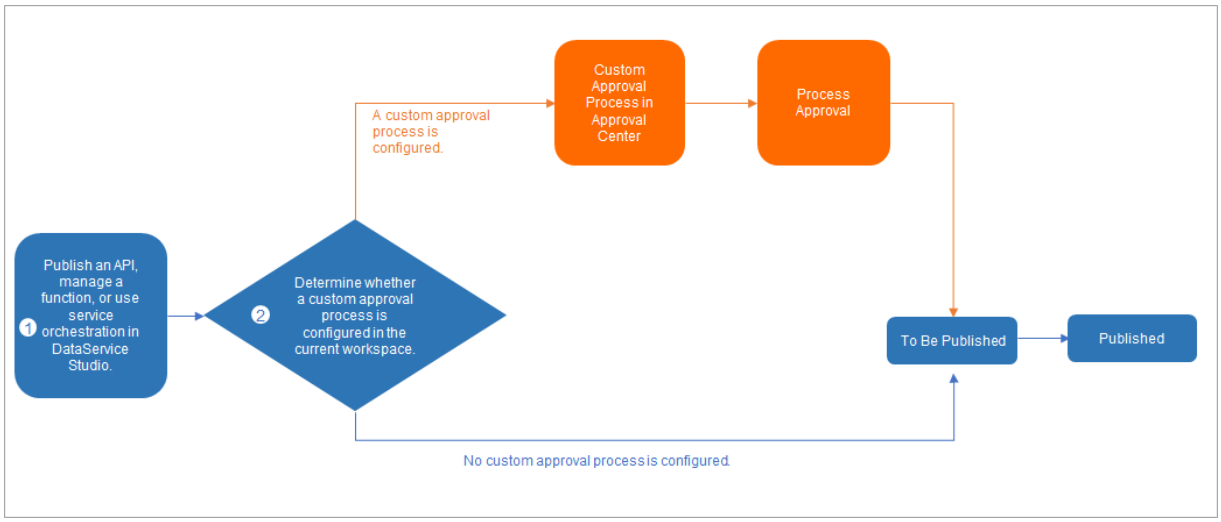
- When a user applies for the permissions on a specific field in a MaxCompute table, DataWorks identifies the type of approval process to be used based on the field.
  - If the field belongs to the data range specified in a custom approval policy, the custom approval process of the approval policy is hit. Then, DataWorks processes the permission application based on the custom approval process configured in Approval Center.
  - If the field does not belong to the data range specified in a custom approval policy, DataWorks processes the permission application based on the default approval process provided by Security Center.
- If a custom approval process is used to process the permission application, DataWorks determines the type of approval policy to be used based on the priorities of approval policies configured in Approval Center.

  When you configure a custom approval policy, you can specify the data range to which the approval policy applies based on a MaxCompute project or data classification in Data Security Guard. In addition, you can configure information such as approvers and notification methods, and set the priorities of MaxCompute project-based and data classification-based approval policies as required. For more information, see Approval policies for MaxCompute data.

Applications for data service permissions and approval processes

After approval policies for data services are configured, a custom approval process can be triggered if a data service operation is performed. Data service operations include publishing APIs, managing functions, and using service orchestration.

When an application is submitted for data service permissions in **Security Center**, the application is processed based on the flowchart shown in the following figure.



- An application is submitted for permissions on data service operations such as publishing APIs, managing functions, or using service orchestration. Then, DataService Studio determines whether a custom approval process is used to process the permission application based on whether a custom approval policy is configured in the current workspace.
  - If the custom approval process of a custom approval policy configured in Approval Center is hit, the permission application is processed based on the custom approval process.
  - If no custom approval process is hit, you can perform data service operations without the need to apply for permissions.
- If a custom approval process is used, DataWorks forwards the permission application based on the

approval policy configured in Approval Center.

When you configure a custom approval policy, you can specify the data range to which the approval policy applies based on a workspace. In addition, you can configure information such as approvers and notification methods. For more information, see Approval policies for data services.

10.2. Create and manage approval policies

10.2.1. Approval policies for MaxCompute data

You can customize approval processes for MaxCompute tables, resources, and functions.

Context

You can specify the data range to which an approval process applies based on a **MaxCompute project** or **data classification in Data Security Guard**. For more information, see the Specify the data range section of this topic.

Limits

- Only workspace administrators and the RAM users that are granted the AliyunDataWorksFullAccess permission can create and manage approval policies.
- Only DataWorks Enterprise Edition and Ultimate Edition allow you to configure approval policies for MaxCompute data.

Create an approval policy

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > Approval Center**.
3. In the left-side navigation pane of the page that appears, choose **Policies > Compute Engine**.

   On the page that appears, you can view a list of created approval policies, and modify and delete these approval policies.
4. Click **Create Policy** in the upper-right corner. Complete the Create Policy wizard.

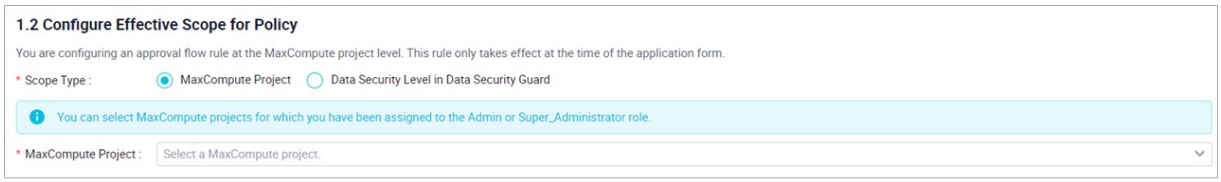
Enter the basic information

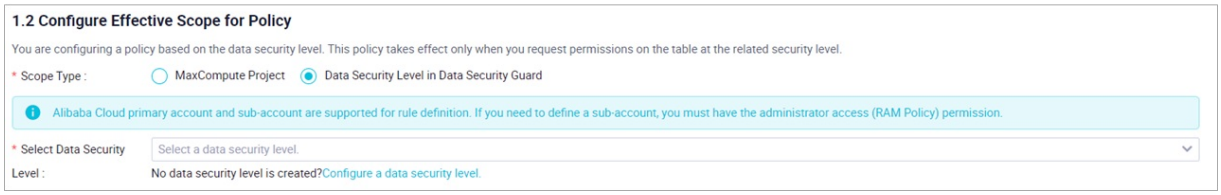
Set the **Policy Name** and **Purpose** parameters based on the actual scenario to which the approval policy applies.

Specify the data range

You must specify the data range to which this approval policy applies based on the actual scenario. After this approval policy is created, the applications for the permissions on the data in this data range must be processed based on this approval policy.

If a MaxCompute compute engine is used, you can specify the data range of an approval policy in a workspace based on a MaxCompute project or data classification in Data Security Guard.





When you specify the data range, take note of the following items:

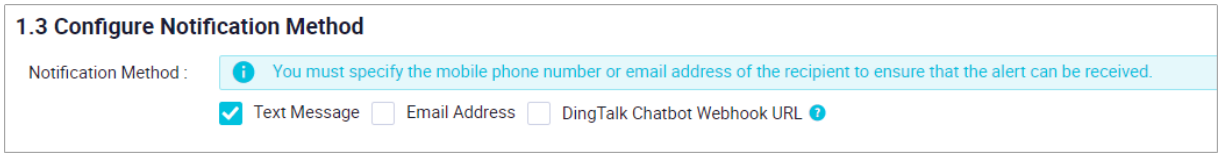
- Specify the data range based on a MaxCompute project
  - You must select an appropriate MaxCompute project from the **MaxCompute Project** drop-down list. This way, when applications are submitted to apply for the permissions on the tables in this MaxCompute project, this approval policy is used to process the applications.
  - A MaxCompute project can be associated with only one MaxCompute project-based approval policy. Otherwise, a policy conflict error is reported.
  - You can select a MaxCompute project in which the current account assumes the administrator or super administrator role. If no MaxCompute project is displayed in the drop-down list, the current account may not have the required permissions. In this case, you must use an account that is assigned the Admin or Super_Administrator role.

    > **Note** A DataWorks administrator is assigned the role_project_admin role in DataWorks workspaces, but not the Admin or Super_Administrator role in MaxCompute projects.
    >
    > To check the role of the current account, run the `whoami` command on the DataStudio page in DataWorks to obtain the account information. Then, run the `show grants for` *Your current account* command to check whether the current account is assigned the Admin or Super_Administrator role in a MaxCompute project.

- Specify the data range based on data classification in Data Security Guard
  - You must select a data security level from the **Select Data Security Level** drop-down list. This way, when applications are submitted to apply for the permissions on the tables at this data security level, this approval policy is used to process the applications.
  - A data security level can be associated with only one data classification-based approval policy. Otherwise, a policy conflict error is reported.
  - You can specify the data range by using an Apsara Stack tenant account or as a RAM user. If you specify the data range as a RAM user, one of the following conditions must be met:
    - The AdministratorAccess policy is attached to the RAM user.
    - The RAM user is granted the AliyunDataWorksFullAccess permission and assigned the Project Owner or Super_Administrator role of all MaxCompute projects.

Configure the notification methods

Three notification methods are supported: text messages, emails, and DingTalk chatbots.



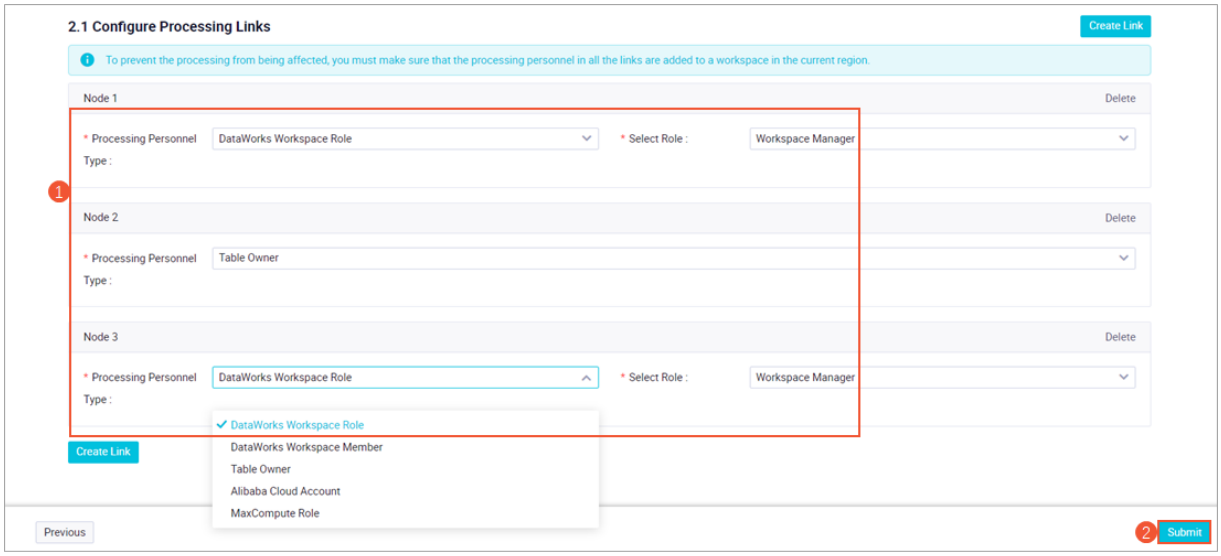
After you configure the notification methods, notifications are sent to approvers based on the configured notification methods when a permission application is submitted for approval.

> **Note** In the Configure Processing Links step, you can specify approvers on each approval node.

- To ensure that the approvers can receive approval notifications by using text messages or emails, you must add the approvers as alert contacts of DataWorks. For more information, see Configure and view alert contacts.
- To ensure that the approvers can receive notifications by using a DingTalk chatbot, select **Custom Keywords** when you set the **Security Settings** parameter in the Add Robot dialog box. Then, enter **DataWorks** in the Custom Keywords field. Make sure that the other check boxes are **cleared** when you set the Security Settings parameter.

  If you do not add DataWorks as a custom keyword or you select other check boxes when you set the Security Settings parameter, the approvers cannot receive notifications by using the DingTalk chatbot.

Configure the approval nodes



When you configure the approval nodes, take note of the following items:

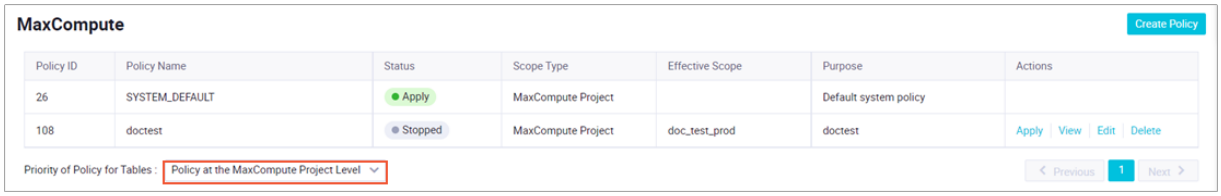
- The approval nodes are sequentially connected. After you configure the approval policy, the approval process specified in the approval policy sequentially flows from node to node. After an approver on an approval node gives approval, the approvers on the next approval node receive a notification and then start approval.
- You can specify different roles as approvers on different approval nodes. The following roles are supported: DataWorks workspace roles, DataWorks workspace member, table owner, Apsara Stack

tenant account, and MaxCompute roles.

> **Note**
>
> - When an application is submitted for approval, DataWorks sends notifications to the approvers on the approval nodes based on the notification methods configured in the preceding step. You must add the approvers as alert contacts of DataWorks. For more information, see Configure and view alert contacts.
> - If multiple users that assume the same role are specified as approvers on an approval node, notifications are sent to all the approvers. In this case, if one of the approvers on an approval node gives approval, the application is forwarded to the next approval node.

Set priorities for approval policies

If both MaxCompute project-based and data classification-based approval policies are configured, a specific data range may hit both types of approval policies. In this case, you can set priorities for the two types of approval policies.



10.2.2. Approval policies for data services

DataWorks allows workspace administrators to configure approval policies for publishing data service APIs in workspaces.

Limits

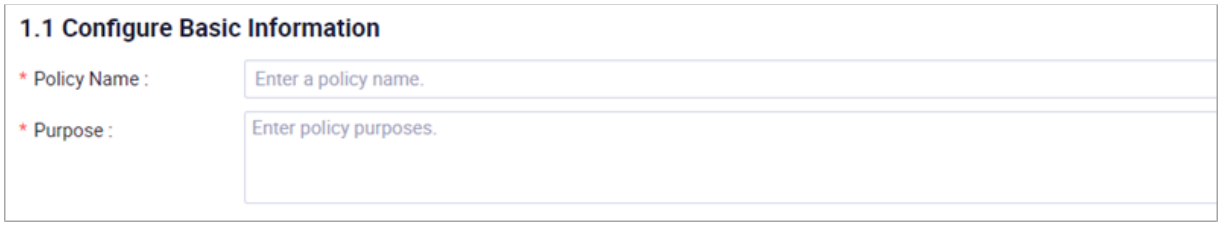
Only DataWorks Enterprise Edition and Ultimate Edition allow you to configure approval policies for data services.

Create an approval policy

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > Approval Center**.
3. In the left-side navigation pane of the page that appears, choose **Policies > DataService Studio**.

   On the page that appears, you can view a list of created approval policies, and modify and delete these approval policies.
4. Click **Create Policy** in the upper-right corner. Complete the Create Policy wizard.

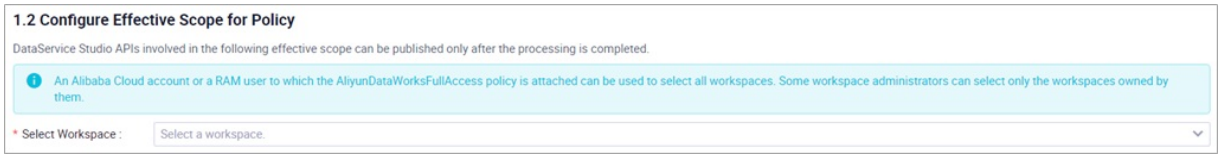
Enter the basic information



Set the **Policy Name** and **Purpose** parameters based on the actual scenario to which the approval policy applies.

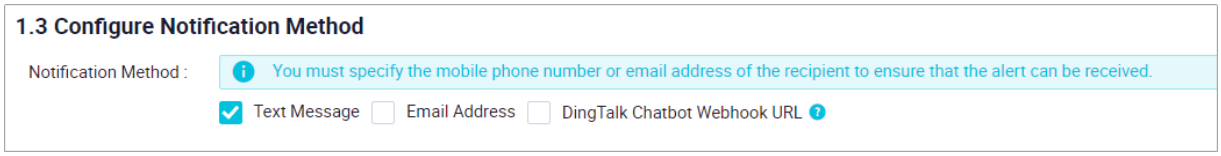
Specify the data range

You must specify the data range to which this approval policy applies based on the actual scenario. After this approval policy is created, the applications for the permissions on the data in this data range must be processed based on this approval policy.



Configure the notification methods

Three notification methods are supported: text messages, emails, and DingTalk chatbots.



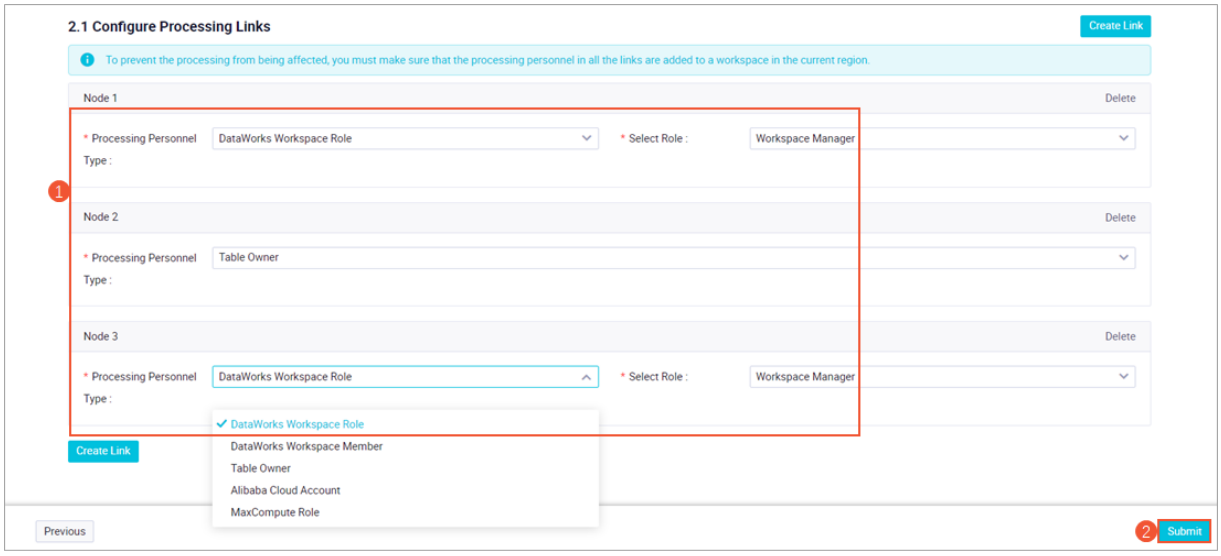
After you configure the notification methods, notifications are sent to approvers based on the configured notification methods when a permission application is submitted for approval.

> **Note** In the Configure Processing Links step, you can specify approvers on each approval node.

- To ensure that the approvers can receive approval notifications by using text messages or emails, you must add the approvers as alert contacts of DataWorks. For more information, see Configure and view alert contacts.
- To ensure that the approvers can receive notifications by using a DingTalk chatbot, select **Custom Keywords** when you set the **Security Settings** parameter in the Add Robot dialog box. Then, enter **DataWorks** in the Custom Keywords field. Make sure that the other check boxes are **cleared** when you set the Security Settings parameter.

  If you do not add DataWorks as a custom keyword or you select other check boxes when you set the Security Settings parameter, the approvers cannot receive notifications by using the DingTalk chatbot.

Configure the approval nodes



When you configure the approval nodes, take note of the following items:

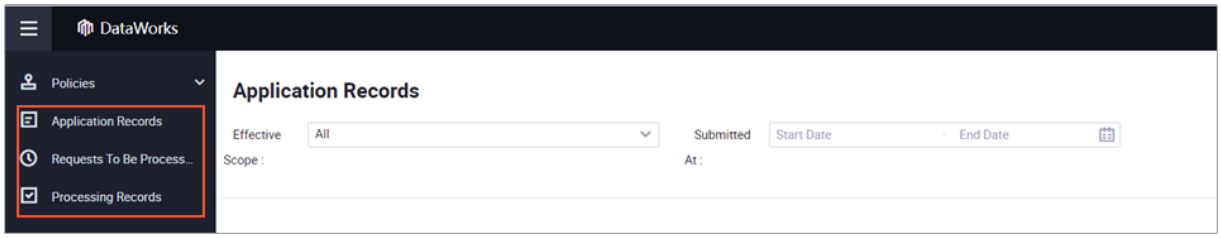
- The approval nodes are sequentially connected. After you configure the approval policy, the approval process specified in the approval policy sequentially flows from node to node. After an approver on an approval node gives approval, the approvers on the next approval node receive a notification and then start approval.
- You can specify different roles as approvers on different approval nodes. The following roles are supported: DataWorks workspace roles, DataWorks workspace member, table owner, Apsara Stack tenant account, and MaxCompute roles.

> **Note**
>
> - When an application is submitted for approval, DataWorks sends notifications to the approvers on the approval nodes based on the notification methods configured in the preceding step. You must add the approvers as alert contacts of DataWorks. For more information, see Configure and view alert contacts.
> - If multiple users that assume the same role are specified as approvers on an approval node, notifications are sent to all the approvers. In this case, if one of the approvers on an approval node gives approval, the application is forwarded to the next approval node.

10.3. Process and view applications

You can view all applications submitted by the current account and the applications that are approved and to be approved by the current account in DataWorks Approval Center.

Approval Center page



- On the **Application Records** page, you can view all applications submitted by the current account.

You can filter application records based on **Effective Scope** and **Submitted At**.

- On the **Requests To Be Processed** page, you can view all applications to be approved by the current account. Find an application that you want to process and click **Process** in the upper-right corner. In the dialog box that appears, confirm the information, enter your comments, and then determine whether to approve or reject the application.
- On the **Processing Records** page, you can view all applications that are approved by the current account. You can filter approval records based on **Effective Scope**, **Submitted At**, and **Initiated By**.

11.Data Quality

11.1. Overview

DataWorks provides a Data Quality service for you to control the data quality of disparate connections. In Data Quality, you can check data quality, configure alert notifications, and manage connections.

Relying on DataWorks, Data Quality provides a comprehensive data quality solution that has various features. For example, you can detect data, compare data, monitor data quality, and use intelligent alerting.

Data Quality monitors data in datasets. Currently, it allows you to monitor MaxCompute tables and DataHub topics. When offline MaxCompute data changes, Data Quality checks data and blocks nodes if it detects exceptions. This prevents nodes from being affected. In addition, Data Quality allows you to manage the check result history so that you can analyze and evaluate the data quality.

For streaming data, Data Quality uses DataHub to monitor data streams and sends alert notifications to subscribers if it detects stream discontinuity. You can also set the alert severity such as warning and error alerts, and the alert frequency to minimize repeated alerts.

The following figure shows the monitoring flowchart in Data Quality.

> **Note** Data Quality monitors the quality of data from MaxCompute and DataHub datasets. To use Data Quality features, you need to create tables and write data to the tables.
>
> You can create MaxCompute tables and write data to the tables in the MaxCompute console or in the DataWorks console.

Log on to the DataWorks console. Click the DataWorks icon in the upper-left corner and choose **All Products > Data Quality** to go to the **Data Quality** page.

11.2. Features

11.2.1. Dashboard

As the homepage of Data Quality, the Dashboard page displays an overview of alerts and blocks for subscribed nodes. You can set filter conditions to view the required alerts and blocks.

Card	Description
My MaxCompute Partition Subscriptions	Displays the number of MaxCompute partitions with alerts or blocks and the number of normal MaxCompute partitions on the current day. You can click this card to go to the Search by Node page for the MaxCompute connection and view alert details.
My DataHub Topic Subscriptions	Displays the number of DataHub topics with alerts and the number of normal DataHub topics on the current day. You can click this card to go to the Search by Node page for the DataHub connection and view alert details.
Current task alarm condition	Displays alerts for MaxCompute and DataHub connections of the current workspace on the current day.

Card	Description
Current task blocking situation	Displays blocks for the MaxCompute connection of the current workspace on the current day.
Task Alarm Situation Trend	Displays the trend chart of alerts for MaxCompute and DataHub connections. You can view the alert trend in the past 7 or 30 days, or a custom time period within the past three months.
Task Blocking Situation Trend Graph	Displays the trend chart of blocks for MaxCompute and DataHub connections. You can view the block trend in the past 7 or 30 days, or a custom time period within the past three months.

11.2.2. My Subscriptions

The My Subscriptions page displays all nodes subscribed by your account.

Go to the My Subscriptions page

Currently, Data Quality allows you to monitor MaxCompute tables and DataHub topics. You can select a connection on the **My Subscriptions** page and search for subscribed nodes of the connection.

1. Log on to the DataWorks console.
2. Click the DataWorks icon in the upper-left corner and choose **All Products > Data Quality**.
3. In the left-side navigation pane, click **My Subscriptions**. The **My Subscriptions** page appears.

Subscribed MaxCompute connections

On the **My Subscriptions** page, select **MaxCompute** from the connection drop-down list in the upper-left corner. All the subscribed MaxCompute connections appear.

- You can click a partition expression on the right to go to the **Rules** page.
- You can click **View Check Results** in the Actions column for a partition expression to go to the Search by Node page.
- Data Quality supports the following four notification methods: **Email**, **Email and SMS**, **DingTalk Chatbot**, and **DingTalk Chatbot @ALL**.
- You can click **Cancel Subscription** to unsubscribe from the connection.

Subscribed DataHub connections

On the **My Subscriptions** page, select **DataHub** from the connection drop-down list in the upper-left corner. All the subscribed DataHub connections appear.

- After you click **Alerts** for a topic, the **Alerts** page appears, allowing you to view detailed information about the rule alert.
- You can click **Notification Method** for a topic to change the notification method of the rule alert.
- You can click **Cancel Subscription** in the Actions column for a topic to unsubscribe from the topic.

11.2.3. Configure monitoring rules

Data Quality can monitor data in the MaxCompute, DataHub, and E-MapReduce data stores. This topic describes how to configure a rule for monitoring a table or topic.

Go to the Monitoring Rules page

1. Log on to the DataWorks console.
2. On the DataStudio page, click the DataWorks icon in the upper-left corner and choose **All Products > Data Quality**.
3. On the Data Quality page, click **Monitoring Rules** in the left-side navigation pane. On the Monitoring Rules page, you can select ODPS, Datahub, or EMR from the Engine/Data Source drop-down list.
   - If you select ODPS or EMR, all tables in the current MaxCompute or E-MapReduce data store appear. You can also switch to another data store or enter a keyword in the search box to search for topics or tables.
   - If you select Datahub, all topics and dimension tables in the current DataHub data store appear. You can also switch to another data store or enter a keyword in the search box to search for topics or tables.
4. Find the target table or topic and click **View Monitoring Rules** in the Actions column. The rule configuration page for the table or topic appears.

   Data Quality allows you to configure template rules and custom rules for a table or topic.

   > **Note** Before you configure a template rule for a table, you must configure a partition filter expression.

Create a template rule

1. Click **View Monitoring Rules** in the Actions column of a table or topic.
2. On the rule configuration page that appears, click the partition filter expression for which you want to configure a template rule. Then, click **Create rules**. In the Create rules right-side pane, the **Template Rules** tab appears.

   On the **Template Rules** tab, click **Add Monitoring Rule** or **Quick Create** to create a template rule.

   - **Click Add Monitoring Rule.**

     Set parameters in the rule configuration section that appears, as described in the following table.

Parameter	Description
Rule Name	The name of the rule.
Rule Type	The type of the rule. Valid values: Rule Type and Soft. ■ If you select Rule Type, error alerts are reported and descendant nodes are blocked, whereas warning alerts are reported but descendant nodes are not blocked. ■ If you select Soft, error alerts are reported but descendant nodes are not blocked, whereas warning alerts are not reported and descendant nodes are not blocked.
Field	The fields to be monitored. You can select **All Fields in Table** or select a field of a numeric type or non-numeric type.

Parameter	Description
Template	The template to apply to the rule. Data Quality supports 37 rule templates. **Note** You can set field-specific rules of the average value, accumulated value, minimum value, and maximum value only for numeric fields.
Comparison Method	The comparison method of the rule. Valid values: **Absolute Value**, **Raise**, and **Drop**.

Parameter	Description
Thresholds	■ You can calculate the fluctuation by using the following formula: Fluctuation = (Sample - Baseline)/Baseline ■ You can calculate the fluctuation variance only for numeric fields such as BIGINT and DOUBLE fields by using the following formula: Fluctuation variance = (Sample - Baseline)/Standard deviation **Note** The sample and baseline are defined in the following way: ■ Sample: the sample value for the current day. For example, if you need to check the fluctuation of table rows on an SQL node in a day, the sample is the number of table rows on the current day. ■ Baseline: the comparison value from the previous N days. Examples: ■ If you need to check the fluctuation of table rows on an SQL node in a day, the baseline is the number of table rows on the previous day. ■ If you need to check the fluctuation of the average number of table rows on an SQL node in seven days, the baseline is the average number of table rows in the last seven days. You can set **Warning Threshold** and **Error Threshold** to monitor data at different severities: ■ If the fluctuation does not exceed the warning threshold, Data Quality determines that data is normal. ■ If the fluctuation exceeds the warning threshold but does not exceed the error threshold, Data Quality reports a warning alert. ■ If the fluctuation exceeds the error threshold, Data Quality reports an error alert. ■ If you do not specify the warning threshold, Data Quality reports error alerts or normal based on the monitoring result. ■ If you do not specify the error threshold, Data Quality reports warning alerts or normal based on the monitoring result. ■ If you specify neither the warning threshold nor the error threshold, Data Quality reports error alerts if it detects anomalies. However, you must specify at least one of the two thresholds. If you specify neither of them, Data Quality applies default values, namely, 10% for the warning threshold and 50% for the error threshold.

- **Click Quick Create.**

  Set parameters in the rule configuration section that appears, as described in the following table.

Parameter	Description
Rule Name	The name of the rule.
Field	The fields to be monitored. You can select **All Fields in Table** or a specific field of a numeric type or non-numeric type.
Trigger	The trigger condition of the rule. If you select All Fields in Table for the Field parameter, **The number of rows is greater than 0** is selected by default.

3. Click **Batch Create**.

Create a custom rule

If template rules do not meet your requirements for monitoring the data quality, you can create custom rules.

1. Click **View Monitoring Rules** in the Actions column of a table or topic.
2. On the rule configuration page that appears, click the partition filter expression for which you want to configure a custom rule. Then, click **Create rules**. In the Create rules right-side pane, the **Template Rules** tab appears.
3. Click the **Custom Rules** tab. On the **Custom Rules** tab, click **Add Monitoring Rule** or **Quick Create** to create a custom rule.
   - **Click Add Monitoring Rule.**

     Set parameters in the rule configuration section that appears, as described in the following table.

Parameter	Description
Rule Name	The name of the rule.
Field	The fields to be monitored. You can select **All Fields in Table**, **SQL Statement**, or a specific field. ■ If you select All Fields in Table or a specific field, you can specify the WHERE clause to customize filter conditions based on business requirements. ■ If you select SQL Statement, you can customize the SQL logic to set a rule. The return value is the value in a row of a column.
Rule Type	The type of the rule. Valid values: Rule Type and Soft. ■ If you select Rule Type, error alerts are reported and descendant nodes are blocked, whereas warning alerts are reported but descendant nodes are not blocked. ■ If you select Soft, error alerts are reported but descendant nodes are not blocked, whereas warning alerts are not reported and descendant nodes are not blocked.

Parameter	Description
Sampling Method	The statistical function. Valid values: **count** and **count/table_count**.
Filter	The filter condition of the rule. For example, if you need to query partitions of the table based on a specific data timestamp, you can specify `pt=$[yyyymmdd-1]` as the filter condition.
Check type	The threshold type of the rule. Valid values: **Numeric type** and **Fluctuation**.
Comparison Method	The comparison method of the rule. If you set Check type to Numeric type, the values that are optional for this parameter include **Greater Than**, **Greater Than or Equal To**, **Equal To**, **Unequal To**, **Less Than**, and **Less Than or Equal To**.
Verification Method	The verification method of the rule. If you set Check type to Numeric type, you can only set this parameter to **Compare with a specified value**.
Expected Value	The expected value of the rule.
Description	The description of the rule.

- **Click Quick Create.**

  Set parameters in the rule configuration section that appears, as described in the following table.

Parameter	Description
Rule Name	The name of the rule.
Trigger	The type of the rule. You can select only **Values Duplicated in Multiple Fields**.
Field	The fields to be monitored.

Associate a custom node with Data Quality monitoring rules

Before you associate a custom node with Data Quality monitoring rules, make sure that the custom node is created and committed to the production environment. For more information, see Create a custom node type.

You can use one of the following methods to associate a custom node with Data Quality monitoring rules:

- Associate a custom node with Data Quality monitoring rules on the Data Quality page.
  i. Log on to the DataWorks console.
  ii. On the DataStudio page, click the DataWorks icon in the upper-left corner and choose **All Products > Data Quality**.
  iii. On the Data Quality page, click **Monitoring Rules** in the left-side navigation pane.
  iv. Select the target data store from the Engine/Data Source drop-down list, find the target table or topic, and then click **View Monitoring Rules** in the Actions column.

v. On the rule configuration page that appears, click the partition filter expression for which monitoring rules are configured.

vi. Click **Manage Linked Nodes**.

vii. In the **Manage Linked Nodes** dialog box, select the target workspace, enter the ID or name of the custom node, and then click **Create**.

- Associate a custom node with Data Quality monitoring rules on the Operation Center page.

i. Log on to the DataWorks console.

ii. On the DataStudio page, click the DataWorks icon in the upper-left corner and choose **All Products > Operation Center**.

iii. In the left-side navigation pane, choose **Cycle Task Maintenance > Cycle Task**.

iv. Find the target node and choose **More > Configure Data Quality Rules** in the Actions column.

v. In the **Configure Data Quality Rules** dialog box, set the **Workspace**, **Table Name**, **Engine type**, **Engine instance**, and **Partition Expression** parameters, and click **Add**.

11.2.4. View monitoring results

The Node Query page displays the monitoring results of rules. After monitoring rules are triggered, you can go to the Node Query page to view the monitoring results of the rules.

Go to the Node Query page

1. Log on to the DataWorks console.
2. On the **DataStudio** page, click ☰ in the upper-left corner and choose **All Products > DataAnalysis**.
3. On the Data Quality page, click **Node Query** in the left-side navigation pane.

   On the **Node Query** page, you can set parameters, such as the **Engine/Data Source**, **Status**, and **My Subscriptions** parameters, to filter nodes and view the monitoring results.

View the monitoring results of E-MapReduce and MaxCompute tables

GUI element	Description
Engine/Data Source	The name of the compute engine. In this example, select **EMR** or **ODPS**.
Engine/Database Instance	The E-MapReduce instance or MaxCompute project where the desired tables reside.
Status	The monitoring result of rules. Pay attention to partitions that trigger alerts or blocks.
Data Timestamp	The data timestamp.
My Subscriptions	Specifies whether to display only monitoring results of tables that you subscribed to.
Run At	The time when rules were triggered.

GUI element	Description
Table Name	The name of the table whose monitoring results you want to view.
Node	The node that triggered rules.
Details	Click **Details** in the **Actions** column of a table. On the page that appears, you can perform the following operations on each rule configured for the table: • Click **View History Check Results** in the Actions column of a rule to view the monitoring result history of the rule. • Enter comments on a rule based on the execution status of the rule. Perform the following steps to enter comments on a rule: i. Click **Problem Handling** in the Actions column of the rule. ii. In the **Problem Handling** dialog box, set the **Handling Method** and **Comments** parameters. iii. Click **OK**. **Notice** You can only use the problem handling feature in DataWorks Enterprise Edition or higher. • Click **Handling Logs** in the Actions column of a rule to view the processing history of the rule.
Rules	Click **Rules** in the Actions column of a table to go to the rule configuration page for the table. On this page, you can view partition filter expressions and rules configured for the table, and modify the rules as required. For more information, see MaxCompute monitoring.
View Log	Click **View Log** in the Actions column of a table to view the operational logs of rules configured for the table.
View Statistics	Click **View Statistics** in the Actions column of a table to view rule execution information about the table, including the number of rows and the table size.

View the monitoring results of DataHub topics

GUI element	Description
Engine/Data Source	The name of the compute engine. In this example, select **Datahub**.
Configure a data source	The name of the DataHub connection.
Status	The monitoring result of rules. Pay attention to topics that trigger alerts or blocks.
Topic	The name of the topic whose monitoring results you want to view.
My Subscriptions	Specifies whether to display only monitoring results of topics that you subscribed to.

GUI element	Description
Clear	Click **Clear** to clear the filter conditions you specified.
View Log	Click **View Log** in the Actions column of a topic to view the operational logs of rules configured for the topic.
Alerts	Click **Alerts** in the Actions column of a topic. On the Alerts page, you can view details about alerts triggered by the topic. On the **Alerts** page, you can click **Close** in the Actions column of an alert. In the message that appears, click **OK** to disable the alert.

11.2.5. Report Template Management

On the **Report Template Management** page, you can create a template of data quality reports. DataWorks can periodically generate and send data quality reports based on the template.

Create a report template

1. Log on to the DataWorks console. On the DataStudio page, click the DataWorks icon in the upper-left corner and choose **All Products > Data Quality**.
2. On the Data Quality page that appears, choose **Configuration > Report Template Management** in the left-side navigation pane. The Report Template Management page appears.
3. Click **Create Report Template**. On the **Create Report Template** page that appears, set required parameters.

Section	Parameter	Description
Basic settings	**Name**	The name of the report template.
	Sending Cycle	The interval at which reports are sent. Valid values: **Every Day**, **Every Week**, **Every Month**, and **Do Not Send**. If you set Sending Cycle to Every Week or Every Month, you also need to specify the specific day on which reports are sent.
	Timespan	The number of days before the current day. DataWorks generates reports based on the data of those days. The maximum value of this parameter is 30.

Section	Parameter	Description
Statistics of Rule Configuration The Statistics of Rule Configuration section displays metrics about rule configuration for offline data and real-time data. You can select metrics based on your needs.	**Offline data**	The metrics about rule configuration for tables in the workspace. The metrics include **Table count**, **Partition expression count**, **Count of rule on offline data**, and **Rule coverage on tables**. The Rule coverage on tables metric indicates the ratio of tables for which quality monitoring rules are configured.
	Realtime data	The metrics about rule configuration for topics in the workspace. The metrics include **Topic count**, **Count of rule on realtime data**, **Count of rule on cut off data**, **Rule coverage on topic**, **Count of rule on delayed data**, and **Count of customized rule**. The Rule coverage on topic metric indicates the ratio of topics for which quality monitoring rules are configured.
Statistics of Rule Execution The Statistics of Rule Execution section displays metrics about rule running for offline data and real-time data. You can select metrics based on your needs. Quality reports display the selected metrics in charts.	**Offline data**	The metrics about rule running for tables in the workspace. The metrics are classified into the following types: **About rules**, **About partitions**, and **About tables**.
	Realtime data	The metrics about rule running for topics in the workspace. The metrics are classified into the following types: **About messages**, **About alarms**, and **About cut-offs**.
Manage Subscriptions	**Subscription Method**	The method used to notify report subscribers of new reports. Currently, DataWorks sends emails to notify report subscribers of new reports.
	Recipient	The recipient of report notifications. You can add multiple recipients.

Section	Parameter	Description
	Actions	The operations that you can perform on the subscription. You can click **Save** or **Cancel** in the Actions column of a subscription to save or cancel the subscription.
	Add Subscription	The button that allows you to add a subscription.

4. Click **Save** in the upper-right corner. A template of data quality reports is generated.

Preview a report template

After creating a report template, you can click **Preview** in the upper-right corner of the Create Report Template page to view the display format of reports generated based on this template.

> **Note** If report subscribers view reports through email notifications, they can only view the reports in tables. If they view reports on the Data Quality page, they can view reports in tables or charts.

11.2.6. Manage rule templates

In Data Quality, you can manage a set of custom rule templates and use the rule templates to improve the efficiency of rule configuration.

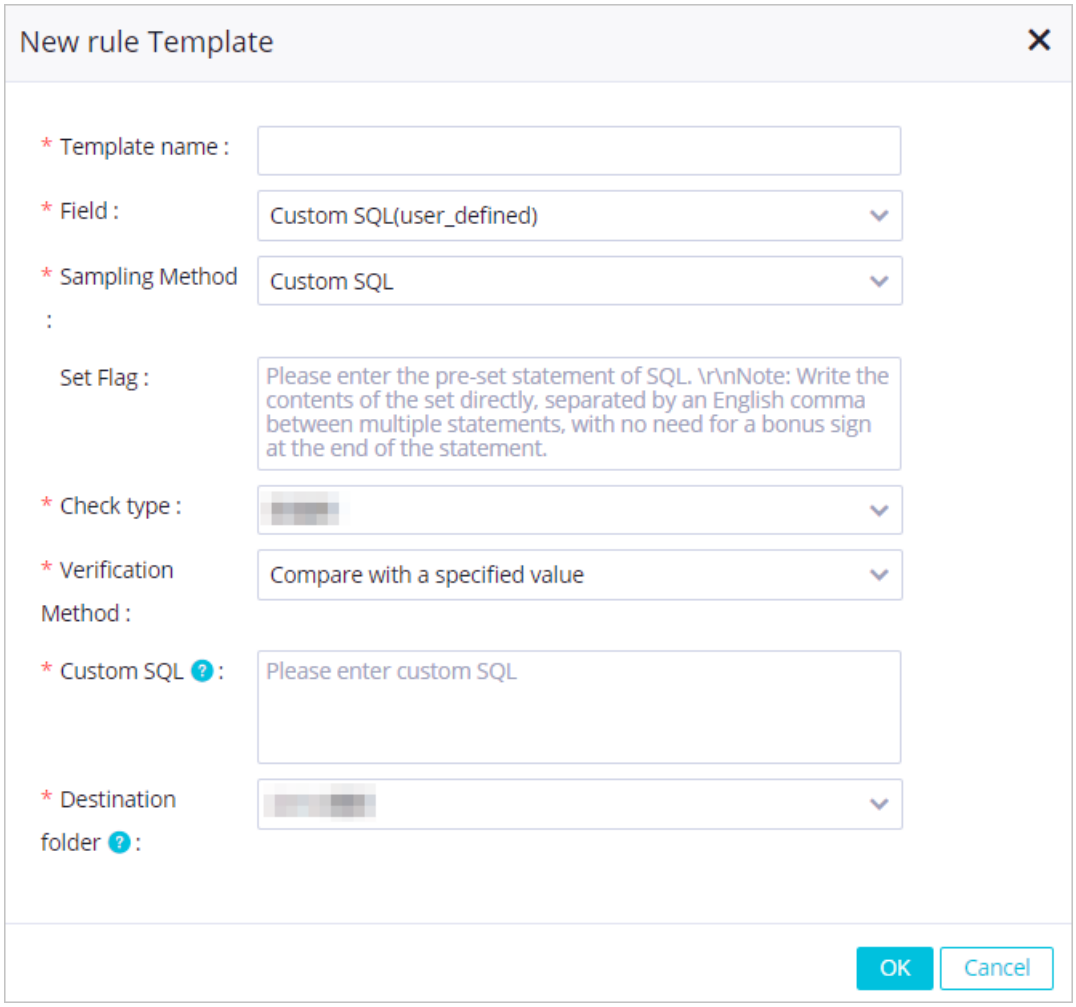
Context

You can create a rule template on the **Rule Templates** and **Monitoring Rules** pages. After the rule template is created, you can manage and use it.

Create a rule template on the Rule Template Management page

1. Log on to the DataWorks console.
2. On the DataStudio page, click in the upper-left corner and choose **All Products > Data Quality**.
3. On the Data Quality page, choose **Configuration > Rule Templates** in the left-side navigation pane.
4. On the Rule Templates page, click  and select **Create Folder**.
5. In the **Create Folder** dialog box, set the **Name** and **Location** parameters and click **OK**.
6. Right-click the folder name and select **Create Rule Template**.

   You can also rename or delete a folder.
7. In the **Create Rule Template** dialog box, set relevant parameters.



Parameter	Description
Template Name	The name of the rule template.
Field	The fields to be monitored and the statistical function. You can only set the two parameters to **Custom SQL**.
Sampling Method	
Set Flag	The `SET` clause of the SQL statement for querying the field to be monitored. **Note** Separate multiple statements with commas (,). You do not need to add a semicolon (;) at the end of each statement.
Check type	The threshold type of the rule. Valid values: **Numeric type** and **Fluctuation**.

Parameter	Description
Verification Method	The verification method of the rule template. The verification methods that can be selected vary with the threshold type. ○ If you set the **Check type** parameter to **Numeric type**, you can only set this parameter to **Compare with a specified value**. ○ If you set the **Check type** parameter to **Fluctuation**, the values that are optional for this parameter include **Compare the current value with the average value of the last 7 days**, **Compare the current value with the average value of the last 30 days**, **Compare the current value with the value 1 day before**, **Compare the current value with the value 7 days before**, **Compare the current value with the value 30 days before**, **The variance between the current value and the value 7 days before**, **The variance between the current value and the value 30 days before**, **Compare with the value 1, 7, and 30 days before** and **Compare with the value of the previous cycle**.
Custom SQL	The custom SQL statement. You can use ${tableName} as the table name.
Location	The name of the folder to which you want to store the custom rule template.

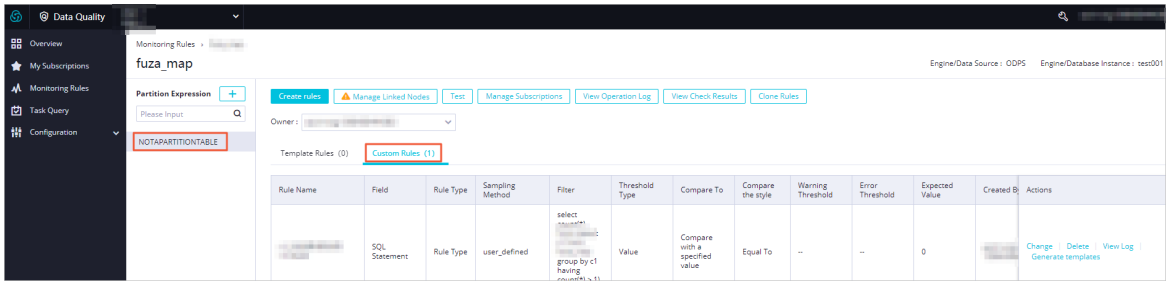
8. Click **OK**.

Create a rule template on the Monitoring Rules page

1. Go to the **Data Quality** page.
2. On the Data Quality page, click **Monitoring Rules** in the left-side navigation pane.
3. On the Monitoring Rules page, select the compute engine or data store, find the target table or topic, and then click **View Monitoring Rules** in the Actions column.

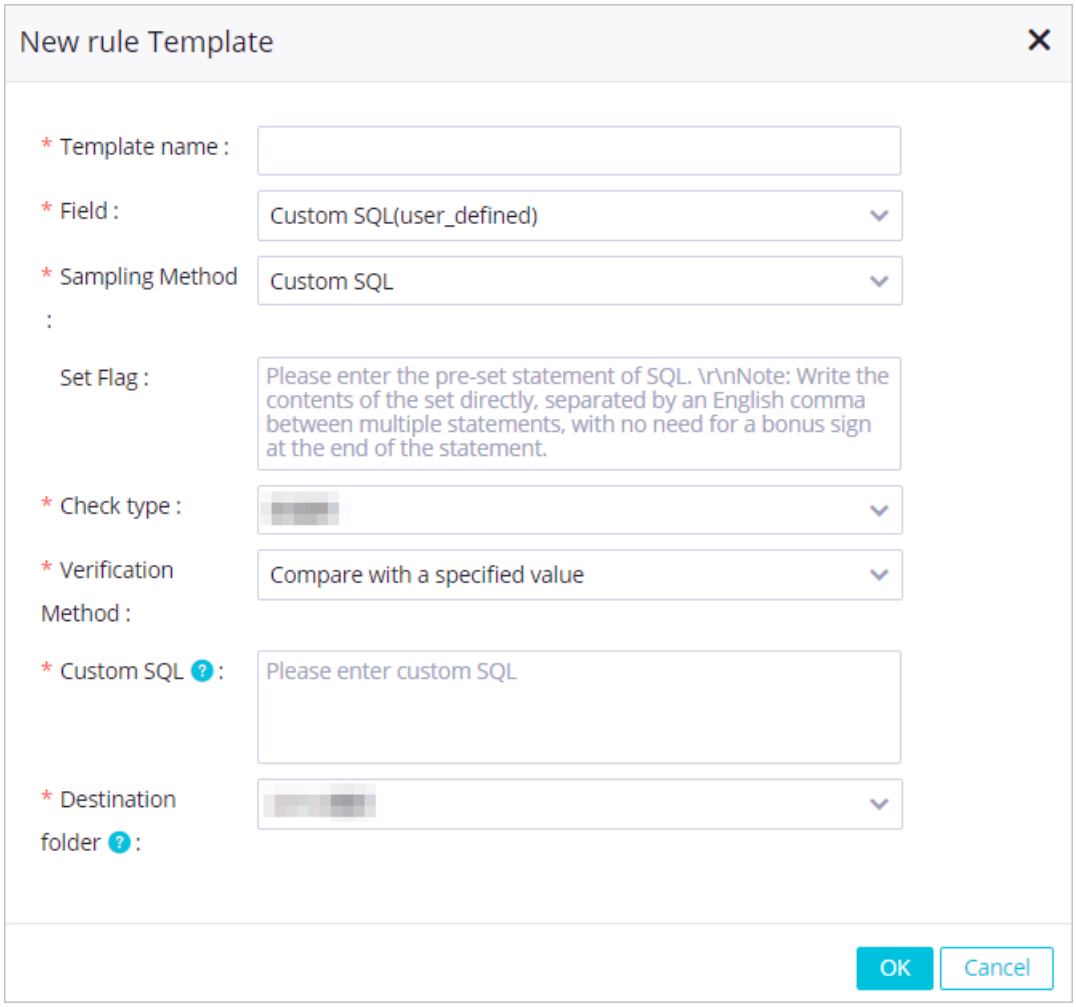
> **Note** This topic uses a MaxCompute table as an example.

4. Click a partition filter expression and then click the **Custom Rules** tab.



> **Note** For more information about how to create custom rules, see Custom rules.

5. On the Custom Rules tab, find the target custom rule and click **Generate Template** in the Actions column.
6. In the **Create Template** dialog box, set relevant parameters.



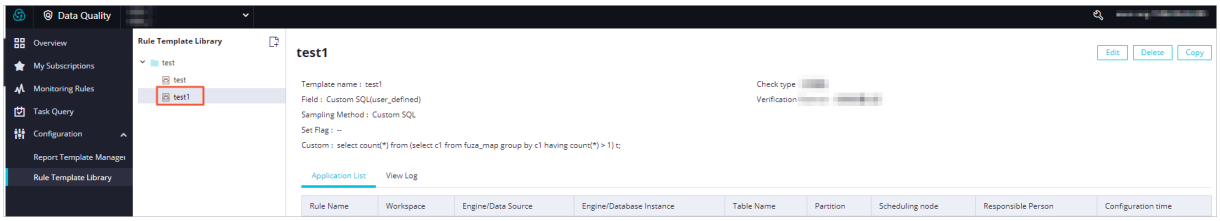
Parameter	Description
Template Name	The name of the rule template.
Field	The fields to be monitored and the statistical function. You can only set the two parameters to **Custom SQL**.
Sampling Method	
Set Flag	The `SET` clause of the SQL statement for querying the field to be monitored. **Note** Separate multiple statements with commas (,). You do not need to add a semicolon (;) at the end of each statement.
Check type	The threshold type of the rule. Valid values: **Numeric type** and **Fluctuation**.

Parameter	Description
Verification Method	The verification method of the rule template. The verification methods that can be selected vary with the threshold type. ○ If you set the **Check type** parameter to **Numeric type**, you can only set this parameter to **Compare with a specified value**. ○ If you set the **Check type** parameter to **Fluctuation**, the values that are optional for this parameter include **Compare the current value with the average value of the last 7 days**, **Compare the current value with the average value of the last 30 days**, **Compare the current value with the value 1 day before**, **Compare the current value with the value 7 days before**, **Compare the current value with the value 30 days before**, **The variance between the current value and the value 7 days before**, **The variance between the current value and the value 30 days before**, **Compare with the value 1, 7, and 30 days before** and **Compare with the value of the previous cycle**.
Custom SQL	The custom SQL statement. You can use ${tableName} as the table name.
Location	The name of the folder to which you want to store the custom rule template.

7. Click **OK**.
8. In the left-side navigation pane, choose **Configuration > Rule Templates** to view the created rule template.

Manage an existing rule template

On the Rule Templates page, you can click the name of a rule template to go to the template details page. On this page, you can view, edit, delete, or copy the rule template.



Action	Description
View	You can view the parameter configuration, the rules that use the rule template, and logs of the rule template: • The **Application List** tab displays the rules that use the rule template. • The **View Log** tab displays the logs of operations performed on the rule template, including the user who performed each operation, the time when each operation was performed, and the operation details.
Edit	Click **Edit** in the upper-right corner. In the **Edit Rule Template** dialog box, modify the required parameters, and click **OK**.

Action	Description
Delete	Click **Delete** in the upper-right corner. In the **Delete Template** message, click **OK**.
Copy	Click **Copy** in the upper-right corner. In the **Copy Rule Template** dialog box, set the **Template Name** and **Location** parameters and click **OK**.

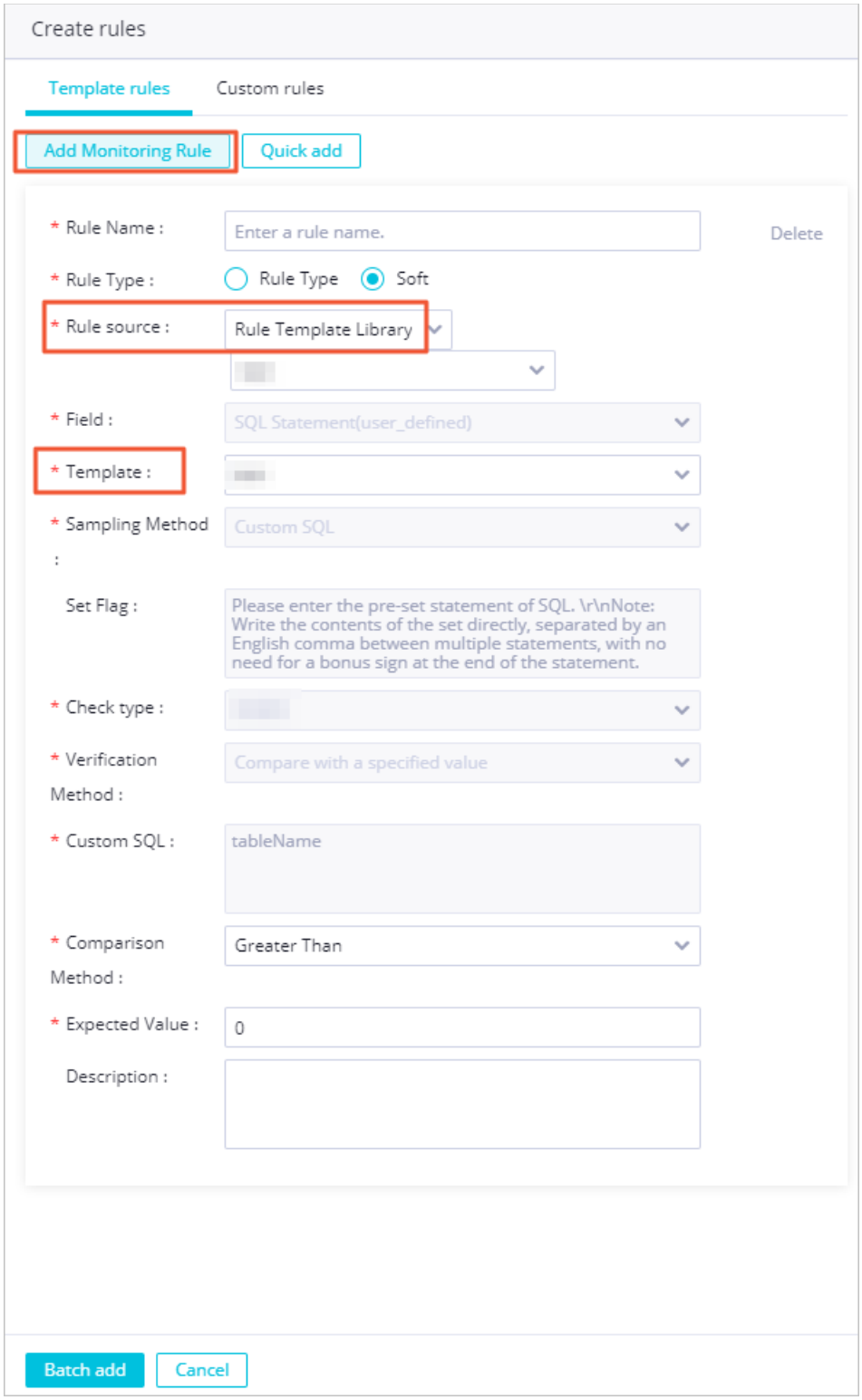
Use a rule template

When you create a monitoring rule, you can select a custom rule template to create the rule based on the rule template.

1. Go to the **Data Quality** page.
2. On the Data Quality page, click **Monitoring Rules** in the left-side navigation pane.
3. On the Monitoring Rules page, select the compute engine or data store, find the target table or topic, and then click **View Monitoring Rules** in the Actions column.

   > **Note** This topic uses a MaxCompute table as an example.

4. Click a partition filter expression and click the **Custom Rules** tab.
5. On the **Template Rules** tab of the **Create rules** right-side pane, click **Add Monitoring Rule**.
6. Set the parameters for the rule. Specifically, set the **Rule Source** parameter to **Rule Templates** and select a rule template. For more information about the parameter description, see Rules.



7. Click **Batch Create**.

11.3. User guide

11.3.1. Configure monitoring rules for MaxCompute

The Monitoring Rules page is the most important part of Data Quality, where you can configure rules to monitor data in E-MapReduce, MaxCompute, and DataHub. This topic describes how to configure monitoring rules for MaxCompute.

Add a MaxCompute connection

1. Log on to the DataWorks console.
2. On the **DataStudio** page, click the DataWorks icon in the upper-left corner and choose **All Products > Data Integration**.
3. On the Data Integration page, click **Connection** in the left-side navigation pane. The **Data Source** page appears.
4. Click **Add Connection** in the upper-right corner to add a MaxCompute connection.

Select the MaxCompute connection

1. On the current page, click ☰ in the upper-left corner and choose **All Products > Data Quality**.
2. On the Data Quality page, click **Monitoring Rules** in the left-side navigation pane.
3. Select **ODPS** from the **Engine/Data Source** drop-down list to display all tables in the MaxCompute data store.

   You can search for a table by table name. Fuzzy search based on the initial letters of a table name is supported.
4. Find the target table and click **View Monitoring Rules** in the Actions column.

Configure a partition filter expression

In Data Quality, you must configure rules based on a partition filter expression:

- To configure rules for a non-partitioned table, you can specify NOTAPARTITIONTABLE as the partition filter expression.
- To configure rules for a partitioned table, you can specify a data timestamp expression, such as $[yyyymmdd], or a regular expression as the partition filter expression.

On the rule configuration page of a table, click **+** next to **Partition Expression** to add a partition filter expression.

You can create a partition filter expression or select a recommended partition filter expression.

- Create a partition filter expression

  In the **Add Partition** dialog box, enter a partition filter expression that conforms to the syntax as required. For a non-partitioned table, select **NOTAPARTITIONTABLE** from the recommended partition filter expressions.

  - For a table with only one partition, follow the format: Partition key=Partition value. The partition value can be either a constant or a system parameter. You must configure partition expressions by using the last partition.

- For a table with multiple partitions, follow the format: Partition key 1\=Partition value/Partition key 2=Partition value/Partition key N=Partition value. Each partition value can be either a constant or a system parameter. You must use brackets [ ] to indicate a parameter, such as $[yyyymmdd-N].

The data timestamp configured in a partition filter expression also determines the recurrence of the partition filter expression. For example, if the data timestamp is the date of five days ago, the partition filter expression is triggered every five days. The following table describes supported partition filter expressions.

Partition filter expression	Description
dt=$[yyyymmdd-N]	Indicates N days before.
dt=$[yyyymm01-1]	Indicates the first day of each month.
dt=$[yyyymm01-Nm]	Indicates the first day of the month that is N months before the current month.
dt=$[yyyymmld-1]	Indicates the last day of each month.
dt=$[yyyymmld-1m]	Indicates the last day of the month that is N months before the current month.
dt=$[hh24miss-1/24]	Indicates one hour before the hour specified by the data timestamp.
dt=$[hh24miss-30/24/60]	Indicates half an hour before the hour specified by the data timestamp.
$[yyyymmdd]	Indicates the data timestamp.
$[yyyymmdd-1]	Indicates one day before the data timestamp of the current instance.
$[yyyymmddhh24miss]	Indicates the data timestamp of the current instance. Follow the `yyyymmddhh24miss` format by understanding the following format description: ○ yyyy indicates a four-digit year. ○ mm indicates a two-digit month. ○ dd indicates a two-digit day. ○ hh24 indicates a two-digit hour (24-hour clock). ○ mi indicates two-digit minutes. ○ ss indicates two-digit seconds.
NOTAPARTITIONTABLE	Indicates the partition filter expression of a non-partitioned table.

- Select a recommended partition filter expression

  This section uses the dt partition as an example to describe how to select a recommended partition filter expression. We recommend that you specify a regular expression as the partition filter expression for a dynamic partitioned table.

i. In the **Add Partition** dialog box, click the Partition Expression field. A drop-down list appears to show you the partition filter expressions recommended by Data Quality.
   - Select a recommended partition filter expression if it meets your expectation.
   - Specify a custom partition filter expression if no recommended partition filter expressions meet your expectation.

ii. After you enter a partition expression, click **Verify**. Data Quality uses the current time, that is, the data timestamp, to calculate data and verify the partition filter expression.

iii. Click **OK**.

If you need to delete a partition filter expression, move the pointer over the partition filter expression and click the **Delete** icon to delete the partition filter expression. When you delete a partition filter expression, all rules configured based on the partition filter expression are also deleted.

Link a partition filter expression to a node

To monitor the quality of data involved in a node, you need to link a partition filter expression to the node.

- The Manage Linked Nodes dialog box lists all committed nodes. Data Quality allows you to link a partition filter expression to a node in another workspace.
- Before you link a partition filter expression to a node in another workspace, make sure that you are an administrator, a developer, or an administration expert in the two workspaces.

You can link a partition filter expression to one or more nodes. After nodes are linked, Data Quality can automatically monitor linked nodes.

> **Note** Data Quality allows you to flexibly link a partition filter expression to a node. You can select a node that is not related to your table.

1. On the rule configuration page of a table, click **Manage Linked Nodes**.
2. In **Manage Linked Nodes** dialog box, enter the name of the node that you want to link to the partition filter expression.
3. Click **Create**.

Create a rule

The Monitoring Rules page is the most important part of Data Quality, where you can create rules for your tables.

Data Quality allows you to create template rules and custom rules as needed. If you need to create a template rule or a custom rule, you can click **Add Monitoring Rule** or **Quick Create**. For more information, see Rules.

After rules are configured, you can click **Batch Create** to save all the configured rules for the current partition filter expression.

Creation method	Parameter	Description
	Rule Name	The name of the rule.

Creation method	Parameter	Description
Add Monitoring Rule	**Rule Type**	The type of the rule. Valid values: • **Rule Type**: If a node reaches the error threshold, Data Quality reports an error alert and determines that the node fails. If a node reaches the warning threshold, Data Quality reports a warning alert and determines that the node is successful. • **Soft**: If a node reaches the error threshold, Data Quality reports an error alert and determines that the node is successful. If a node reaches the warning threshold, Data Quality does not report a warning alert and determines that the node is successful.
	Auto-Generated Threshold	You can use the dynamic threshold feature only in DataWorks Enterprise Edition or higher.
	Rule Source	The source of the rule. Valid values: **Built-in Template** and **Rule Templates**.
	Field	The fields to be monitored. You can select **All Fields in Table** or a specific field. If you select a field, you can apply the rule to the specified field in the table. **Note** In this example, select All Fields in Table and set other parameters for the table-specific rule.
	Template	• If you set **Rule Source** to **Built-in Template**, the built-in table-specific rules appear. • If you set **Rule Source** to **Rule Templates**, you must set parameters, such as **Sampling Method** and **Set Flag**.
	Comparison Method	The comparison method of the rule. Valid values: **Absolute Value**, **Raise**, and **Drop**.
	Thresholds	The warning threshold and error threshold of the fluctuation. You can adjust the slider to specify thresholds or directly enter thresholds.
	Description	The description of the rule.
	Rule Name	The name of the rule.

Creation method	Parameter	Description
Quick Create	**Field**	The fields to be monitored. You can select All Fields in Table or a specific field. If you select a field, you can apply the rule to the specified field in the table.
	Trigger	• The trigger condition of the rule. If you select All Fields in Table for the Field parameter, you can set this parameter to **The number of columns is greater than 0** or **Table row number dynamic threshold**. **Notice** You can use the dynamic threshold feature only in DataWorks Enterprise Edition or higher. • If you select a field for the Field parameter, you can select **The field value already exists**, **Null Field**, or **Unique value dynamic threshold**. **Notice** You can use the dynamic threshold feature only in DataWorks Enterprise Edition or higher.

Test rules

After rules are configured for a partition filter expression, you can test all these rules and view the test results.

Note You can manually run these rules to test their configuration and notification methods. We recommend that you test rules as required.

1. On the rule configuration page of a table, click **Test**.
2. In the **Test** dialog box, set the **Data Timestamp** parameter.

Parameter	Description
Partition	The partition filter expression for which rules are run. The actual partition key varies with the data timestamp. For a non-partitioned table, use NOPARTITIONTABLE as the partition filter expression.
Data Timestamp	The data timestamp for testing rules. The default value is the current time.

3. Click **Test**.
4. In the Test dialog box, click **The test is complete. Click to view the results** to view the test results on the **Node Query** page.

Manage subscriptions

By default, Data Quality sends notifications to the user who created a partition filter expression. You can add other users so that Data Quality sends notifications to them.

1. On the rule configuration page of a table, click **Manage Subscriptions**.
2. In the **Manage Subscriptions** dialog box, specify the notification method and notification receiver.

   Data Quality supports the following four methods: **Email**, **Email and SMS**, **DingTalk Chatbot**, and **DingTalk Chatbot @ALL**.

   > **Note** Add a DingTalk chatbot and obtain a webhook URL. Then, copy the webhook URL to the Manage Subscriptions dialog box.

3. Click **Save**.

View operational logs

On the rule configuration page of a table, click **View Operation Log**. In the **Operations Logs** right-side pane, you can view the information about each operation, including the user who performed the operation, the time when the operation was performed, and the operation details.

The **Details** column displays the details of each operation performed on the current partition filter expression, including the rule configuration details.

View check results

On the rule configuration page of a table, click **View Check Results** to go to the **Node Query** page. On this page, you can view the check results for all rules under the current partition filter expression.

Clone rules

1. On the rule configuration page of a table, click **Clone Rules**.
2. In **Clone Rules** dialog box, set the **Target Expression** parameter.
3. Select **Clone Subscribers** or **Change Table Names in Custom Rules** as required.
4. Click **Clone**.

11.3.2. Configure monitoring rules for DataHub

The Monitoring Rules page is the most important part of Data Quality, where you can configure rules to monitor data in E-MapReduce (EMR), MaxCompute, and DataHub. This topic describes how to configure monitoring rules for DataHub.

Context

DataHub monitoring supports the following features:

- Templates for monitoring stream discontinuity and data latency
- Stream processing features, such as custom Flink SQL, dimension table JOIN, multi-stream JOIN, and window functions

Procedure

1. Add a DataHub data source.
   i. Log on to the DataWorks console.

ii. On the **DataStudio** page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.

iii. On the Data Integration page, click **Connection** in the left-side navigation pane. The **Data Source** page appears.

iv. Click **New data source** in the upper-right corner to add a DataHub data source. For more information, see Configure a DataHub connection.

2. Select the DataHub data source.

i. On the current page, click the icon in the upper-left corner and choose **All Products > Data Quality**.

ii. On the page that appears, click **Monitoring Rules** in the left-side navigation pane.

iii. On the Monitoring Rules page, select **Datahub** from the **Engine/Data Source** drop-down list and select the newly added DataHub data source from the Engine/Database Instance drop-down list. All the topics in the selected DataHub data source are displayed.

Parameter	Description
Configure Flink/SLS Resources	After you add a data source, click Configure Flink/SLS Resources to configure Realtime Compute and Log Service resources related to the data source.
Topics	The Topics tab lists all the topics in the DataHub data source. You can click the following buttons in the Actions column for a topic: ■ **View Monitoring Rules**: Click it to create rules for the topic. You can create template rules and custom rules. ■ **Manage Subscriptions**: Click it to view and modify subscribers to the topic, and change the notification method. You can use a DingTalk chatbot to receive notifications. The changed notification method takes effect for all subscribers to the topic.
Dimension Tables	When you create custom rules for a topic, you can create and join dimension tables. If the collected data streams lack some fields for a dimension table, you must supplement fields to data streams before data analysis and declare the dimension table in Data Quality. DataHub supports the dimension tables of ApsaraDB for HBase, Lindorm, ApsaraDB RDS, Tablestore, Taobao Distributed Data Layer (TDDL), and MaxCompute. Flink SQL does not design the data definition language (DDL) syntax for dimension tables. You can use the standard CREATE TABLE statement. However, you must add `period for system_time` to specify the period of a dimension table and declare that the dimension table stores time-varying data. **Note** When you declare a dimension table, you must specify the primary key. When you join a dimension table with another table, the ON condition must contain an equivalence condition for each primary key of the tables.

iv. Click the **Topics** tab. Find the topic for which you want to configure monitoring rules and click **View Monitoring Rules** in the Actions column.

3. On the rule configuration page of the topic, click **Create Rule**.
4. Create a monitoring rule.

   In Data Quality, you can create template rules and custom rules.

   - On the Template Rules tab of the Create rules panel, click **Create Template Rule**. Two templates are available: **Data Delay** and **Stream Discontinuity**.

     For example, you can select **Data Delay** for the Template Type parameter.

Parameter	Description
Rule Name	The name of the rule. The name can be a maximum of 255 characters in length.
Field Type	The fields to be monitored. By default, this parameter is set to All Fields in Table.
Template Type	■ **Data Delay**: monitors the interval between the time when data is generated and the time when data is written to DataHub based on the data timestamp field. If the interval exceeds a specified threshold, an alert is generated. **Note** ■ Before you configure a stream discontinuity rule, you must activate Realtime Compute in Flink and create a project. ■ The data timestamp field supports two data types: TIMESTAMP and STRING (yyyy -MM -dd H dd HH:mm:ss). ■ **Stream Discontinuity**: monitors the period during which no data is written to DataHub. If the period exceeds a specified threshold, an alert is generated.

Parameter	Description
Alerts Threshold	The maximum number of alerts generated for data latency. Data Quality reports an alert when the number of alerts generated for data latency exceeds this threshold. This parameter is displayed only when you select Data Delay for the Template Type parameter.
Data Timestamp Field	The data timestamp field of the topic for which the rule is created. This field supports two data types: TIMESTAMP and STRING (yyyy-MM-dd HH:mm:ss). This parameter is displayed only when you select Data Delay for the Template Type parameter.
Alert Frequency	The interval at which alerts are reported. You can set the alert interval to 10 minutes, 30 minutes, 1 hour, or 2 hours.
Warning Threshold	The warning threshold, in seconds. The value must be an integer and less than the error threshold.
Error Threshold	The error threshold, in seconds. The value must be an integer and greater than the warning threshold.

- If template rules do not meet your requirements for monitoring the data quality of DataHub topics, you can create a custom rule. On the Custom Rules tab of the Create rules panel, click **Create Custom Rule**.

  > **Note**
  > - The field in the SELECT clause must be a column. Make sure that you can compare the field values with the warning threshold and error threshold.
  > - The FROM clause must include the current topic and all its columns.

Parameter	Description
Rule Name	The name of the rule. The name must be unique in the topic and can be a maximum of 20 characters in length.

Parameter	Description
Script	The custom SQL script that is used to set a rule. The return value of the SELECT clause must be unique. You can refer to the following sample statements: ■ Use a simple SQL statement. `select id as a from zmr_tst02;` ■ Join the topic and a dimension table named test_dim. `select e.id as eid` `from zmr_test02 as e` `join test_dim for system_time as of proctime() as w` `on e.id=w.id` ■ Join the topic and another topic named dp1test_zmr01. `select count(newtab.biz_date) as aa` `from (select o.*` `from zmr_test02 as o` `join dp1test_zmr01 as p` `on o.id=p.id)newtab` `group by` `id.biz_date,biz_date_str,total_price,'timestamp'`
Warning Threshold	The warning threshold, in minutes. The value must be an integer and less than the error threshold.
Error Threshold	The error threshold, in minutes. The value must be an integer and greater than the warning threshold.
Minimum Alert Interval	The minimum interval at which alerts are reported, in minutes.
Description	The description of the rule.

5. Click **Batch Create**. After rules are created for the topic, you can perform the following operations:
   - **View Log**: Click it to view the operational logs of the rules.
   - **Manage Subscriptions**: Click it to view and modify subscribers to the rules, and change the notification method. The changed notification method takes effect for all the subscribers of the rules.

     Data Quality supports the following methods: **Email**, **Email and SMS**, **DingTalk Chatbot**, and **DingTalk Chatbot @ALL**.

     > **Note** Add a DingTalk chatbot and obtain the webhook address of the chatbot. Then, copy the webhook address to the Manage Subscriptions dialog box.

12.Data Map

12.1. Overview

Data Map is developed based on Data Management and uses roles to control the permissions for using different features, such as the permissions for creating and previewing data. Data Map helps you build a better enterprise-level knowledge base.

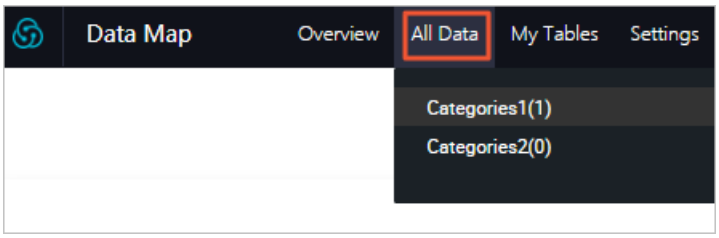
On the homepage of Data Map, you can enter keywords to search for tables by name. You can also click a table in one of the following sections to view the table data: **Recently Viewed Tables**, **Recently Read Tables**, **Most Viewed Tables**, or **Most Read Tables**.

- If you prefer a powerful search engine, go to the homepage to search for data.

  > **Note** The homepage appears when you go to the **Data Map** page. To return to the homepage from other pages, click **Data Map** in the upper-left corner.

- If you want to find tables by workspace or cluster, click **All Data** in the top navigation bar. On the page that appears, you can view tables on different tabs, such as **MaxCompute** or **EMR**. You can also find a table and perform the following operations on it: **Add to Favorites**, **Apply for Permission**, **View Lineage**, or **View DDL Statement**.

  If you have added tables to categories, move the pointer over **All Data** in the top navigation bar and select a category. Tables in the category appear. For more information, see Manage categories of and permissions on MaxCompute tables.

- If you want to view the overall data of the current tenant, click **Overview** in the top navigation bar. For more information, see View overall data.
- If you want to modify tables that are owned by your account, click **My Data** in the top navigation bar. For more information, see View and manage tables and data permissions.
- If you are a category administrator or workspace administrator and want to modify the workspace configurations or global categories, click **Configuration Management** in the top navigation bar. For more information, see Manage categories of and permissions on MaxCompute tables.

12.2. Configure whitelists and category management permissions

If you want to view MaxCompute table data or collect metadata in Data Map, you must configure a whitelist for your MaxCompute project or the desired data source. Then, add the Classless Inter-Domain Routing (CIDR) blocks of the region where your DataWorks workspace resides to the whitelist. If you want to manage categories in Data Map, you must grant the related permissions to the account you use. This topic describes how to configure the whitelists and grant category management permissions.

Context

- Data Map provides a platform to manage both metadata and the data assets of enterprises. This platform allows you to search global data, view metadata details, preview data, view data lineages, and manage data categories. Data Map helps you search for, understand, and use data. If you want to use Data Map to view the table data in a MaxCompute project, check whether a whitelist is configured for the project. If a whitelist is configured, make sure that the CIDR blocks of the region where your DataWorks workspace resides are in the whitelist. Otherwise, you cannot view the table data in Data Map. Therefore, to ensure that Data Map can access MaxCompute projects, you must configure whitelists for the projects in advance.

  > **Note** Only MaxCompute requires whitelist configuration. For other products, you can directly view data in Data Map.

- The metadata collection feature allows you to collect metadata from different data sources. This way, you can manage the metadata in a centralized manner. After the metadata of a data source is collected, you can view the metadata in Data Map. Before you collect metadata from the data source, check whether a whitelist is configured for the data source. If a whitelist is configured, make sure that the CIDR blocks of the region where your DataWorks workspace resides are in the whitelist.
- The category management feature allows you to effectively organize and manage tables by category. For more information, see Manage categories of and permissions on MaxCompute tables. Before you use this feature, make sure that you have the required permissions.
  - If you use an Alibaba Cloud account to manage categories, you have the permissions by default.
  - If you use a RAM user to manage categories, you must attach the **AliyunDataWorksFullAccess** policy to the RAM user.

Configure a whitelist for a MaxCompute project to allow Data Map to access the project

1. Check whether a whitelist is configured for your MaxCompute project. For more information, see *View an IP address whitelist* in the MaxCompute documentation.

   If a whitelist is not configured for the project, Data Map can access the table data in the project. If a whitelist is configured, proceed to the next step.

2. Configure the whitelist.

   Add the desired CIDR blocks of the region where your DataWorks workspace resides to the whitelist. The following table lists the CIDR blocks of each region. For more information about how to configure a whitelist, see *Configure IP address whitelists*.

Region	CIDR block or IP address
China (Hangzhou)	100.64.0.0/10,11.193.102.0/24,11.193.215.0/24,11.194.110.0/24,11.194.73.0/24,118.31.157.0/24,47.97.53.0/24,11.196.23.0/24,47.99.12.0/24,47.99.13.0/24,114.55.197.0/24,11.197.246.0/24,11.197.247.0/24

Region	CIDR block or IP address
China (Shanghai)	11.193.109.0/24,11.193.252.0/24,47.101.107.0/24,47.100.129.0/24,106.15.14.0/24,10.117.28.203,10.143.32.0/24,10.152.69.0/24,10.153.136.0/24,10.27.63.15,10.27.63.38,10.27.63.41,10.27.63.60,10.46.64.81,10.46.67.156,11.192.97.0/24,11.192.98.0/24,11.193.102.0/24,11.218.89.0/24,11.218.96.0/24,11.219.217.0/24,11.219.218.0/24,11.219.219.0/24,11.219.233.0/24,11.219.234.0/24,118.178.142.154,118.178.56.228,118.178.59.233,118.178.84.74,120.27.160.26,120.27.160.81,121.43.110.160,121.43.112.137,100.64.0.0/10,10.117.39.238
China (Shenzhen)	100.106.46.0/24,100.106.49.0/24,10.152.27.0/24,10.152.28.0/24,11.192.91.0/24,11.192.96.0/24,11.193.103.0/24,100.64.0.0/10,120.76.104.0/24,120.76.91.0/24,120.78.45.0/24,47.106.63.0/26,47.106.63.128/26,47.106.63.192/26,47.106.63.64/26
China (Chengdu)	11.195.52.0/24,11.195.55.0/24,47.108.22.0/24,100.64.0.0/10
China (Zhangjiakou)	11.193.235.0/24,47.92.22.0/24,100.64.0.0/10
China (Hong Kong)	10.152.162.0/24,11.192.196.0/24,11.193.11.0/24,100.64.0.0/10,47.89.61.0/24,47.91.171.0/24,11.193.118.0/24,47.75.228.0/24,47.56.45.0/25,47.244.92.128/25,47.101.109.0/24
Singapore (Singapore)	100.106.10.0/24,100.106.35.0/24,10.151.234.0/24,10.151.238.0/24,10.152.248.0/24,11.192.153.0/24,11.192.40.0/24,11.193.8.0/24,100.64.0.0/10,47.88.147.0/24,47.88.235.0/24,11.193.162.0/24,11.193.163.0/24,11.193.220.0/24,11.193.158.0/24,47.74.162.0/24,47.74.203.0/24,47.74.161.0/24,11.197.188.0/24
Australia (Sydney)	11.192.100.0/24,11.192.134.0/24,11.192.135.0/24,11.192.184.0/24,11.192.99.0/24,100.64.0.0/10,47.91.49.0/24,47.91.50.0/24,11.193.165.0/24,47.91.60.0/24
China (Beijing)	100.106.48.0/24,10.152.167.0/24,10.152.168.0/24,11.193.50.0/24,11.193.75.0/24,11.193.82.0/24,11.193.99.0/24,100.64.0.0/10,47.93.110.0/24,47.94.185.0/24,47.95.63.0/24,11.197.231.0/24,11.195.172.0/24,47.94.49.0/24,182.92.144.0/24,39.99.77.0/26,39.99.77.64/26,39.99.77.128/26,39.104.220.192/26,39.107.7.0/26,39.107.7.64/26,182.92.32.128/26,182.92.32.192/26
US (Silicon Valley)	10.152.160.0/24,100.64.0.0/10,47.89.224.0/24,11.193.216.0/24,47.88.108.0/24
US (Virginia)	47.88.98.0/26,47.88.98.64/26,47.88.98.128/26,47.88.98.192/26,47.252.91.0/26,47.252.91.64/26,47.252.91.128/26,47.252.91.192/26,10.128.134.0/24,11.193.203.0/24,11.194.68.0/24,11.194.69.0/24,100.64.0.0/10
Malaysia (Kuala Lumpur)	11.193.188.0/24,11.221.205.0/24,11.221.206.0/24,11.221.207.0/24,100.64.0.0/10,11.214.81.0/24,47.254.212.0/24,11.193.189.0/24
Germany (Frankfurt)	11.192.116.0/24,11.192.168.0/24,11.192.169.0/24,11.192.170.0/24,11.193.106.0/24,100.64.0.0/10,11.192.116.14,11.192.116.142,11.192.116.160,11.192.116.75,11.192.170.27,47.91.82.22,47.91.83.74,47.91.83.93,47.91.84.11,47.91.84.110,47.91.84.82,11.193.167.0/24,47.254.138.0/24

Region	CIDR block or IP address
Japan (Tokyo)	100.105.55.0/24,11.192.147.0/24,11.192.148.0/24,11.192.149.0/24,100.64.0.0/10,47.91.12.0/24,47.91.13.0/24,47.91.9.0/24,11.199.250.0/24,47.91.27.0/24,11.59.59.0/24,47.245.51.128/26,47.245.51.192/26,47.91.0.128/26,47.91.0.192/26
UAE (Dubai)	11.192.107.0/24,11.192.127.0/24,11.192.88.0/24,11.193.246.0/24,47.91.116.0/24,100.64.0.0/10
India (Mumbai)	11.194.10.0/24,11.246.70.0/24,11.246.71.0/24,11.246.73.0/24,11.246.74.0/24,100.64.0.0/10,149.129.164.0/24,11.194.11.0/24,11.59.62.0/24,147.139.23.0/26,147.139.23.128/26,147.139.23.64/26,149.129.165.192/26
UK (London)	11.199.93.0/24,100.64.0.0/10
Indonesia (Jakarta)	11.194.49.0/24,11.200.93.0/24,11.200.95.0/24,11.200.97.0/24,100.64.0.0/10,149.129.228.0/24,10.143.32.0/24,11.194.50.0/24,11.59.135.0/24,147.139.156.0/26,147.139.156.128/26,147.139.156.64/26,149.129.230.192/26
China North 2 Ali Gov 1	11.194.116.0/24,100.64.0.0/10,39.107.188.202 If the preceding CIDR blocks cannot be added, add the following information: 11.194.116.160,11.194.116.161,11.194.116.162,11.194.116.163,11.194.116.164,11.194.116.165,11.194.116.167,11.194.116.169,11.194.116.170,11.194.116.171,11.194.116.172,11.194.116.173,11.194.116.174,11.194.116.175,39.107.188.0/24.
China East 2 Finance	140.205.46.128/25,140.205.48.0/25,140.205.48.128/25,140.205.49.0/25,140.205.49.128/25,11.192.156.0/25,11.192.157.0/25,11.192.164.0/25,11.192.165.0/25,11.192.166.0/25,11.192.167.0/25,106.11.245.0/26,106.11.245.128/26,106.11.245.192/26,106.11.245.64/26,140.205.39.0/24,106.11.225.0/24,106.11.226.0/24,106.11.227.0/24,106.11.242.0/24,100.104.8.0/24

Configure a whitelist for metadata collection from a data source

1. Check whether a whitelist is configured for the data source.

   Data Map allows you to collect metadata from the following types of data sources:

   - Collect metadata from an EMR data source
   - Collect metadata from an AnalyticDB for PostgreSQL data source
   - Collect metadata from a MySQL data source
   - Collect metadata from a PostgreSQL data source
   - Collect metadata from an SQL Server data source
   - Collect metadata from an Oracle data source
   - Collect metadata from an AnalyticDB for MySQL 2.0 data source
   - Collect metadata from an AnalyticDB for MySQL 3.0 data source
   - Collect metadata from a Hologres data source

   The method used to check whether a whitelist is configured varies based on the data source type. You can consult technical support personnel.

If a whitelist is not configured for the data source, you can directly use Data Map to collect metadata from the data source. If a whitelist is configured, proceed to the next step.

2. Configure the whitelist.

   Add the desired CIDR blocks of the region where your DataWorks workspace resides to the whitelist. The following table lists the CIDR blocks of each region. The position where the CIDR blocks are added varies based on the data source type. You can consult technical support personnel.

Region	CIDR block or IP address
China (Shanghai)	100.104.189.64/26,11.115.110.10/24,11.115.109.9/24,47.102.181.128/26,47.102.181.192/26,47.102.234.0/26,47.102.234.64/26,100.104.38.192/26
China (Hangzhou)	100.104.135.128/26,11.193.215.233/24,11.194.73.32/24,118.31.243.0/26,118.31.243.64/26,118.31.243.128/26,118.31.243.192/26,100.104.242.0/26
China (Shenzhen)	100.104.46.128/26,11.192.91.119/24,120.77.195.128/26,120.77.195.192/26,120.77.195.64/26,47.112.86.0/26,100.104.138.128/26
China (Beijing)	100.104.37.128/26,11.193.82.20/24,11.197.254.171/24,39.107.223.0/26,39.107.223.64/26,39.107.223.128/26,39.107.223.192/26,100.104.152.128/26
China (Chengdu)	100.104.88.64/26,11.195.57.28/24,47.108.46.0/26,47.108.46.64/26,47.108.46.128/26,47.108.46.192/26,100.104.248.128/26
China (Zhangjiakou)	100.104.197.0/26,11.193.236.121/24,47.92.185.0/26,47.92.185.64/26,47.92.185.128/26,47.92.185.192/26,100.104.75.64/26

Configure category management permissions for a RAM user

If you use a RAM user to manage categories, you must attach the **AliyunDataWorksFullAccess** policy to the RAM user.

What's next

After the whitelists and category management permissions are configured, you can view MaxCompute table data, collect metadata, or manage categories in Data Map.

12.3. View overall data

This topic describes how to view the overall data of a tenant on the Overview page.

Procedure

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > DataMap(Data Management)**.
3. In the top navigation bar, click **Overview**.

   The **Overview** page displays the offline statistics of the current tenant.

   > **Note** The data on the Overview page is generated on the previous day.

Section	Description
Total Number of Projects	The total number of projects of the specified type for the current tenant.
Total Number of Tables	The total number of tables in the destination project or destination database for the current tenant.
Storage	The total storage space that is occupied by all the tables in the destination MaxCompute project for the current tenant.
Storage trend chart	The offline statistics on the trend of storage usage.
Top Projects by Table Storage	The top projects that occupy the most storage space for the current tenant.
Top Tables by Occupied Storage	The top tables that occupy the most storage space for the current tenant. You can click a table name to go to the details page of the table. **Note** The logical storage space that is occupied by projects and tables is calculated in a T+1 manner. The numbers next to the project and table names indicate the sizes of the occupied logical storage space. The project storage volume includes not only the table storage volume but also the storage volumes of resources, data in the recycle bin, and other system files. Therefore, the project storage volume is larger than the table storage volume. You are charged for the logical storage volume of a table rather than the physical storage volume of a table.
Most Frequently Used Tables	The most frequently referenced tables for the current tenant. You can click a table name to go to the details page of the table.

12.4. View and manage tables and data permissions

This topic describes how to view and manage tables on the Owned by Me, Managed by Me, Managed by Tenant Account, and My Favorites pages. This topic also describes how to view and manage data permissions.

Context

Data Map updates data one day after the data is generated. If you want to query real-time data, we recommend that you use SQL statements.

Go to the Owned by Me page

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products >**

Data governance > DataMap(Data Management).

3. In the top navigation bar, click **My Data**. The **Owned by Me** page appears.

View and manage tables on the Owned by Me page

On the **Owned by Me** page, you can search for your desired table by keyword, environment, project or data source, and visibility. You can also view the details about a table and perform operations on the table.

Parameter	Description
Table Name	The name of the table. You can click a table name to go to the details page of the table.
Display Name	The display name of the table. You can click the icon in the **Display Name** column of a table to modify the display name of the table.
Project/Data Store	The name of the project or data source to which the table belongs. The suffix of the name varies based on the environment where the table resides. For example, **_dev** indicates that the table resides in the development environment.
Hide or Show	You can click the icon in the Hide or Show column of a table to display or hide the table. Valid values: **Show**, **Hide**, and **Within the Project Only**.
TTL (Days)	The TTL of the table. The value is the same as that you set when you created the table.
Environment	The environment where the table resides. Valid values: **Development** and **Production**.
Storage	The volume of data that is stored in the table.
Favorites	The number of times that users add the table to favorites.
Views in Last 30 Days	The number of times that users view the table in the last 30 days.
Created At	The time when the table was created.
Actions	The operations that you can perform on the table. You can click **Delete** or **Change Category** in the Actions column of a table to delete the table or change the category of the table.
Edit, Change Owner, Delete, and Change Category	The operations that you can perform on multiple tables at a time. You can select tables and click **Edit**, **Change Owner**, **Delete**, or **Change Category** to modify the tables, change the owners of the tables, delete the tables, or change the categories of the tables.

View and manage tables on the Managed by Me page

In the left-side navigation pane, click **Managed by Me**. On the page that appears, you can search for your desired table by keyword, project or data source, and environment. You can also view the details about a table and perform operations on the table.

Parameter	Description
Table Name	The name of the table. You can click a table name to go to the details page of the table.
Display Name	The display name of the table. You can click the icon in the **Display Name** column of a table to modify the display name of the table.
Project/Data Store	The name of the project or data source to which the table belongs. The suffix of the name varies based on the environment where the table resides. For example, **_dev** indicates that the table resides in the development environment.
TTL (Days)	The TTL of the table. The value is the same as that you set when you created the table.
Environment	The environment where the table resides. Valid values: **Development** and **Production**.
Storage	The volume of data that is stored in the table.
Favorites	The number of times that users add the table to favorites.
Views in Last 30 Days	The number of times that users view the table in the last 30 days.
Created At	The time when the table was created.
Actions	The operations that you can perform on the table. You can click **Delete** or **Change Category** in the Actions column of a table to delete the table or change the category of the table.
Edit, Change Owner, Delete, and Change Category	The operations that you can perform on multiple tables at a time. You can select tables and click **Edit**, **Change Owner**, **Delete**, or **Change Category** to modify the tables, change the owners of the tables, delete the tables, or change the categories of the tables.

View and manage tables on the Managed by Tenant Account page

In the left-side navigation pane, click **Managed by Tenant Account**. On the page that appears, you can search for your desired table by keyword and project or data source and view the details about a table.

Parameter	Description
Table Name	The name of the table. You can click a table name to go to the details page of the table.
Display Name	The display name of the table.

Parameter	Description
Project/Data Store	The name of the project or data source to which the table belongs. You can click a project or data source name in the Project/Data Store column of a table to go to the details page of the project or data source.
TTL (Days)	The TTL of the table. The value is the same as that you set when you created the table.
Environment	The environment where the table resides. Valid values: **Development** and **Production**.
Storage	The volume of data that is stored in the table.
Favorites	The number of times that users add the table to favorites.
Views in Last 30 Days	The number of times that users view the table in the last 30 days. You can click the icon to arrange the tables by the number of views in ascending or descending order.
Created At	The time when the table was created.

View tables on the Tables of Other Tenants page

In the left-side navigation pane, click **Tables of Other Tenants**. On the page that appears, you can search for your desired table by table name and project or data source name and view the details about a table.

Parameter	Description
Table Name	The name of the table. You can click a table name to go to the details page of the table.
Display Name	The display name of the table.
Project/Data Store	The name of the project or data source to which the table belongs. You can click a project or data source name in the Project/Data Store column of a table to go to the details page of the project or data source.
Physical Storage	The volume of data that is stored in the table.
TTL (Days)	The TTL of the table. The value is the same as that you set when you created the table.
Created At	The time when the table was created.

View and manage tables on the My Favorites page

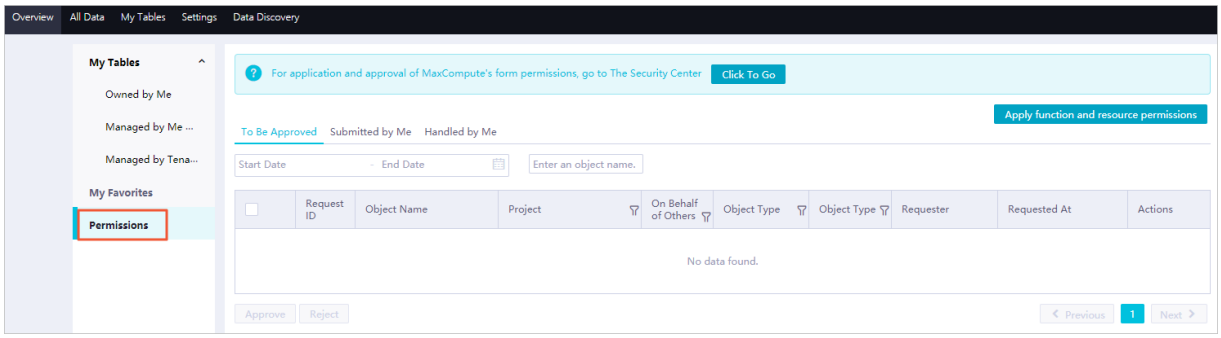
In the left-side navigation pane, click **My Favorites**. On the page that appears, you can view the tables that you have added to favorites. You can specify the following conditions to search for the tables that you have added to favorites: **Data Type**, **Project/Data Store**, and **Table Name**.

You can click **Remove from Favorites** in the Actions column of a table to remove the table from your favorites.

View and manage data permissions

In the left-side navigation pane, click **Permission Management**. On the page that appears, you can view and manage data permissions.

You can click **Apply for Function and Resource Permissions** in the upper-right corner of the **Permission Management** page to request permissions. You can also view permission request details on the **To Be Approved**, **Submitted by Me**, and **Handled by Me** tabs.



- **Apply for Function and Resource Permissions**
  i. On the Permission Management page, click **Apply for Function and Resource Permissions** in the upper-right corner.
  ii. In the **Apply for Function and Resource Permissions** dialog box, configure the parameters. The following table describes the parameters.

Parameter	Description
Object Type	The type of the object on which you want to request permissions. Valid values: **Functions** and **Resources**.
Grant To	The account to which permissions will be granted. Valid values: **Current Account** and **Specified Account**. ■ If you select **Current Account**, permissions will be granted to you after the request is approved. ■ If you select **Specified Account**, you must also set the **Username** parameter. Permissions will be granted to the specified account after the request is approved.
Project Name	The name of the MaxCompute project that contains the function or resource on which you want to request permissions. Fuzzy match is supported.
Function Name	The name of the function or resource in the project. If the resource is a file, enter the full name of the file, including the file name extension, such as my_mr.jar.
Validity Period	The validity period of the permissions, in days. If this parameter is not specified, the permissions are permanently valid. After the validity period is exceeded, the system automatically revokes the permissions.

Parameter	Description
Reason	The reason why you request the permissions.

- **To Be Approved**

  If you are the workspace administrator, you can view and approve the requests for permissions on all objects such as tables, resources, and functions in the workspace on the **To Be Approved** tab.

- **Submitted by Me**

  On the **Permission Management** page, click the **Submitted by Me** tab.

  On the **Submitted by Me** tab, you can view the permission requests that you have submitted.

- **Handled by Me**

  On the **Permission Management** page, click the **Handled by Me** tab.

  If you are the workspace administrator, you can view the permission requests that you have handled for all objects such as tables, resources, and functions in the workspace on the **Handled by Me** tab.

Manually synchronize a table

1. In the left-side navigation pane, choose **My Tools > Manually Sync Table** to go to the **Manually Sync Table** page.
2. Enter a table GUID in the **Table GUID** field. Then, click **Manually Sync Table** to synchronize the table.

12.5. Manage categories of and permissions on MaxCompute tables

This topic describes how to manage categories of and permissions on MaxCompute tables that are in your owned or managed workspaces on the Configuration Management page of Data Map.

Go to the Configuration Management page

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > DataMap(Data Management)**.
3. On the Data Map page, click **Configuration Management** in the top navigation bar. The **Manage Categories** tab is displayed.

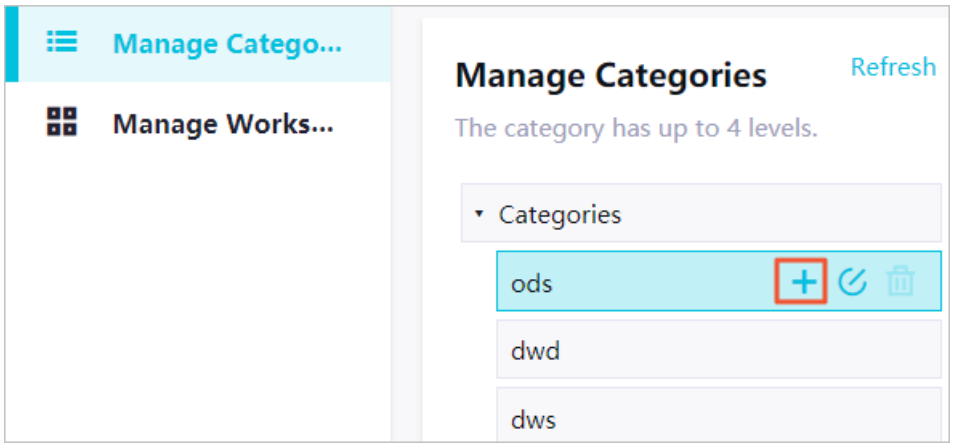
   The Configuration Management page allows you to manage categories and permissions on MaxCompute tables in a workspace.

Manage categories

On the **Manage Categories** page, you can perform the following steps to create a category and add tables to and remove tables from the category.

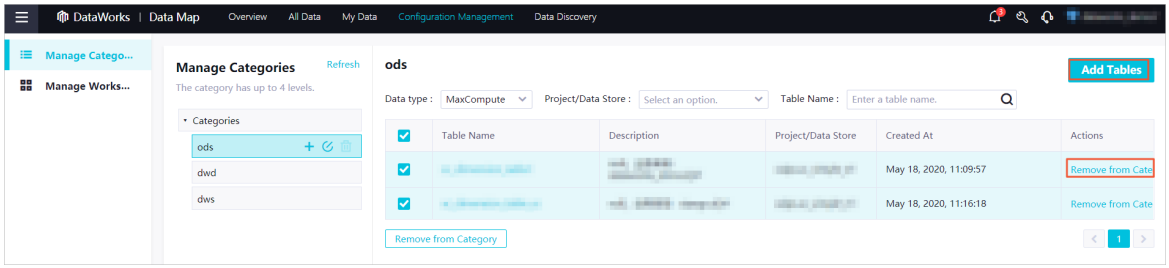
1. On the **Manage Categories** page, move the pointer over **Categories** and click the + icon. In the field that appears, enter a category name and press Enter to create a level-1 category.

2. Move the pointer over the level-1 category and click the + icon. In the field that appears, enter a category name and press Enter to create a level-2 category.



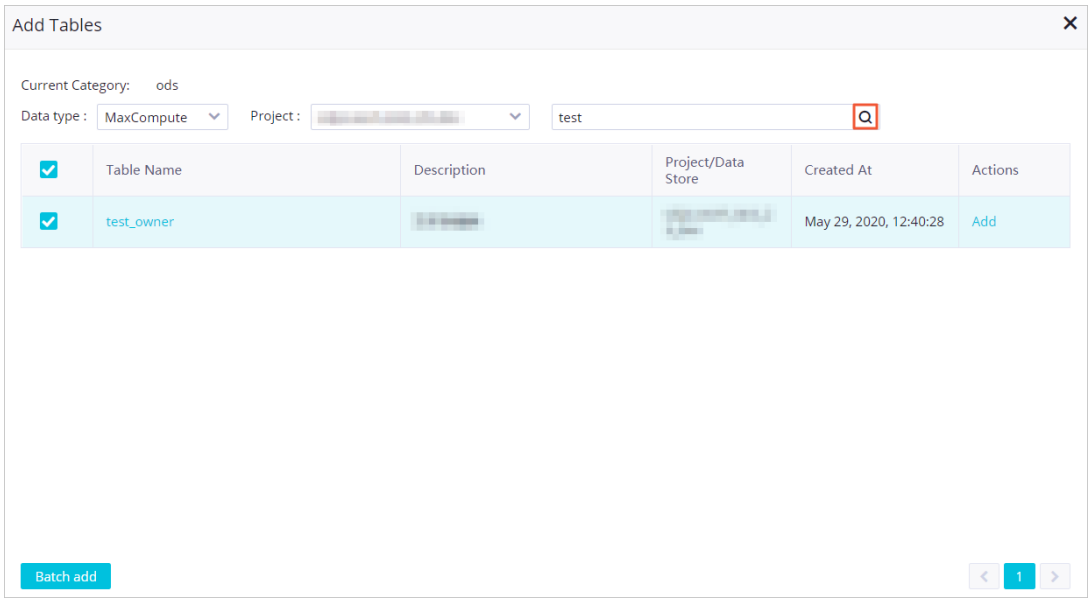
Use the same method to create more categories. DataWorks allows you to create a maximum of four levels of categories. You can click the icon to edit a category or click the icon to delete a category.

3. Add tables to and remove tables from a category.



- Add tables to a category
  a. Select the category and click **Add Tables** in the upper-right corner.

b. In the **Add Tables** dialog box, specify the table type and project, enter a table name or keyword, and then click the icon to search for tables.



c. If you want to add a table to the category, find the table and click **Add** in the Actions column.

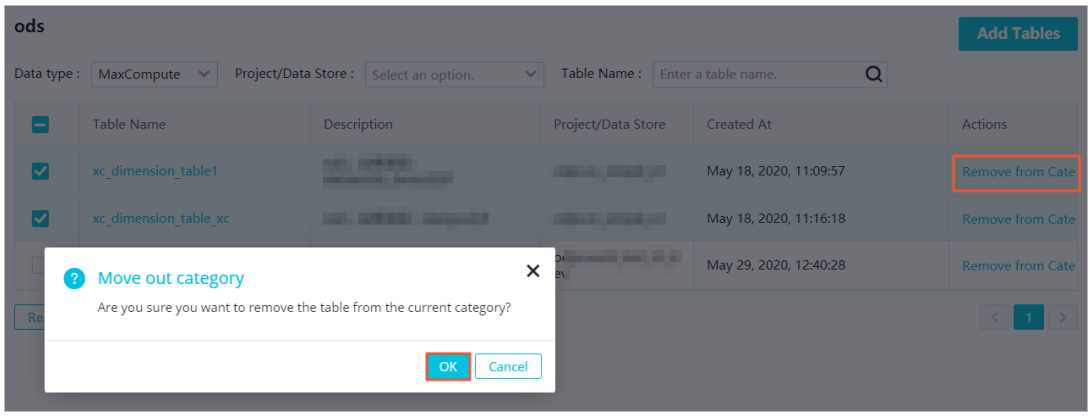
If you want to add multiple tables at a time, select the tables and click **Batch add**.

- Remove tables from a category

a. Select the category. If you want to remove a table from the category, find the table and click **Remove** in the Actions column.

If you want to remove multiple tables at a time, select the tables and click **Remove from Category**.

b. In the **Move out category** message, click **OK**.



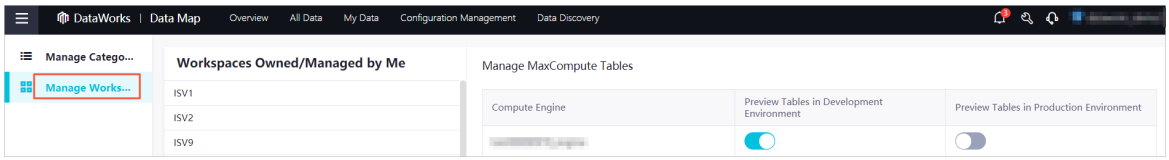
Manage permissions on MaxCompute tables

On the **Manage Workspaces** page, you can specify whether MaxCompute tables can be previewed in a compute engine in the development and production environments.

1. In the left-side navigation pane, click **Manage Workspaces**.
2. In the **Workspaces Owned/Managed by Me** section, click the workspace for which you want to

manage permissions on MaxCompute tables.

3. In the **Manage MaxCompute Tables** section, turn on or off the switch in the **Preview Tables in Development Environment** or **Preview Tables in Production Environment** column.



4. Turn on the switch in the **Preview Tables in Production Environment** column. In the **Attention** message, click **I already know that I am sure to open it**.

> **Note**
>
> - If you use a workspace in basic mode, you can turn on or off only the switch in the **Preview Tables in Production Environment** column.
> - After the switch in the **Preview Tables in Production Environment** column is turned on, all members of the workspace can preview MaxCompute tables in the production environment without requesting permissions. This may cause the leak of sensitive data. Therefore, exercise caution before you turn on the switch.

12.6. Table details

12.6.1. View the details of a table

This topic describes how to go to the details page of a table and view the details of the table, such as the basic information, output information, and lineage information.

Go to the details page of a table

1.
2.
3. In the top navigation bar, click **All Data**.
4. On the All Data page, click a tab based on your business requirements, such as MaxCompute.
5. On the tab that appears, click the name of the table that you want to view.

   On the details page that appears, you can view the **basic information**, **business information**, **permission information**, **technical information**, **detailed information**, **output information**, **lineage information**, **reference records**, and **usage notes** of the table. You can also preview data in the table.

View basic information

In the **Table Basic Information** section, you can view the numbers of times that the table is read, the table is added to favorites, and the table is viewed. You can also view the number and code of the output nodes, MaxCompute project name, region where the current workspace resides, region to which the engine belongs, owner, creation time, time-to-live (TTL), storage capacity, description, and tags of the table. You can also check whether the table is a partitioned tabled.

You can perform the following operations in the **Table Basic Information** section:

- View the code of an output node of the table: Click **View Code** next to **Output Node**. On the

Operation Center page, view the node code.

- View the details of a MaxCompute project: Click the MaxCompute project name. On the page that appears, view the details of the MaxCompute project to which the table belongs.
- Edit the description of the table: Click the icon next to **Description**, edit the description, and then click the icon.
- Add a tag to or remove a tag from the table: Click the icon next to **Tags**, enter a tag name, and then press **Enter**.

  To remove a tag from the table, move the pointer over the tag and click the icon.

View business information

In the **Business Information** section, you can view the DataWorks workspace name, environment type, category, and display name of the table.

You can perform the following operations in the **Business Information** section:

- View the details of the workspace: Click the DataWorks workspace name. On the page that appears, view the details of the DataWorks workspace to which the table belongs.
- Edit the display name of the table: Click the icon next to **Display Name**, edit the display name, and then click the icon.

View permission information

In the **Permission Information** section, you can view your permissions on the table.

To modify your permissions on the table, perform the following steps:

1. Click **View More** in the upper-right corner of the **Permission Information** section. Then, the Permission application tab of the Data access control page appears.
2. On the **Permission application** tab, specify **User**, **Application duration**, and **Reason for application** in the Application Information section.

   > **Note** If you do not specify **Application duration**, the permissions that you request will be permanently valid after your request is approved.

3. Click **Apply for permission**.

View technical information

In the **Technical Information** section, you can view the technical type, time when the data definition language (DDL) statement was last modified, time when data was last modified, time when data was last viewed, and compute engine information.

In the **Technical Information** section, you can click **View** next to **Compute Engine Information**. In the **Compute Engine Information** dialog box, you can view or copy the information about the compute engine.

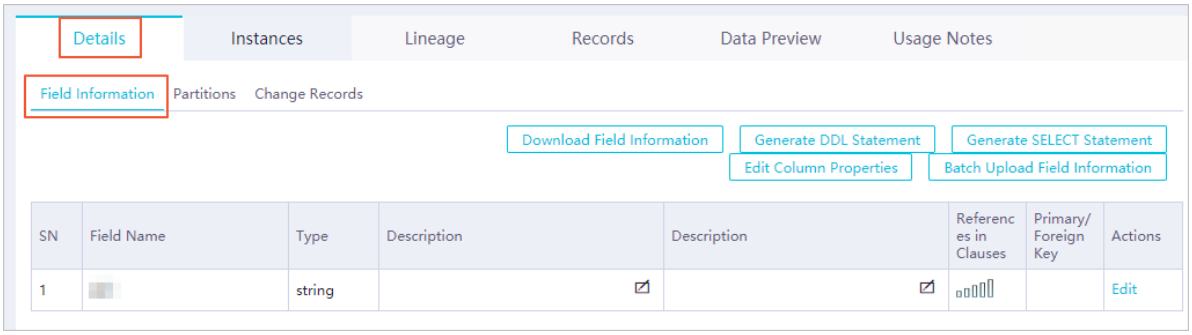
> **Note** By default, the time format yyyy-MM-dd HH:mm:ss is used to describe the compute engine.

View detailed information

The **Details** tab contains the following subtabs: **Field Information**, **Partitions**, and **Change Records**.

- **Field Information** subtab

  On the **Field Information** subtab, you can view the name, data type, description, business description, and popularity of fields. You can also check whether a field is a primary key or a foreign key.



Button	Description
Edit	Click this button, modify the field description and business description, and then click **Saved** or **Cancel**.
Upload	Click this button and drag the file that you want to upload from your on-premises machine to the **Batch Upload Field Information** dialog box.
Download	Click this button to download the field information of the current table.
Generate DDL Statement	Click this button. In the **Generate DDL Statement** dialog box, view or copy the DDL statement used to create the current table.
Generate SELECT Statement	Click this button. In the **Generate SELECT Statement** dialog box, view or copy the `SELECT` statement used to query data in the current table.

- **Partitions** subtab

  On the **Partitions** subtab, you can view the name, number of records, storage capacity, creation time, and last update time of each partition in the current table.

- **Change Records** subtab

  On the **Change Records** subtab, you can view the description, type, granularity, time, and operator of changes performed on the current table.

  On this subtab, you can also select a change type from the drop-down list in the upper-left corner to filter the table changes.

  Change types include **Create Table**, **Modify Table**, **Delete Table**, **Create Partition**, **Delete Partition**, **Change Owner**, and **Change TTL**.

View output information

If the table data periodically changes with the related node, you can view the change status and data that is continuously updated on the **Instances** tab.

On this tab, you can also click **View Code** or **View Log** in the **Actions** column of a node to view the code or logs of the node.

View lineage information

On the Lineage tab, you can view the source and destination of data and manage the lineage information with ease.

The **Lineage** tab contains the following subtabs: **Table Lineage**, **Field Lineage**, and **Impact Analysis**.

- The **Table Lineage** subtab consists of the **Graph Analysis** and **View by Level** parts.
  - **Graph Analysis**: displays the ancestor and descendant tables of a specified level for the current table and the number of ancestor and descendant tables for each table.
  - **View by Level**: displays the parent and child tables at one level of the current table by default. You can search for the parent and child tables based on the globally unique identifier (GUID).
- On the **Field Lineage** subtab, you can select a field from the **Field Name** drop-down list to view the lineage information of the field.
- On the **Impact Analysis** subtab, you can query the node that generates a lineage and the full link of the lineage based on information such as the lineage level, field, node type, table name, workspace name, and table owner.

  You can click **Manual update** to rerun the impact analysis. You can also download the impact analysis result or send the impact analysis result to the owners of descendant tables of the current table by email.

View reference records

The **Records** tab contains the following subtabs: **Foreign Key References** and **Access Statistics**.

- **Foreign Key References** subtab: On this subtab, you can check the number of users who reference the current table.
- **Access Statistics** subtab: On this subtab, you can view the reference records in a line chart.

Preview data

On the **Data Preview** tab, you can preview the data of the current table.

> **Notice** Only authorized users can preview tables in the production environment. If you do not have the required permissions, click **Apply Now**.

View usage notes

On the **Usage Notes** tab, you can edit usage notes, check the historical versions of the usage notes, and view the business description of data.

12.6.2. Request permissions on tables

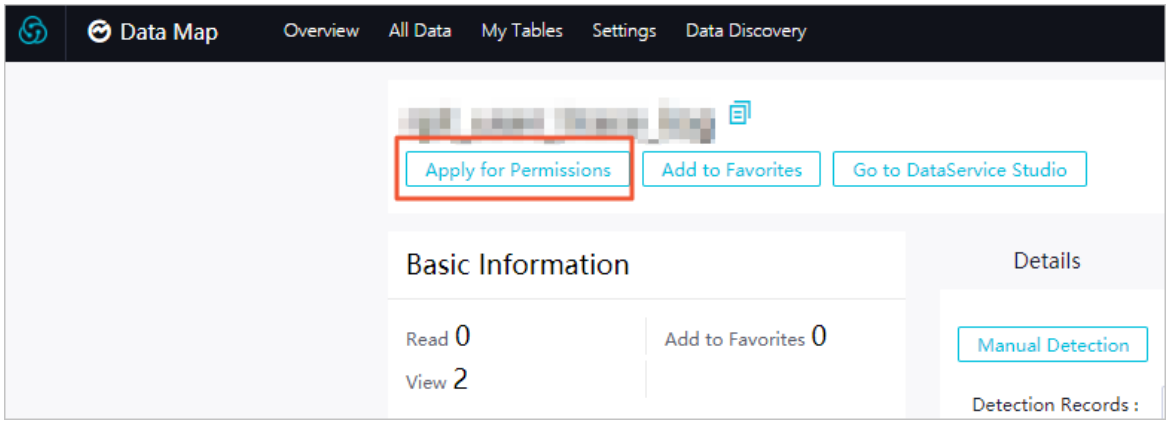
This topic describes how to request permissions on tables in Security Center or Data Map.

Go to the details page of a table

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > DataMap(Data Management)**.
3. In the top navigation bar, click **All Data**.
4. On the All Data page, click a tab based on your business requirements, such as MaxCompute.
5. On the tab that appears, click the name of the table on which you want to request permissions.

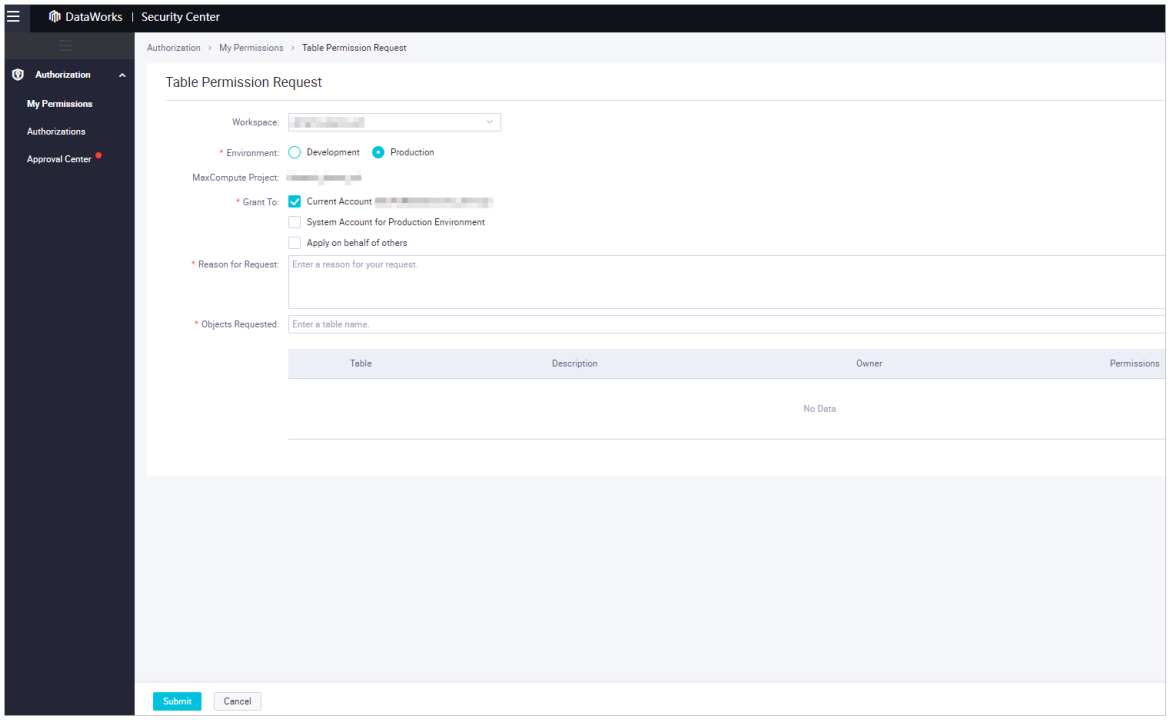
Request permissions on tables in Security Center

1. On the table details page, click **Apply for Permission**.



> **Note** If the table is hidden, the **Apply for Permission** button does not appear on the details page of the table.

2. On the **Table Permission Request** page, configure the parameters.



Parameter	Description
Workspace	The workspace to which the table belongs.
Environment	The environment to which the table belongs. For a workspace in standard mode, you can request permissions on the table in both the development environment and production environment. For a workspace in simple mode, you can request permissions on the table only in the production environment.
MaxCompute Project	The name of the MaxCompute project that is associated with the DataWorks workspace you selected. The value is automatically generated and cannot be changed.
Grant To	The account for which you request permissions on the table. Valid values: **Current Account** and **System Account for Production Environment**.
Valid Until	The validity period of the permissions on the table. Valid values: **1 Month**, **3 Months**, **6 Months**, **1 Year**, **Permanent**, and **Others**.
Reason for Request	The reason why you want to request permissions on the table. Enter a reason for faster approval.
Objects Requested	The name of the table.

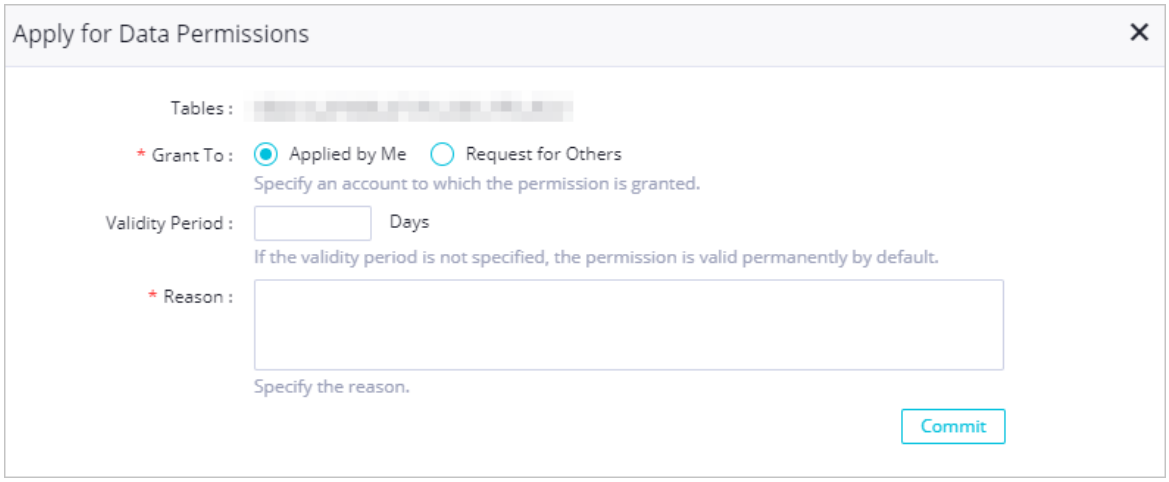
3. Click **Submit**.

Request permissions on tables in Data Map

1. On the table details page, click **Apply for Permission**.

> **Note** If the table is hidden, the **Apply for Permission** button does not appear on the details page of the table.

2. In the **Apply for Permission** dialog box, configure the parameters.



Parameter	Description
Table	The name of the table on which you want to request permissions. The value is automatically generated and cannot be changed.
Grant To	Specifies whether you want to request permissions on the table for the current account or another account. Valid values: **Current Account** and **Specified Account**.
Username	The username of the account for which you request permissions on the table. **Note** This parameter is available only when you set the **Grant To** parameter to **Specified Account**.
Validity Period	The validity period of the permissions on the table. If you do not set this parameter, the permissions are permanently valid.
Reason	The reason why you want to request permissions on the table. Enter a reason for faster approval.

3. Click **Submit**.

View the request status

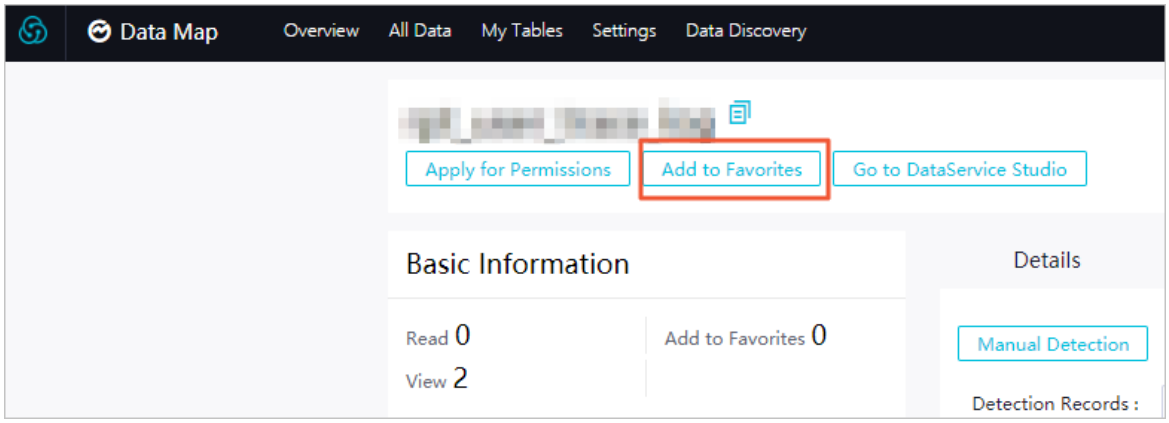
1. In the top navigation bar, click **My Data**.
2. On the My Data page, click **Permission Management** in the left-side navigation pane.
3. On the page that appears, click the **Submitted by Me** tab.
4. Find the required request record and click **View** in the Actions column to view the request status.

12.6.3. Add a table to favorites

This topic describes how to add a table to favorites, remove a table from favorites, and view the tables added to favorites.

Procedure

1. Go to the details page of a table.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > DataMap(Data Management)**.
   iii. In the top navigation bar, click **All Data**.
   iv. On the **All Data** page, click a tab based on your business requirements, such as MaxCompute.
   v. On the tab that appears, click the name of the table that you want to add to favorites.
2. On the table details page, click **Add to Favorites**.



3. In the top navigation bar, click **My Data**.
4. On the **My Data** page, click **My Favorites** in the left-side navigation pane.

   On the page that appears, you can view all the tables that you have added to favorites and remove tables from favorites. To remove a table from favorites, find the table that you want to remove and click **Remove from Favorites** in the Actions column.

12.6.4. Go to DataService Studio to create an API

This topic describes how to go to DataService Studio from the details page of a table to create an API.

Procedure

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > DataMap(Data Management)**.
3. In the top navigation bar, click **All Data**.
4. On the **All Data** page, click a tab based on your business requirements, such as MaxCompute.
5. On the details page of the table, click **Create API in DataService Studio**.
6. On the **DataService Studio** page, create an API based on tables or register the existing API as required. For more information, see Overview.

12.7. Data discovery

12.7.1. Collect metadata from an EMR data source

This topic describes how to create a crawler to collect metadata from an E-MapReduce (EMR) data source to DataWorks. You can view the collected metadata in Data Map.

Prerequisites

An EMR cluster is associated with your DataWorks workspace.

Procedure

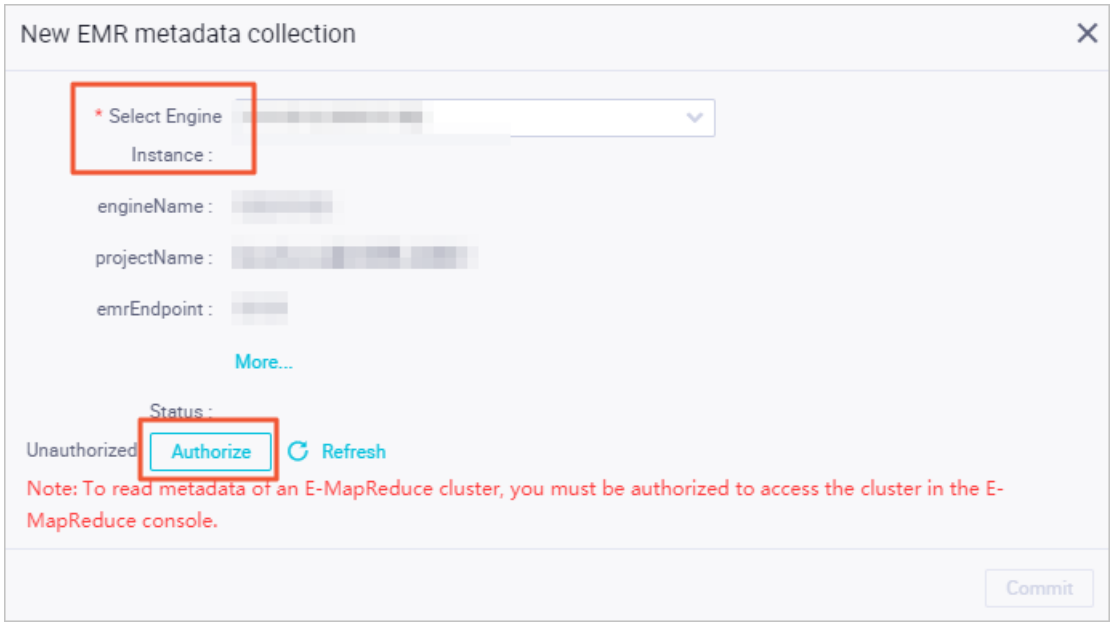
1. Go to the **Data Discovery** page.

i. Log on to the DataWorks console.

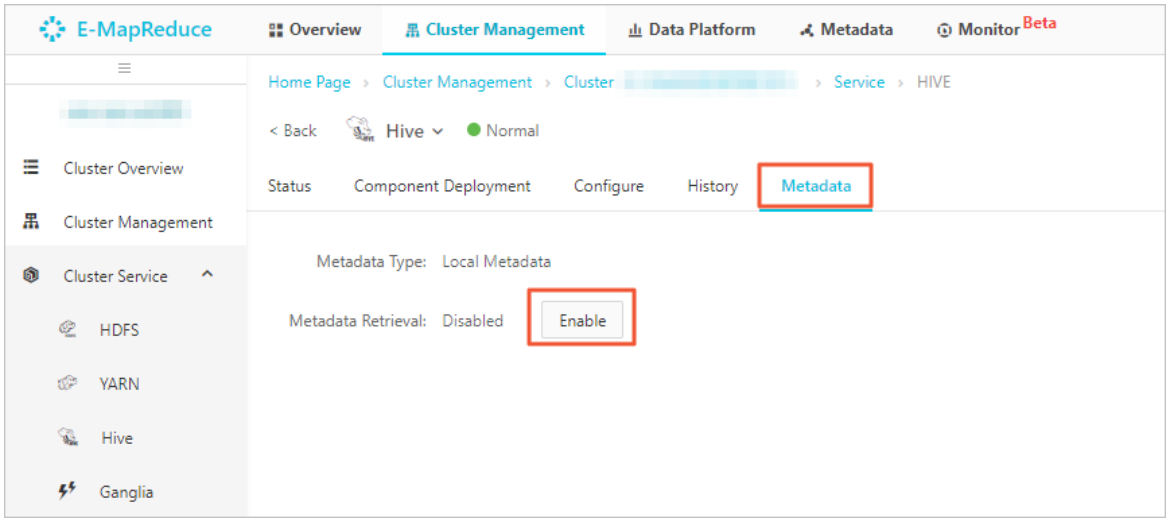
ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > DataMap(Data Management)**.

iii. On the Data Map page, click **Data Discovery** in the top navigation bar.

2. On the **E-MapReduce Metadata Crawler** page, click **Create Crawler**.
3. In the **Create Crawler** dialog box, select the associated EMR cluster from the **Select a cluster** drop-down list and click **Authorize**.



4. On the page that appears, click the **Metadata** tab and click **Enable**.



5. In the **Confirm Operation** message, click **OK**.
6. Return to the **Create Crawler** dialog box on the **E-MapReduce Metadata Crawler** page and click **Refresh**.
7. After the authorization status changes to **Authorized**, click **Commit**.
8. On the **E-MapReduce Metadata Crawler** page, find the newly created crawler and click **Obtain All** in the Actions column.

Click **Refresh** in the upper-right corner of the page and verify that the value in the **Running Status** column of the created crawler changes to **Collected**.

> **Note** After full metadata from the EMR data source is collected, the system automatically synchronizes new metadata from the data source.

If you want to delete the created crawler, click **Delete** in the Actions column. In the **Delete Instance** message, click **OK**.

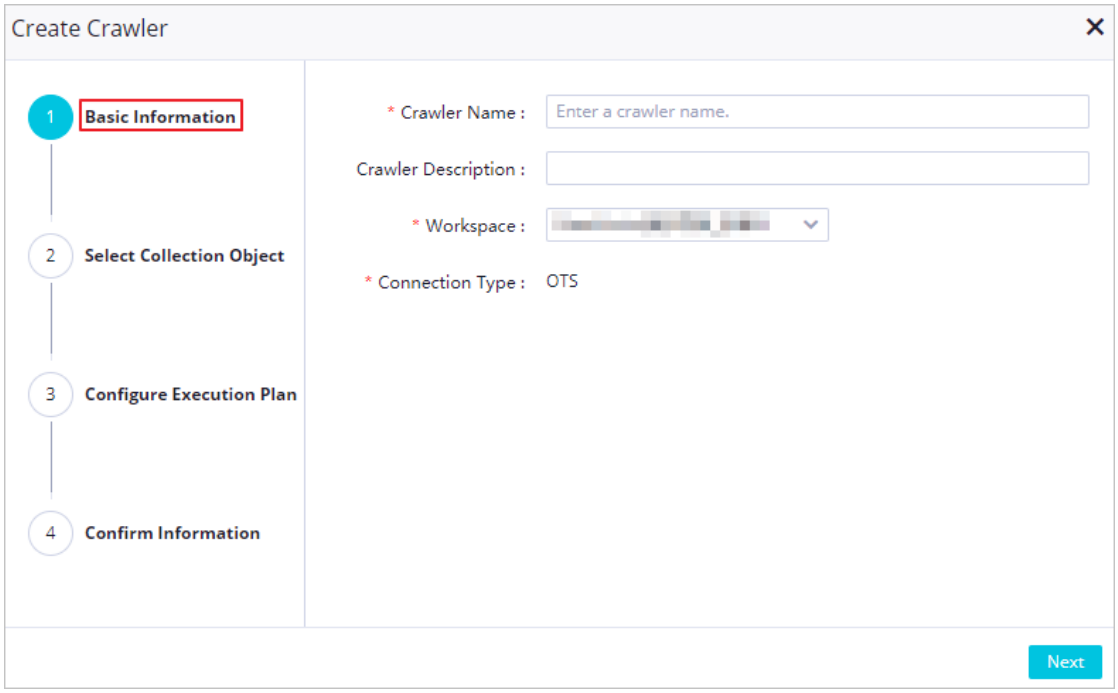
9. View the metadata collected from the EMR data source.
   i. In the top navigation bar, click **All Data**.
   ii. Click the **E-MapReduce** tab.
   iii. On the **E-MapReduce** tab, click the name of the table that stores the collected metadata and view the table details.

12.7.2. Collect metadata from a Tablestore data source

You can collect information about the schema and lineage of a table to Data Map. This way, the inner structure and association relationships of the table can be clearly displayed. This topic describes how to create a crawler and collect metadata from a Tablestore data source to DataWorks. You can view the collected metadata in Data Map.

1. Go to the **Data Discovery** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > DataMap(Data Management)**.
   iii. On the Data Map page, click **Data Discovery** in the top navigation bar.
2. In the left-side navigation pane, click **OTS**.
3. On the **OTSMetadata Crawler** page, click **Create Crawler**.
4. In the **Create Crawler** dialog box, perform the following steps:

i. In the **Basic Information** step, configure the parameters.



Parameter	Description
Crawler Name	Required. The name of the crawler. You must specify a unique name.
Crawler Description	The description of the crawler.
Workspace	The workspace of the data source from which you want to collect metadata.
Data Source Type	The type of data source from which you want to collect metadata. The default value is **OTS** and cannot be changed.

ii. Click **Next**.

iii. In the **Select Collection Object** step, select a data source from the **Data Source** drop-down list.

If no data source is available, click **Create** to go to the **Data Source** page and add a Tablestore data source. For more information, see Configure a Tablestore connection.

iv. Click Start Testing next to **Test Crawler Connectivity**.

v. If the message **The connectivity test is successful** appears, click **Next**.

If the message **The connectivity test failed** appears, check whether you have configured a valid data source.

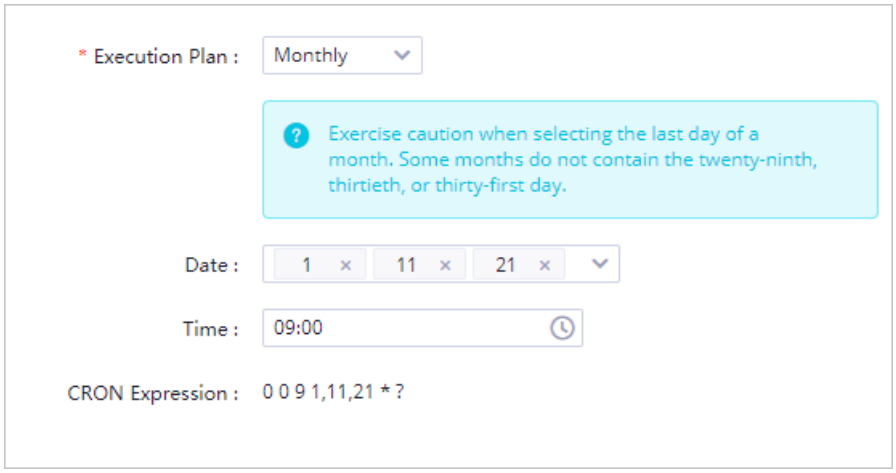
vi. In the **Configure Execution Plan** step, specify **Execution Plan**.

Valid values of the **Execution Plan** parameter: **On-demand Execution**, **Monthly**, **Weekly**, **Daily**, and **Hourly**. The running plan that is generated varies based on the running cycle. The system collects metadata from the Tablestore data source based on the running cycle that you specify. The following descriptions explain each value and provide examples:

- On-demand Execution: The system collects metadata from the Tablestore data source based on your business requirements.
- Monthly: The system automatically collects metadata from the Tablestore data source once at a specific time on several specific days of each month.

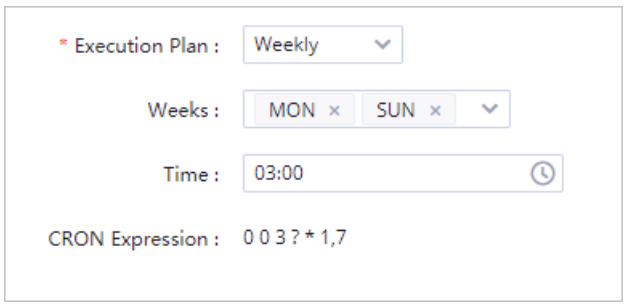
> **Notice** Some months do not have the 29th, 30th, or 31st day. In this case, the system does not collect metadata from the Tablestore data source on these dates. We recommend that you do not select the last day of a month.

The following figure shows that the system automatically collects metadata from the Tablestore data source once at 09:00 on the 1st, 11th, and 21th days of each month. **Cron Expression** is automatically generated based on the values of Date and Time.



- Weekly: The system automatically collects metadata from the Tablestore data source once at a specific time on several specific days of each week.

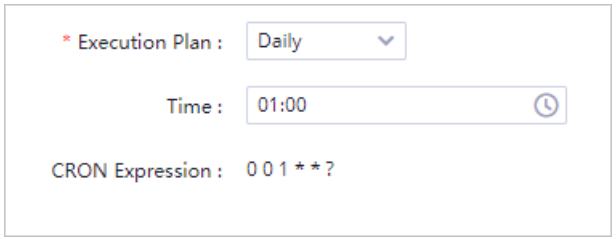
  The following figure shows that the system automatically collects metadata from the Tablestore data source once at 03:00 on Sunday and Monday of each week.



  If the **Time** parameter is not specified, the system automatically collects Tablestore metadata once at 00:00:00 on the specific days of each week.

- Daily: The system automatically collects metadata from the Tablestore data source once at a specific time of each day.

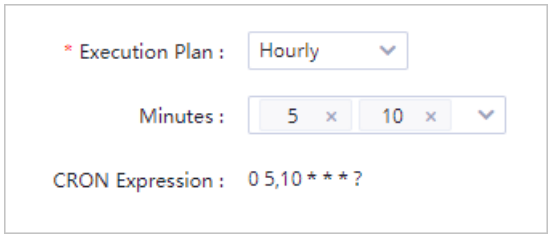
  The following figure shows that the system automatically collects metadata from the Tablestore data source once at 01:00 each day.

- Hourly: The system automatically collects metadata from the Tablestore data source once from the `N × 5` th minute of each hour.

  > **Note** For a Tablestore metadata collection task that runs each hour, you can configure the time to a multiple of 5 minutes.

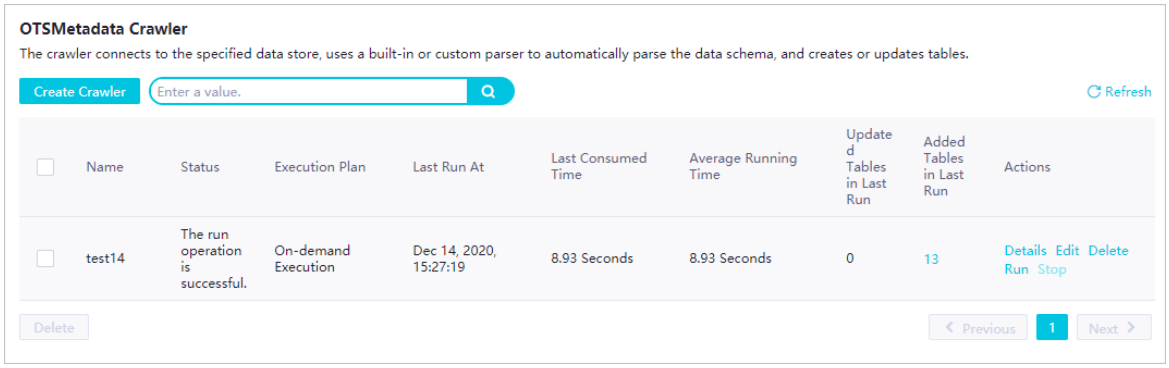
  The following figure shows that the system automatically collects metadata from the Tablestore data source from the 5th and 10th minutes of each hour.

vii. Click **Next**.

viii. In the **Confirm Information** step, check the information that you specified and click **Confirm**.

5. On the **OTSMetadata Crawler** page, you can view the information about your crawler and manage your crawler.



The following descriptions show the information that you can view and the operations that you can perform:

- You can view **Status**, **Execution Plan**, **Last Run At**, **Last Consumed Time**, **Average Running Time**, Updated Tables in Last Run, and Added Tables in Last Run of your crawler.
- You can click **Details**, **Edit**, **Delete**, **Run**, or **Stop** in the **Actions** column to perform the desired operation.

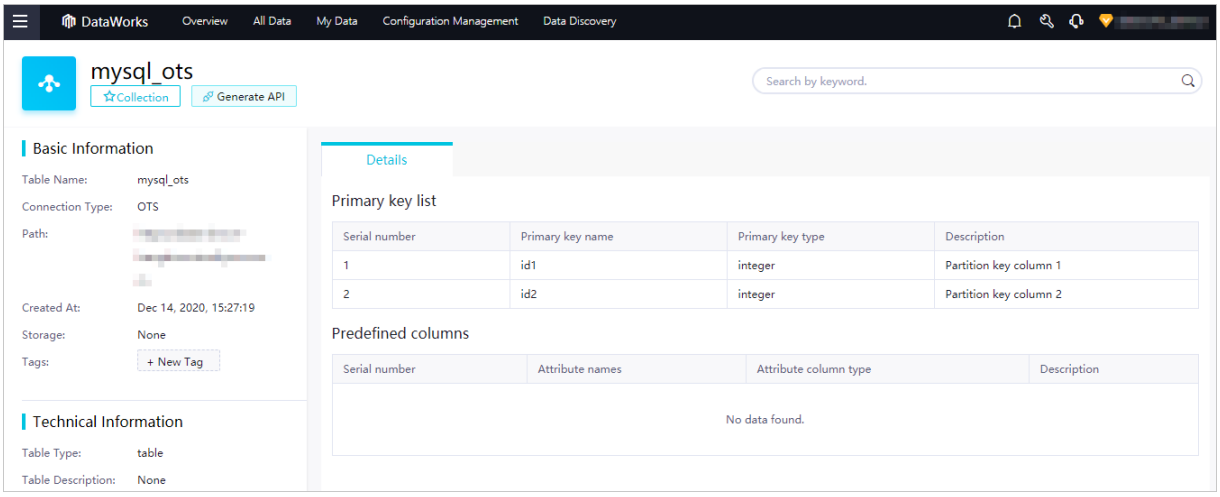
- **Details**: View **Crawler Name**, **Data Source Type**, and **Execution Plan** configured for the crawler.
- **Edit**: Modify the configurations of the crawler.
- **Delete**: Delete the crawler.
- **Run**: Run the task to collect metadata from the Tablestore data source. The **Run** entry point is available only when **Execution Plan** is set to **On-demand Execution**.
- **Stop**: Stop running the crawler.

Result

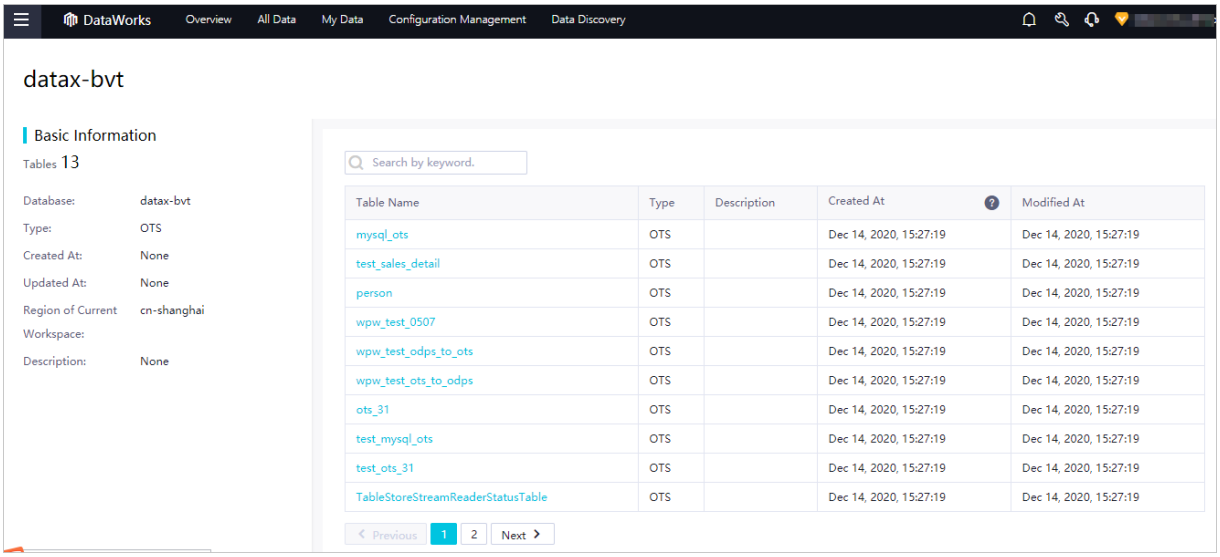
After the metadata in the Tablestore data source is collected, switch back to the previous page and click **All Data** in the top navigation bar. On the page that appears, click the **OTS** tab in the upper part. On the OTS tab, you can view the table that stores the collected Tablestore metadata.

Click the **table name**, **workspace**, or **database** to view the related details.

Example 1: View the details of the *mysql_ots* table.



Example 2: View all tables in the *datax-bvt* database.



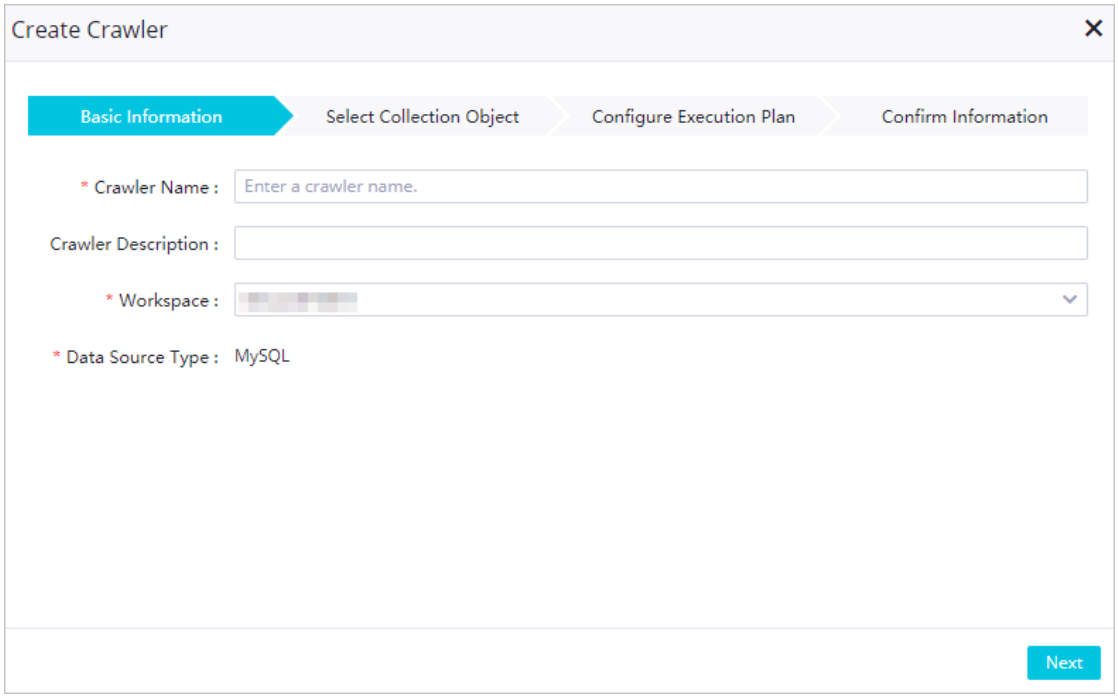
12.7.3. Collect metadata from a MySQL data source

This topic describes how to create a crawler to collect metadata from a MySQL data source to DataWorks. You can view the collected metadata on the Data Map page.

Procedure

1. Go to the **Data Discovery** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > DataMap(Data Management)**.
   iii. In the top navigation bar, click **Data Discovery**.
2. In the left-side navigation pane, click **MySQL**.
3. On the **MySQLMetadata Crawler** page, click **Create Crawler**.
4. In the **Create Crawler** dialog box, set the parameters in each step.

i. In the **Basic Information** step, set the parameters.



Parameter	Description
Crawler Name	Required. The name of the crawler. You must set a unique name.
Crawler Description	The description of the crawler.
Workspace	The workspace of the data source from which you want to collect metadata.
Data Source Type	The type of the data source from which you want to collect metadata. The default value is **MySQL** and cannot be changed.

ii. Click **Next**.

iii. In the **Select Collection Object** step, select a data source from the **Data Source** drop-down list.

If no data sources are available, click **Create** to go to the **Data Source** page and add a MySQL data source. For more information, see Configure a MySQL connection.

> **Note** You can select an ApsaraDB RDS for MySQL instance or a MySQL data source that is accessible from the Internet by using a Java Database Connectivity (JDBC) connection string.

iv. Click Start Testing next to **Test Crawler Connectivity**.

v. If the message **The connectivity test is successful** appears, click **Next**.

vi. In the **Configure Execution Plan** step, configure an execution plan.

Valid values of the **Execution Plan** parameter are **On-demand Execution**, **Monthly**, **Weekly**, **Daily**, and **Hourly**.

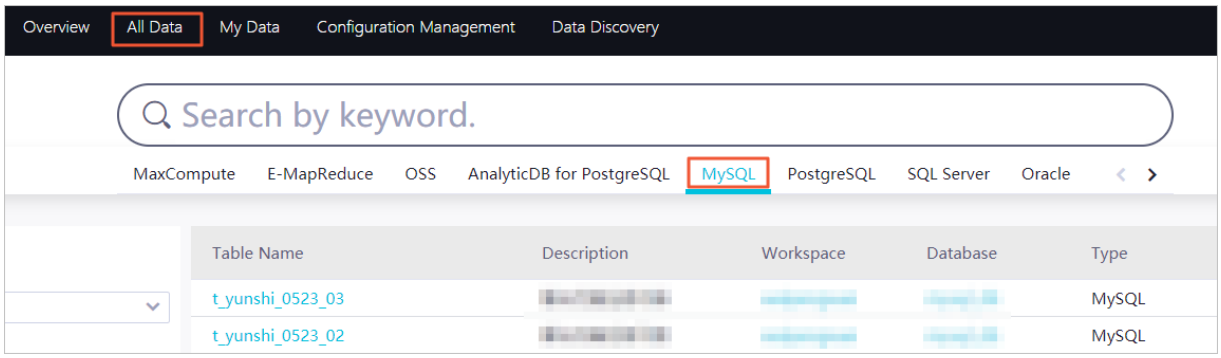
vii. Click **Next**.

viii. In the **Confirm Information** step, check the information that you specified and click **Confirm**.

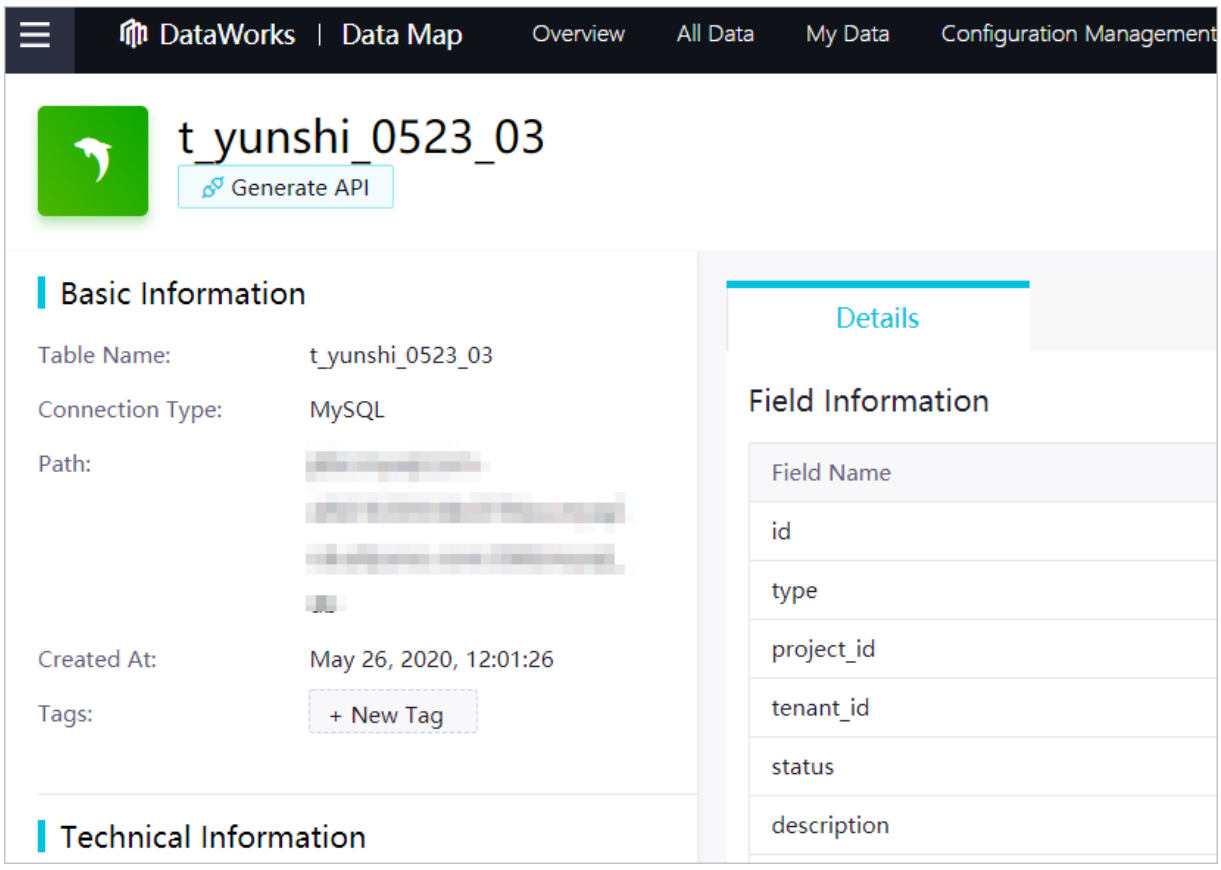
5. On the **MySQLMetadata Crawler** page, find the created crawler and click **Run** in the Actions column.

Result

After the metadata in the MySQL data source is collected, click **All Data** in the top navigation bar. Select **MySQL** from the options in the upper part of the page. You can view the tables that store the collected MySQL metadata.



Click the name of the table to view the table details.



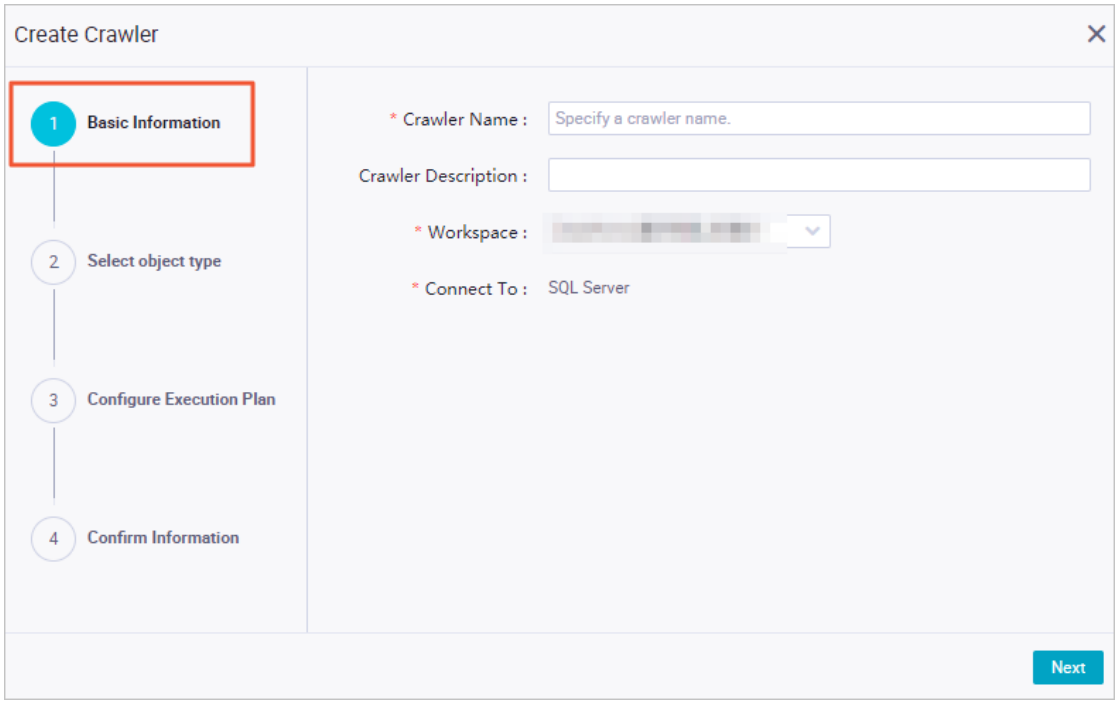
12.7.4. Collect metadata from an SQL Server data source

This topic describes how to create a crawler to collect metadata from an SQL Server data source. You can view the collected metadata on the Data Map page.

Procedure

1. Go to the **Data Discovery** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > DataMap(Data Management)**.
   iii. In the top navigation bar, click **Data Discovery**.
2. In the left-side navigation pane, click **SQL Server**.
3. On the **SQL ServerMetadata Crawler** page, click **Create Crawler**.
4. In the **Create Crawler** dialog box, set the parameters in each step.

i. In the **Basic Information** step, set the parameters.



Parameter	Description
Crawler Name	Required. The name of the crawler. You must set a unique name.
Crawler Description	The description of the crawler.
Workspace	The workspace of the data source from which you want to collect metadata.
Data Source Type	The type of the data source from which you want to collect metadata. The default value is **SQL Server**.

ii. Click **Next**.

iii. In the **Select Collection Object** step, select a data source from the **Data Source** drop-down list.

If no data sources are available, click **Create** to go to the **Data Source** page and add an SQL Server data source. For more information, see Configure an SQL Server data source.

iv. Click Start Testing next to **Test Crawler Connectivity**.

v. If the message **The connectivity test is successful** appears, click **Next**.

If the message **The connectivity test of the data source failed, and the data source cannot be connected to the resource group** appears, check whether you have configured a valid data source.

vi. In the **Configure Execution Plan** step, configure an execution plan.

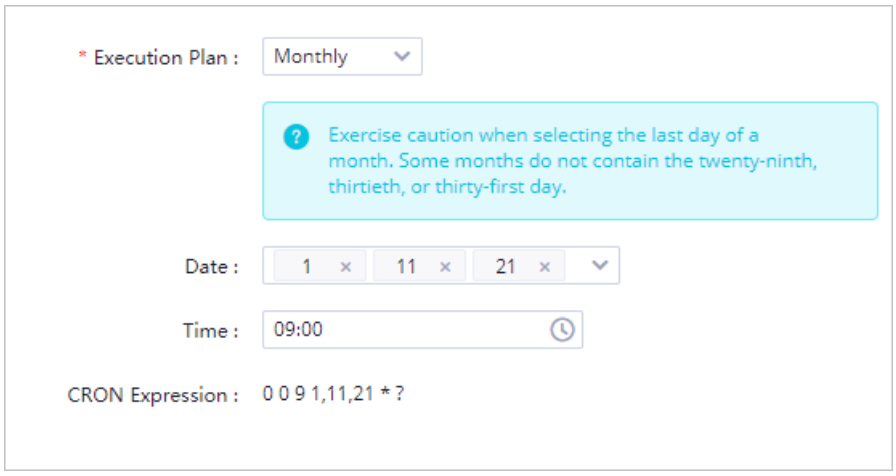
Valid values of the **Execution Plan** parameter are **On-demand Execution**, **Monthly**, **Weekly**, **Daily**, and **Hourly**. The execution plan that is generated varies based on the execution cycle. The system collects metadata from the SQL Server data source based on the

execution cycle that you specify. The following descriptions explain each value and provide examples:

- On-demand Execution: The system collects metadata from the SQL Server data source based on your business requirements.
- Monthly: The system automatically collects metadata from the SQL Server data source once at a specific time on several specific days of each month.

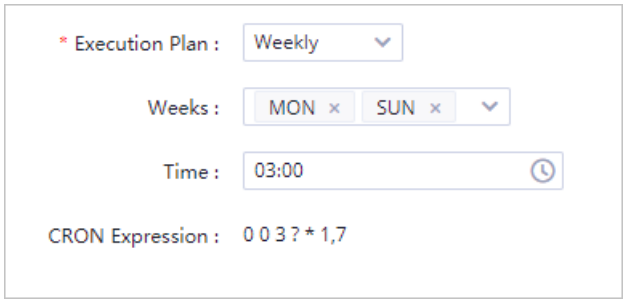
  > **Notice** Some months do not have the 29th, 30th, or 31st day. In these months, the system does not collect metadata from the SQL Server data source on these dates. We recommend that you do not select the last days of a month.

  The following figure shows that the system automatically collects metadata from the SQL Server data source once at 09:00 on the 1st, 11th, and 21st days of each month. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.



- Weekly: The system automatically collects metadata from the SQL Server data source once at a specific time on several specific days of each week.

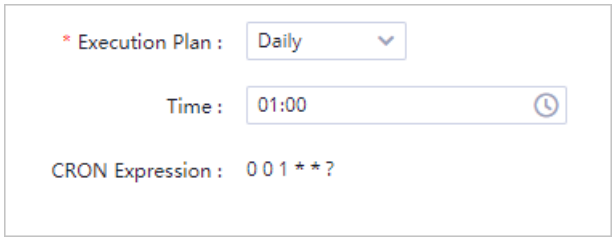
  The following figure shows that the system automatically collects metadata from the SQL Server data source once at 03:00 on Sunday and Monday of each week.



  If the **Time** parameter is not set, the system automatically collects metadata from the SQL Server data source once at 00:00:00 on the specific days of each week.

- Daily: The system automatically collects metadata from the SQL Server data source once at a specific time of each day.

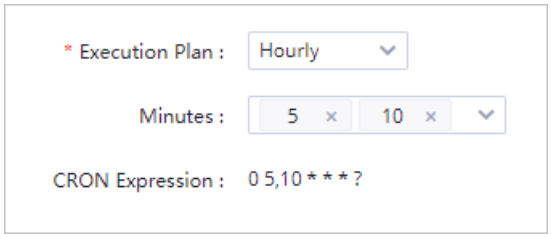
  The following figure shows that the system automatically collects metadata from the SQL Server data source once at 01:00 each day.

- Hourly: The system automatically collects metadata from the SQL Server data source once from the `N × 5` th minute of each hour.

  > **Note** For an SQL Server metadata collection task that is run each hour, you can set the time to a multiple of 5 minutes.

  The following figure shows that the system automatically collects metadata from the SQL Server data source from the 5th and 10th minutes of each hour.

vii. Click **Next**.

viii. In the **Confirm Information** step, check the information that you specified and click **Confirm**.

5. On the **SQL ServerMetadata Crawler** page, find the created crawler and click **Run** in the Actions column.

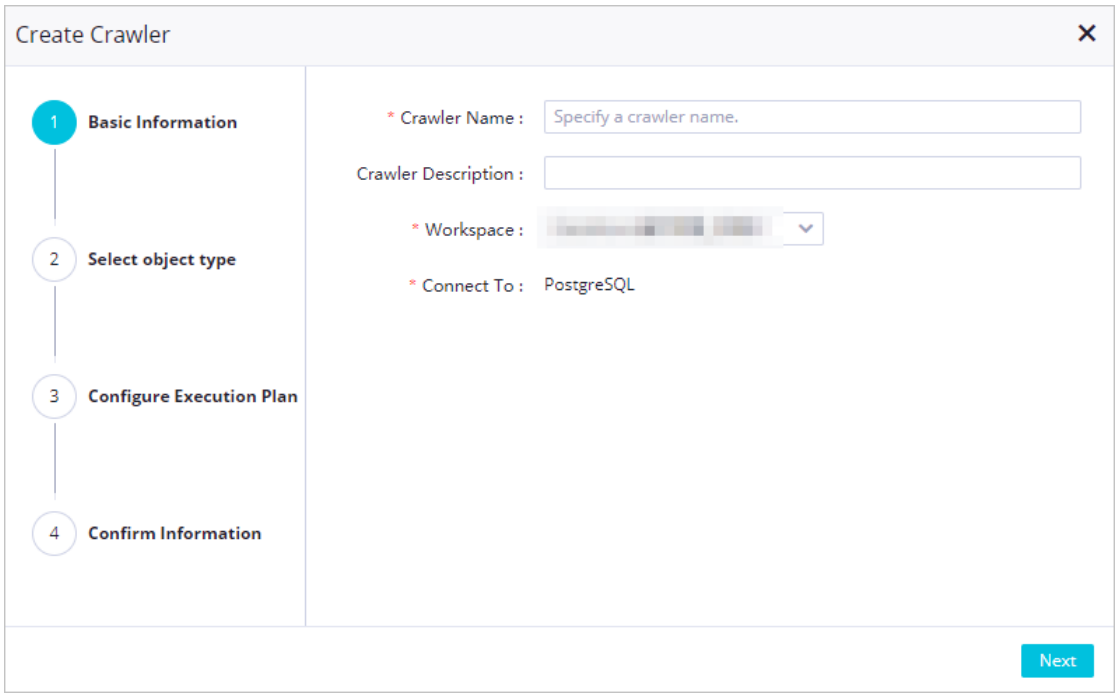
12.7.5. Collect metadata from a PostgreSQL data source

This topic describes how to create a crawler to collect metadata from a PostgreSQL data source to DataWorks. You can view the collected metadata on the Data Map page.

Procedure

1. Go to the **Data Discovery** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > DataMap(Data Management)**.
   iii. In the top navigation bar, click **Data Discovery**.
2. In the left-side navigation pane, click **PostgreSQL**.

3. On the **PostgreSQLMetadata Crawler** page, click **Create Crawler**.
4. In the **Create Crawler** dialog box, set the parameters in each step.
   i. In the **Basic Information** step, set the parameters.



Parameter	Description
Crawler Name	Required. The name of the crawler. You must set a unique name.
Crawler Description	The description of the crawler.
Workspace	The workspace of the data source from which you want to collect metadata.
Data Source Type	The type of the data source from which you want to collect metadata. The default value is **PostgreSQL** and cannot be changed.

   ii. Click **Next**.
   iii. In the **Select Collection Object** step, select a data source from the **Data Source** drop-down list.

   If no data sources are available, click **Create** to go to the **Data Source** page and add a PostgreSQL data source. For more information, see Configure a PostgreSQL connection.
   iv. Click Start Testing next to **Test Crawler Connectivity**.
   v. If the message **The connectivity test is successful** appears, click **Next**.

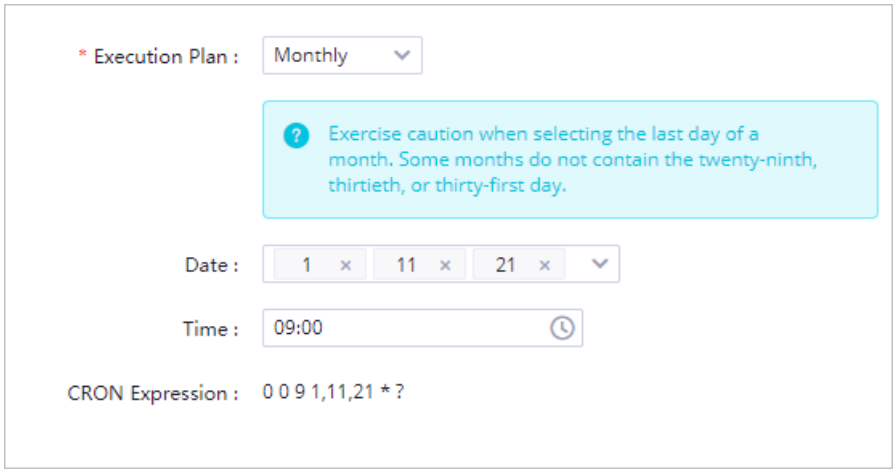
   If the message **The connectivity test of the data source failed, and the data source cannot be connected to the resource group** appears, check whether you have configured a valid data source.
   vi. In the **Configure Execution Plan** step, configure an execution plan.

Valid values of the **Execution Plan** parameter are **On-demand Execution**, **Monthly**, **Weekly**, **Daily**, and **Hourly**. The execution plan that is generated varies based on the execution cycle. The system collects metadata from the PostgreSQL data source based on the execution cycle that you specify. The following descriptions explain each value and provide examples:

- On-demand Execution: The system collects metadata from the PostgreSQL data source based on your business requirements.
- Monthly: The system automatically collects metadata from the PostgreSQL data source once at a specific time on several specific days of each month.

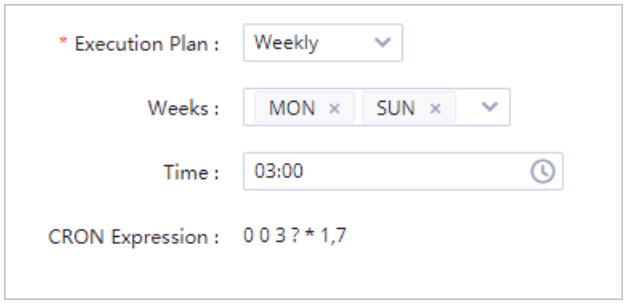
  > **Notice** Some months do not have the 29th, 30th, or 31st day. In these months, the system does not collect metadata from the PostgreSQL data source on these dates. We recommend that you do not select the last days of a month.

  The following figure shows that the system automatically collects metadata from the PostgreSQL data source once at 09:00 on the 1st, 11th, and 21st days of each month. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.



- Weekly: The system automatically collects metadata from the PostgreSQL data source once at a specific time on several specific days of each week.

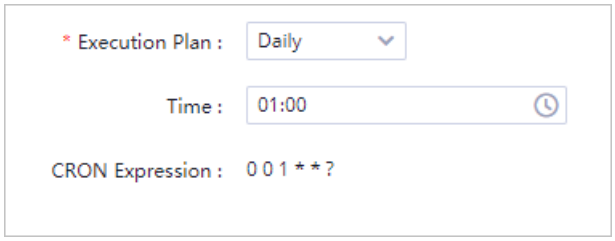
  The following figure shows that the system automatically collects metadata from the PostgreSQL data source once at 03:00 on Sunday and Monday of each week.



  If the **Time** parameter is not set, the system automatically collects metadata from the PostgreSQL data source once at 00:00:00 on the specific days of each week.

- Daily: The system automatically collects metadata from the PostgreSQL data source once at a specific time of each day.

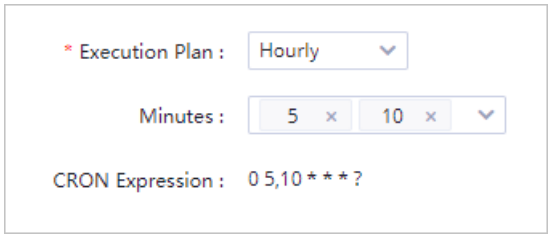
  The following figure shows that the system automatically collects metadata from the PostgreSQL data source once at 01:00 each day.

- Hourly: The system automatically collects metadata from the PostgreSQL data source once from the `N × 5` th minute of each hour.

  > **Note** For a PostgreSQL metadata collection task that is run each hour, you can set the time to a multiple of 5 minutes.

  The following figure shows that the system automatically collects metadata from the PostgreSQL data source from the 5th and 10th minutes of each hour.

vii. Click **Next**.

viii. In the **Confirm Information** step, check the information that you specified and click **Confirm**.

5. On the **PostgreSQLMetadata Crawler** page, find the created crawler and click **Run** in the Actions column.

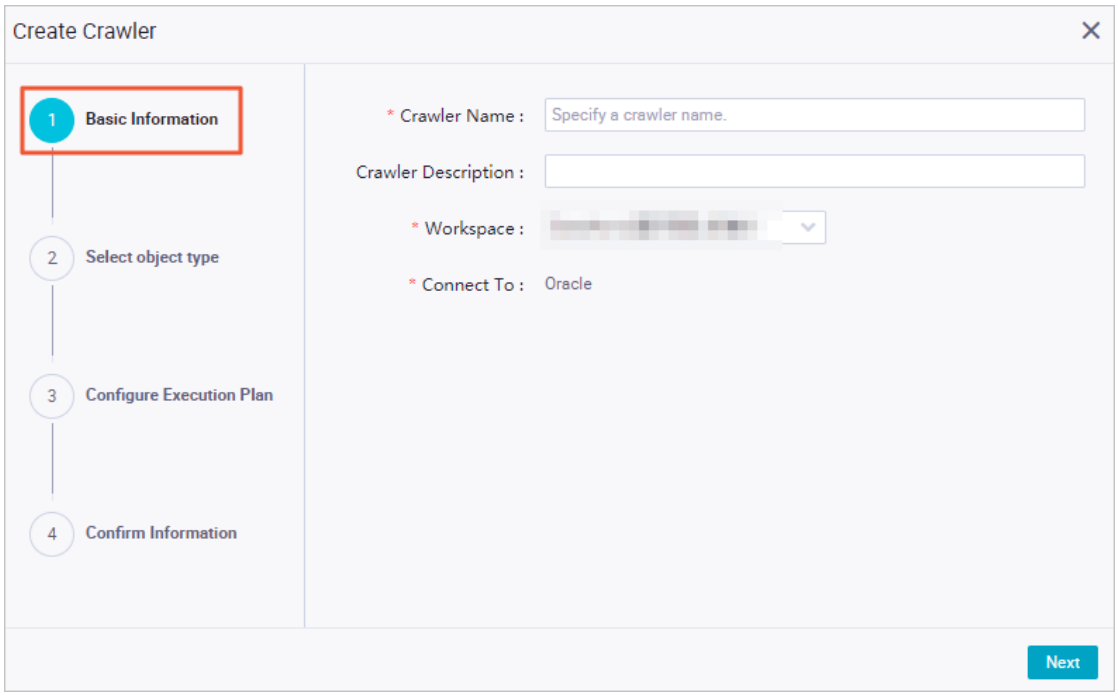
12.7.6. Collect metadata from an Oracle data source

This topic describes how to create a crawler to collect metadata from an Oracle data source to DataWorks. You can view the collected metadata on the Data Map page.

Procedure

1. Go to the **Data Discovery** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the ☰ icon in the upper-left corner and choose **All Products > Data governance > DataMap(Data Management)**.
   iii. In the top navigation bar, click **Data Discovery**.
2. In the left-side navigation pane, click **Oracle**.

3. On the **OracleMetadata Crawler** page, click **Create Crawler**.
4. In the **Create Crawler** dialog box, set the parameters in each step.
   i. In the **Basic Information** step, set the parameters.



Parameter	Description
Crawler Name	Required. The name of the crawler. You must set a unique name.
Crawler Description	The description of the crawler.
Workspace	The workspace of the data source from which you want to collect metadata.
Data Source Type	The type of the data source from which you want to collect metadata. The default value is **Oracle** and cannot be changed.

   ii. Click **Next**.
   iii. In the **Select Collection Object** step, select a data source from the **Data Source** drop-down list.

   If no data sources are available, click **Create** to go to the **Data Source** page and add an Oracle data source. For more information, see Configure an Oracle connection.

   iv. Click Start Testing next to **Test Crawler Connectivity**.
   v. If the message **The connectivity test is successful** appears, click **Next**.

   If the message **The connectivity test of the data source failed, and the data source cannot be connected to the resource group** appears, check whether you have configured a valid data source.

   vi. In the **Configure Execution Plan** step, configure an execution plan.

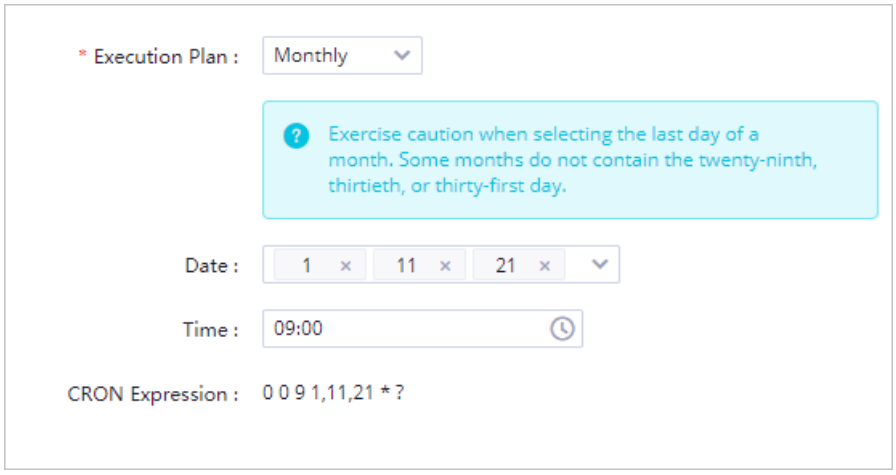
   Valid values of the **Execution Plan** parameter are **On-demand Execution**, **Monthly**,

Weekly, **Daily**, and **Hourly**. The execution plan that is generated varies based on the execution cycle. The system collects metadata from the Oracle data source based on the execution cycle that you specify. The following descriptions explain each value and provide examples:

- On-demand Execution: The system collects metadata from the Oracle data source based on your business requirements.
- Monthly: The system automatically collects metadata from the Oracle data source once at a specific time on several specific days of each month.

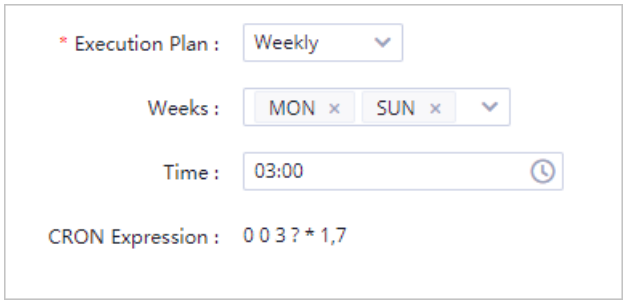
> **Notice** Some months do not have the 29th, 30th, or 31st day. In these months, the system does not collect metadata from the Oracle data source on these dates. We recommend that you do not select the last days of a month.

The following figure shows that the system automatically collects metadata from the Oracle data source once at 09:00 on the 1st, 11th, and 21st days of each month. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.



- Weekly: The system automatically collects metadata from the Oracle data source once at a specific time on several specific days of each week.

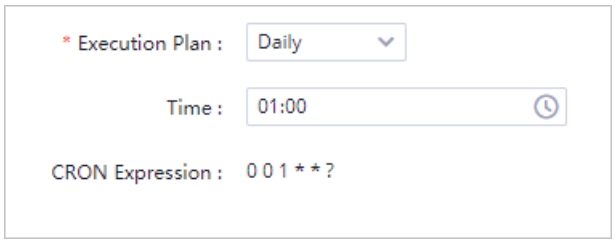
  The following figure shows that the system automatically collects metadata from the Oracle data source once at 03:00 on Sunday and Monday of each week.



  If the **Time** parameter is not specified, the system automatically collects Oracle metadata once at 00:00:00 on the specific days of each week.

- Daily: The system automatically collects metadata from the Oracle data source once at a specific time of each day.

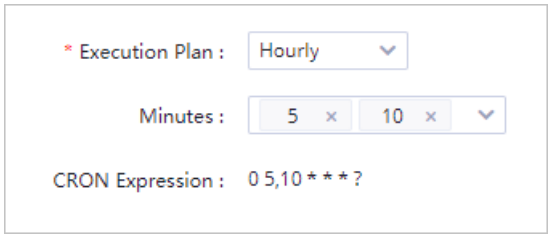
  The following figure shows that the system automatically collects metadata from the Oracle data source once at 01:00 each day.

- Hourly: The system automatically collects metadata from the Oracle data source once from the `N × 5` th minute of each hour.

  > **Note** For an Oracle metadata collection task that is run each hour, you can set the time to a multiple of 5 minutes.

  The following figure shows that the system automatically collects metadata from the Oracle data source from the 5th and 10th minutes of each hour.

vii. Click **Next**.

viii. In the **Confirm Information** step, check the information that you specified and click **Confirm**.

5. On the **OracleMetadata Crawler** page, find the created crawler and click **Run** in the Actions column.

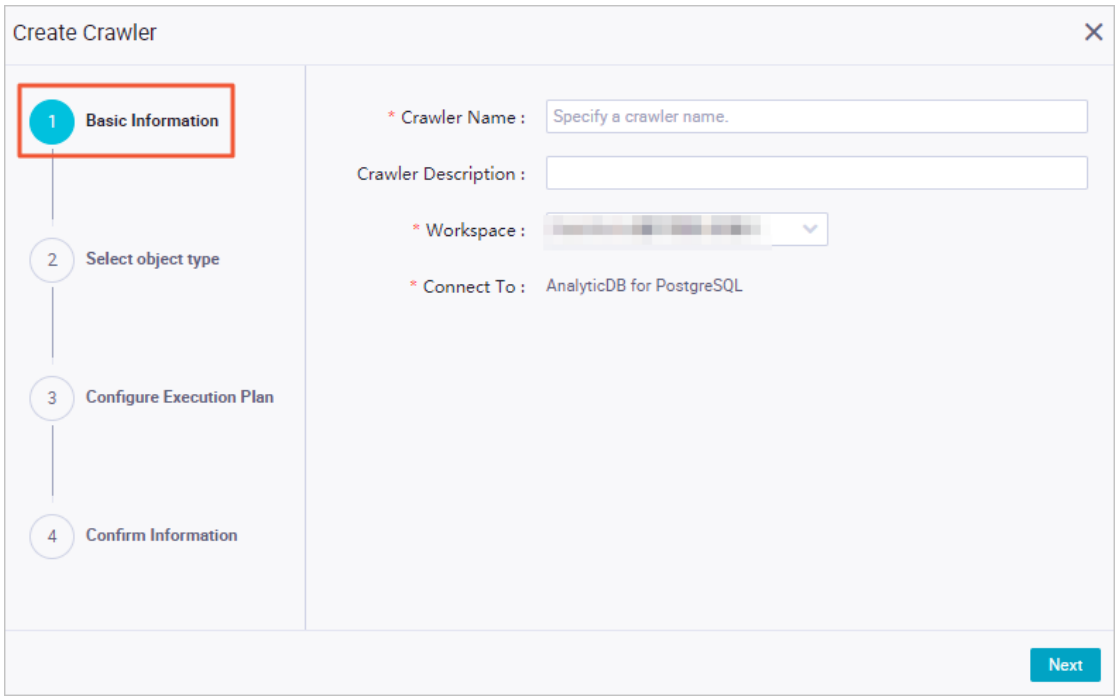
12.7.7. Collect metadata from an AnalyticDB for PostgreSQL data source

This topic describes how to create a crawler to collect metadata from an AnalyticDB for PostgreSQL data source to DataWorks. You can view the collected metadata in Data Map.

Procedure

1. Go to the **Data Discovery** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > DataMap(Data Management)**.
   iii. In the top navigation bar, click **Data Discovery**.
2. In the left-side navigation pane, click **AnalyticDB for PostgreSQL**.

3. On the **AnalyticDB for PostgreSQLMetadata Crawler** page, click **Create Crawler**.
4. In the **Create Crawler** dialog box, set the parameters in each step.
   i. In the **Basic Information** step, set the parameters.



Parameter	Description
Crawler Name	Required. The name of the crawler. You must set a unique name.
Crawler Description	The description of the crawler.
Workspace	The workspace of the data source from which you want to collect metadata.
Data Source Type	The type of the data source from which you want to collect metadata. The default value is **AnalyticDB for PostgreSQL** and cannot be changed.

   ii. Click **Next**.
   iii. In the **Select Collection Object** step, select a data source from the **Data Source** drop-down list.

   If no data sources are available, click **Create** to go to the **Data Source** page and add an AnalyticDB for PostgreSQL data source.
   iv. Click Start Testing next to **Test Crawler Connectivity**.
   v. If the message **The connectivity test is successful** appears, click **Next**.

   If the message **The connectivity test of the data source failed, and the data source cannot be connected to the resource group** appears, check whether you have configured a valid data source.
5. On the **AnalyticDB for PostgreSQLMetadata Crawler** page, find the created crawler and click

Run in the Actions column.

After the crawler is run, click the number in the **Tables found during Last Run** column to view the details of the updated or created tables.

> **Notice** The **Run** button is available only in the Actions column of a crawler that needs to be manually triggered.

You can also perform the following operations on the AnalyticDB for PostgreSQLMetadata Crawler page:

- Click **Details** in the Actions column that corresponds to a crawler. In the **Crawler Details** dialog box, view the detailed information about the crawler.
- Click **Edit** in the Actions column that corresponds to a crawler. In the **Edit Crawler** dialog box, modify the configurations of the crawler.
- Click **Delete** in the Actions column that corresponds to a crawler. In the **Confirm** message, click **Ok** to delete the crawler.
- Find a crawler that is running and click **Stop** in the **Actions** column to stop the crawler.

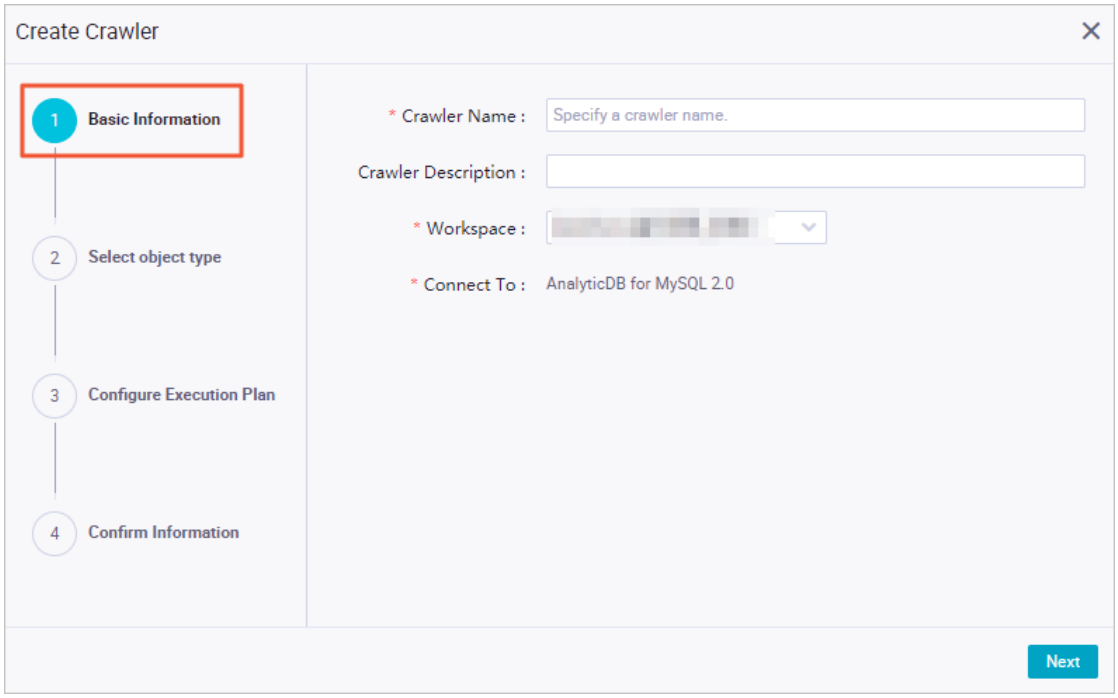
12.7.8. Collect metadata from an AnalyticDB for MySQL 2.0 data source

This topic describes how to create a crawler to collect metadata from an AnalyticDB for MySQL 2.0 data source to DataWorks. You can view the collected metadata on the Data Map page.

Procedure

1. Go to the **Data Discovery** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > DataMap(Data Management)**.
   iii. In the top navigation bar, click **Data Discovery**.
2. In the left-side navigation pane, click **AnalyticDB for MySQL 2.0**.
3. On the **AnalyticDB for MySQL 2.0Metadata Crawler** page, click **Create Crawler**.
4. In the **Create Crawler** dialog box, set the parameters in each step.

i. In the **Basic Information** step, set the parameters.



Parameter	Description
Crawler Name	Required. The name of the crawler. You must set a unique name.
Crawler Description	The description of the crawler.
Workspace	The workspace of the data source from which you want to collect metadata.
Data Source Type	The type of the data source from which you want to collect metadata. The default value is **AnalyticDB for MySQL 2.0** and cannot be changed.

ii. Click **Next**.

iii. In the **Select Collection Object** step, select a data source from the **Data Source** drop-down list.

If no data sources are available, click **Create** to go to the **Data Source** page and add an AnalyticDB for MySQL V2.0 data source.

iv. Click Start Testing next to **Test Crawler Connectivity**.

v. If the message **The connectivity test is successful** appears, click **Next**.

If the message **The connectivity test of the data source failed, and the data source cannot be connected to the resource group** appears, check whether you have configured a valid data source.

vi. In the **Configure Execution Plan** step, configure an execution plan.

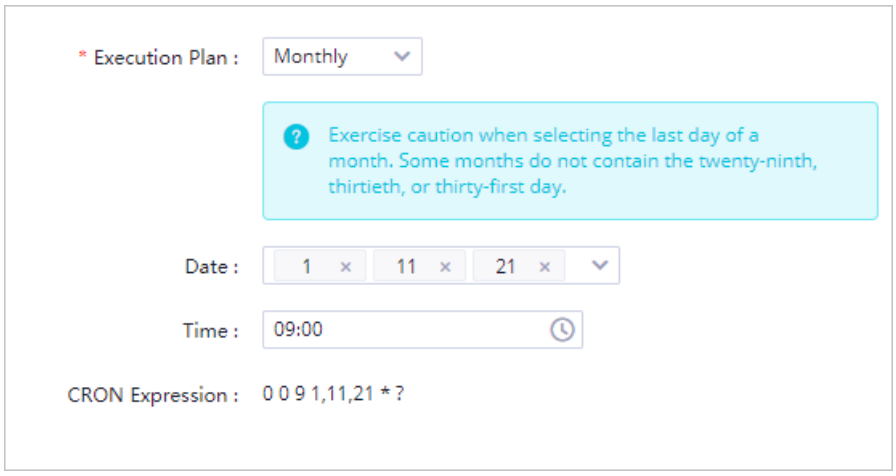
Valid values of the **Execution Plan** parameter are **On-demand Execution**, **Monthly**, **Weekly**, **Daily**, and **Hourly**. The execution plan that is generated varies based on the execution cycle. The system collects metadata from the AnalyticDB for MySQL V2.0 data

source based on the execution cycle that you specify. The following descriptions explain each value and provide examples:

- On-demand Execution: The system collects metadata from the AnalyticDB for MySQL V2.0 data source based on your business requirements.
- Monthly: The system automatically collects metadata from the AnalyticDB for MySQL V2.0 data source once at a specific time on several specific days of each month.

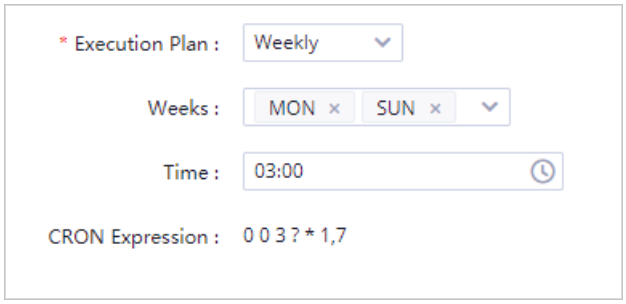
> **Notice** Some months do not have the 29th, 30th, or 31st day. In these months, the system does not collect metadata from the AnalyticDB for MySQL V2.0 data source on these dates. We recommend that you do not select the last days of a month.

The following figure shows that the system automatically collects metadata from the AnalyticDB for MySQL V2.0 data source once at 09:00 on the 1st, 11th, and 21st days of each month. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.



- Weekly: The system automatically collects metadata from the AnalyticDB for MySQL V2.0 data source once at a specific time on several specific days of each week.

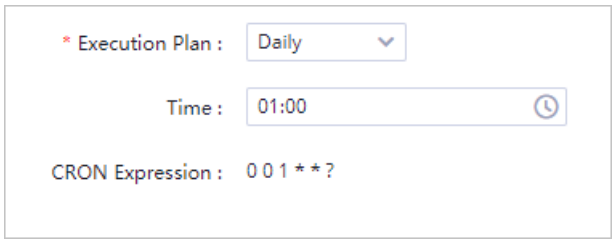
  The following figure shows that the system automatically collects metadata from the AnalyticDB for MySQL V2.0 data source once at 03:00 on Sunday and Monday of each week.



If the **Time** parameter is not specified, the system automatically collects AnalyticDB for MySQL V2.0 metadata once at 00:00:00 on the specific days of each week.

- Daily: The system automatically collects metadata from the AnalyticDB for MySQL V2.0 data source once at a specific time of each day.

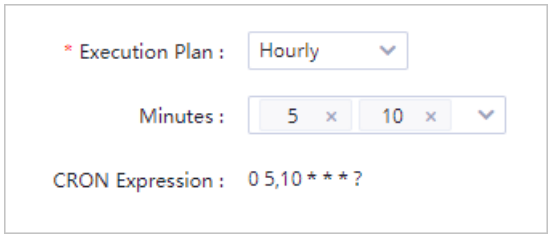
  The following figure shows that the system automatically collects metadata from the AnalyticDB for MySQL V2.0 data source once at 01:00 each day.

- Hourly: The system automatically collects metadata from the AnalyticDB for MySQL V2.0 data source once from the `N × 5` th minute of each hour.

  > **Note** For an AnalyticDB for MySQL V2.0 metadata collection task that is run each hour, you can set the time to a multiple of 5 minutes.

  The following figure shows that the system automatically collects metadata from the AnalyticDB for MySQL V2.0 data source from the 5th and 10th minutes of each hour.

vii. Click **Next**.

viii. In the **Confirm Information** step, check the information that you specified and click **Confirm**.

5. On the **AnalyticDB for MySQL 2.0Metadata Crawler** page, find the created crawler and click **Run** in the Actions column.

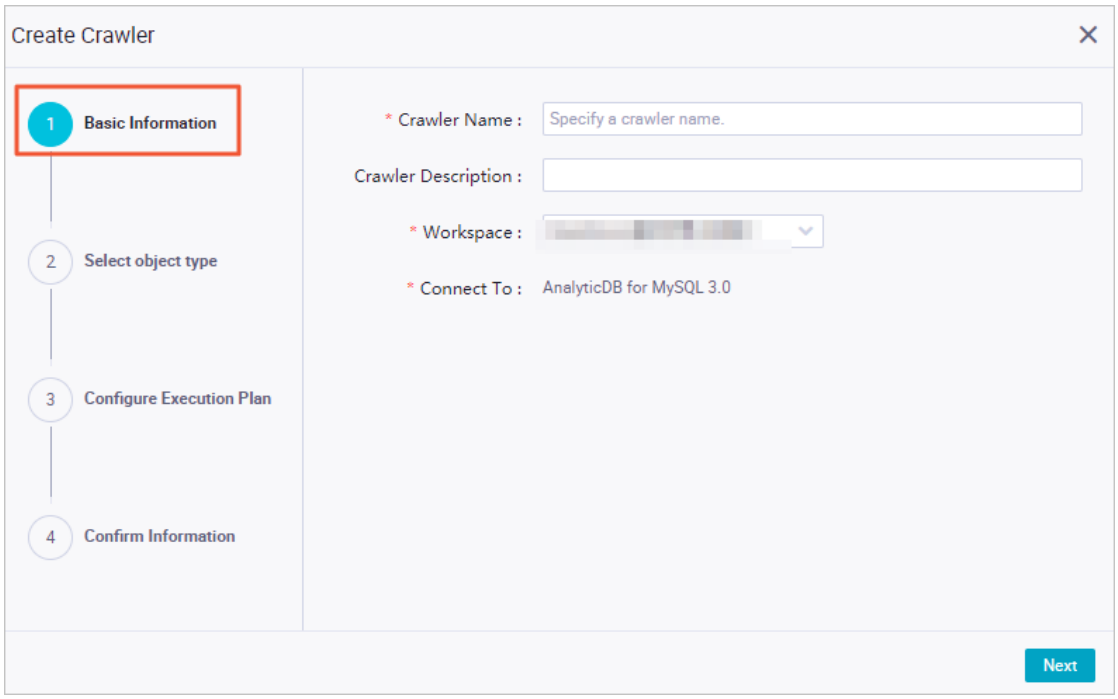
12.7.9. Collect metadata from an AnalyticDB for MySQL 3.0 data source

This topic describes how to create a crawler to collect metadata from an AnalyticDB for MySQL 3.0 data source to DataWorks. You can view the collected metadata on the Data Map page.

Procedure

1. Go to the **Data Discovery** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > DataMap(Data Management)**.
   iii. In the top navigation bar, click **Data Discovery**.
2. In the left-side navigation pane, click **AnalyticDB for MySQL 3.0**.

3. On the **AnalyticDB for MySQL 3.0Metadata Crawler** page, click **Create Crawler**.
4. In the **Create Crawler** dialog box, set the parameters in each step.
   i. In the **Basic Information** step, set the parameters.



Parameter	Description
Crawler Name	Required. The name of the crawler. You must set a unique name.
Crawler Description	The description of the crawler.
Workspace	The workspace of the data source from which you want to collect metadata.
Data Source Type	The type of the data source from which you want to collect metadata. The default value is **AnalyticDB for MySQL 3.0** and cannot be changed.

   ii. Click **Next**.
   iii. In the **Select Collection Object** step, select a data source from the **Data Source** drop-down list.

   If no data sources are available, click **Create** to go to the **Data Source** page and add an AnalyticDB for MySQL V3.0 data source.
   iv. Click Start Testing next to **Test Crawler Connectivity**.
   v. If the message **The connectivity test is successful** appears, click **Next**.

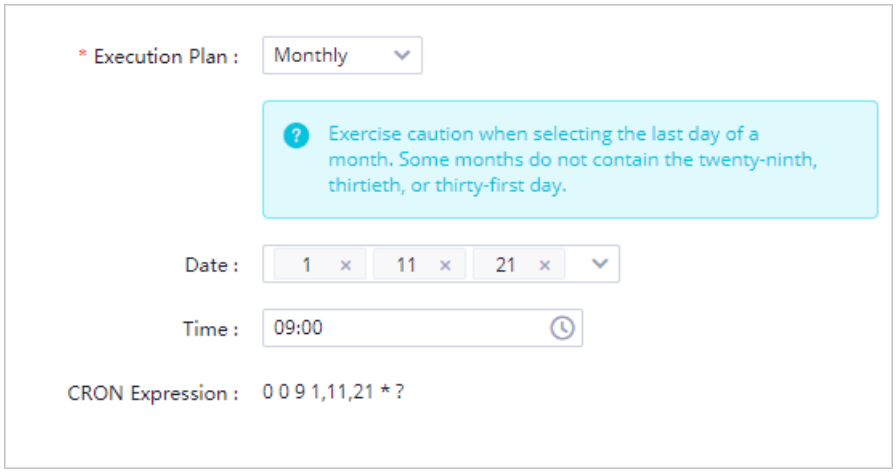
   If the message **The connectivity test of the data source failed, and the data source cannot be connected to the resource group** appears, check whether you have configured a valid data source.
   vi. In the **Configure Execution Plan** step, configure an execution plan.

Valid values of the **Execution Plan** parameter are **On-demand Execution**, **Monthly**, **Weekly**, **Daily**, and **Hourly**. The execution plan that is generated varies based on the execution cycle. The system collects metadata from the AnalyticDB for MySQL 3.0 data source based on the execution cycle that you specify. The following descriptions explain each value and provide examples:

- On-demand Execution: The system collects metadata from the AnalyticDB for MySQL 3.0 data source based on your business requirements.
- Monthly: The system automatically collects metadata from the AnalyticDB for MySQL 3.0 data source once at a specific time on several specific days of each month.

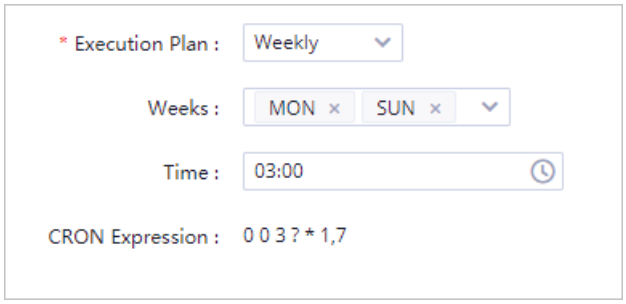
  > **Notice** Some months do not have the 29th, 30th, or 31st day. In these months, the system does not collect metadata from the AnalyticDB for MySQL 3.0 data source on these dates. We recommend that you do not select the last days of a month.

  The following figure shows that the system automatically collects metadata from the AnalyticDB for MySQL 3.0 data source once at 09:00 on the 1st, 11th, and 21st days of each month. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.



- Weekly: The system automatically collects metadata from the AnalyticDB for MySQL 3.0 data source once at a specific time on several specific days of each week.

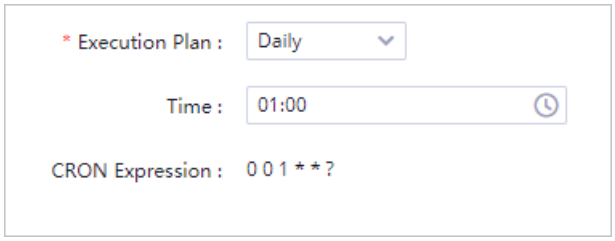
  The following figure shows that the system automatically collects metadata from the AnalyticDB for MySQL 3.0 data source once at 03:00 on Sunday and Monday of each week.



  If the **Time** parameter is not specified, the system automatically collects AnalyticDB for MySQL 3.0 metadata once at 00:00:00 on the specific days of each week.

- Daily: The system automatically collects metadata from the AnalyticDB for MySQL 3.0 data source once at a specific time of each day.

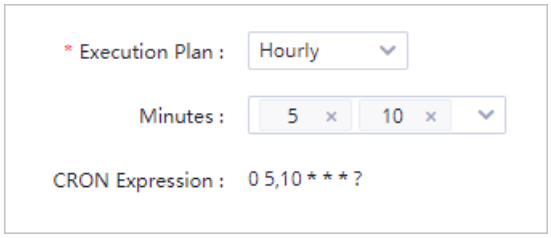
  The following figure shows that the system automatically collects metadata from the AnalyticDB for MySQL 3.0 data source once at 01:00 each day.

- Hourly: The system automatically collects metadata from the AnalyticDB for MySQL 3.0 data source once from the `N × 5` th minute of each hour.

  > **Note** For an AnalyticDB for MySQL 3.0 metadata collection task that is run each hour, you can set the time to a multiple of 5 minutes.

  The following figure shows that the system automatically collects metadata from the AnalyticDB for MySQL 3.0 data source from the 5th and 10th minutes of each hour.

vii. Click **Next**.

viii. In the **Confirm Information** step, check the information that you specified and click **Confirm**.

5. On the **AnalyticDB for MySQL 3.0Metadata Crawler** page, find the created crawler and click **Run** in the Actions column.

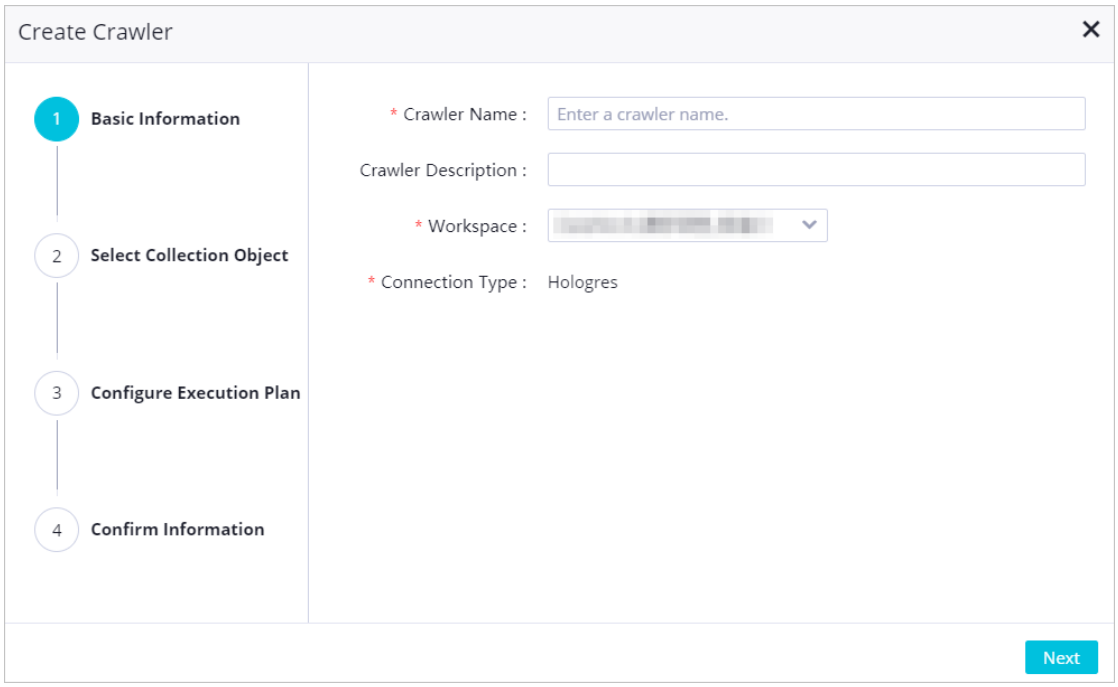
12.7.10. Collect metadata from a Hologres data source

This topic describes how to create a crawler to collect metadata from a Hologres data source to DataWorks. You can view the collected metadata on the Data Map page.

Procedure

1. Go to the **Data Discovery** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > DataMap(Data Management)**.
   iii. In the top navigation bar, click **Data Discovery**.
2. In the left-side navigation pane, click **Hologres**.

3. On the **HologresMetadata Crawler** page, click **Create Crawler**.
4. In the **Create Crawler** dialog box, set the parameters in each step.
   i. In the **Basic Information** step, set the parameters.



Parameter	Description
Crawler Name	Required. The name of the crawler. You must set a unique name.
Crawler Description	The description of the crawler.
Workspace	The workspace of the data source from which you want to collect metadata.
Data Source Type	The type of the data source from which you want to collect metadata. The default value is **Hologres** and cannot be changed.

   ii. Click **Next**.
   iii. In the **Select Collection Object** step, select a data source from the **Data Source** drop-down list.
   iv. Click Start Testing next to **Test Crawler Connectivity**.
   v. If the message **The connectivity test is successful** appears, click **Next**.

   If the message **The connectivity test of the data source failed, and the data source cannot be connected to the resource group** appears, check whether you have configured a valid data source.
   vi. In the **Configure Execution Plan** step, configure an execution plan.

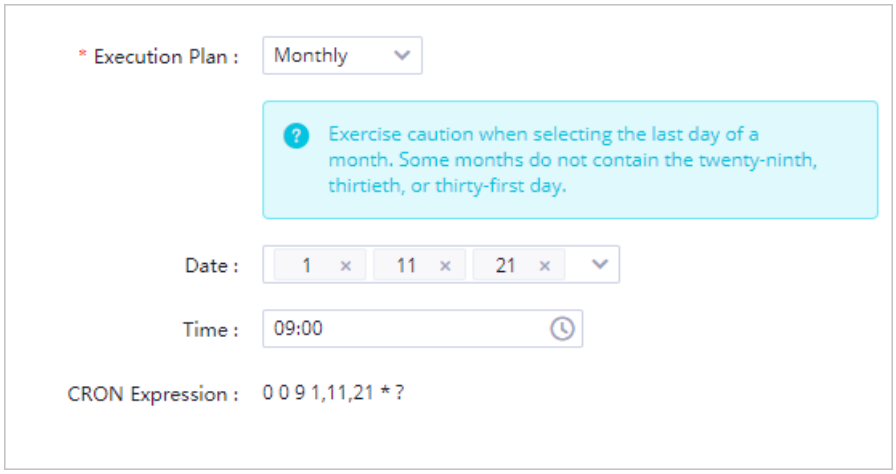
   Valid values of the **Execution Plan** parameter are **On-demand Execution**, **Monthly**, **Weekly**, **Daily**, and **Hourly**. The execution plan that is generated varies based on the

execution cycle. The system collects metadata from the Hologres data source based on the execution cycle that you specify. The following descriptions explain each value and provide examples:

- On-demand Execution: The system collects metadata from the Hologres data source based on your business requirements.
- Monthly: The system automatically collects metadata from the Hologres data source once at a specific time on several specific days of each month.

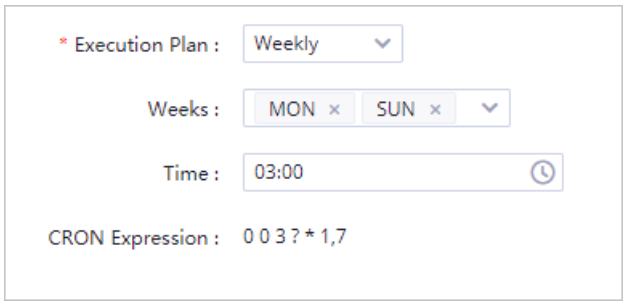
  > **Notice** Some months do not have the 29th, 30th, or 31st day. In these months, the system does not collect metadata from the Hologres data source on these dates. We recommend that you do not select the last days of a month.

  The following figure shows that the system automatically collects metadata from the Hologres data source once at 09:00 on the 1st, 11th, and 21st days of each month. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.

- Weekly: The system automatically collects metadata from the Hologres data source once at a specific time on several specific days of each week.

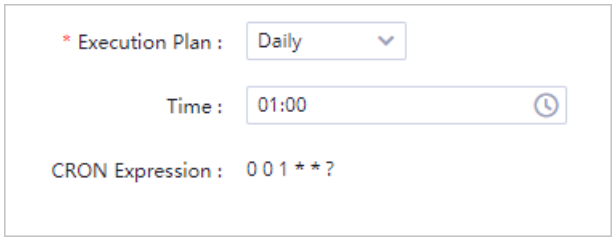
  The following figure shows that the system automatically collects metadata from the Hologres data source once at 03:00 on Sunday and Monday of each week.

  If the **Time** parameter is not specified, the system automatically collects Hologres metadata once at 00:00:00 on the specific days of each week.

- Daily: The system automatically collects metadata from the Hologres data source once at a specific time of each day.

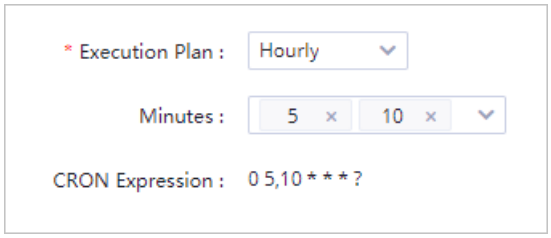
  The following figure shows that the system automatically collects metadata from the Hologres data source once at 01:00 each day.

- Hourly: The system automatically collects metadata from the Hologres data source once from the `N × 5` th minute of each hour.

  > **Note** For a Hologres metadata collection task that is run each hour, you can set the time to a multiple of 5 minutes.

  The following figure shows that the system automatically collects metadata from the Hologres data source from the 5th and 10th minutes of each hour.

vii. Click **Next**.

viii. In the **Confirm Information** step, check the information that you specified and click **Confirm**.

5. On the **HologresMetadata Crawler** page, find the created crawler and click **Run** in the Actions column.

12.7.11. Collect metadata from a CDH Hive data source

DataWorks allows you to collect metadata that includes the table schema and the linage information of tables in Data Map. You can view the schema of a table and the relationships between tables. This topic describes how to create a crawler to collect metadata from a CDH Hive data source to DataWorks. You can view the collected metadata on the Data Map page.

Prerequisites

A CDH cluster is associated with the current DataWorks workspace. For more information, see Associate a CDH cluster with a workspace.

Limits

You cannot collect metadata across regions. You must create a crawler in the region where the source metadata resides to collect the metadata.

Create a crawler

1. Go to the **Data Discovery** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > DataMap(Data Management)**.
   iii. In the top navigation bar, click **Data Discovery**.
2. Create a crawler.
   i. In the left-side navigation pane, click **CDH Hive**.
   ii. On the **CDH Hive Metadata Crawler** page, click **Create Crawler**.
3. Configure the crawler.
   i. Select a CDH cluster.

      In the **Create Crawler** dialog box, select the cluster from which you want to collect metadata from the drop-down list.
   ii. Configure Execution Plan

      In the **Create Crawler** dialog box, specify a value for the **Execution Plan** parameter from the drop-down list.

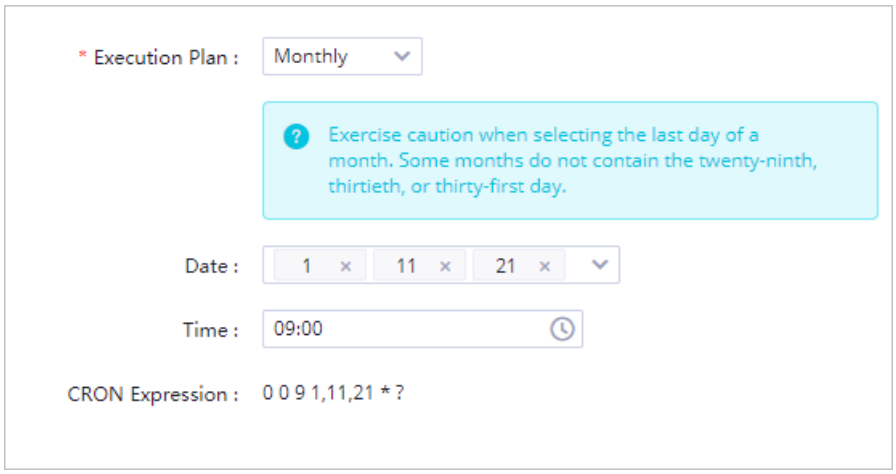
      Valid values of the **Execution Plan** parameter are **On-demand Execution**, **Monthly**, **Weekly**, **Daily**, **Hourly**, and **Customize**. The execution plan that is generated varies based on the execution cycle. The system collects metadata from the CDH Hive data source based on the execution cycle that you specify. The following descriptions explain each value and provide examples:

      - On-demand Execution: You need to manually run the crawler. The system collects metadata from the CDH Hive data source based on your business requirements.

- Monthly: The system automatically collects metadata from the CDH Hive data source once at a specific time on several specific days of each month.

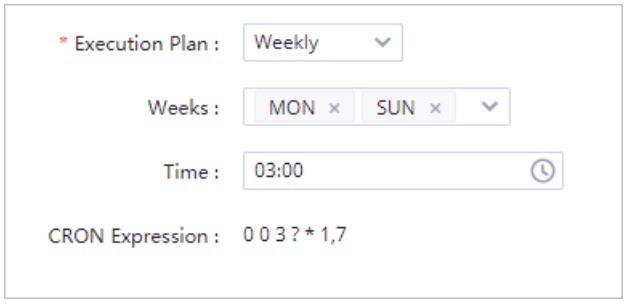
  > **Notice** Some months do not have the 29th, 30th, or 31st day. In these months, the system does not collect metadata from the CDH Hive data source on these dates. We recommend that you do not select the last days of a month.

  The following figure shows that the system automatically collects metadata from the CDH Hive data source once at 09:00 on the 1st, 11th, and 21st days of each month. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.

- Weekly: The system automatically collects metadata from the CDH Hive data source once at a specific time on several specific days of each week.

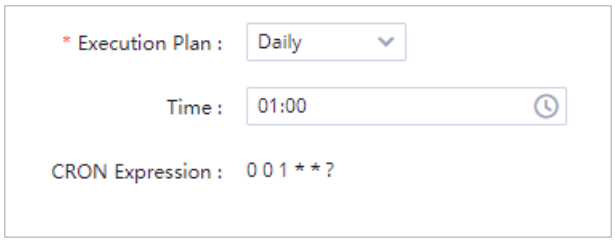
  The following figure shows that the system automatically collects metadata from the CDH Hive data source once at 03:00 on Sunday and Monday of each week. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.

  If the **Time** parameter is not set, the system automatically collects metadata from the CDH Hive data source once at `00:00:00` on the specific days of each week.

- Daily: The system automatically collects metadata from the CDH Hive data source once at a specific time of each day.

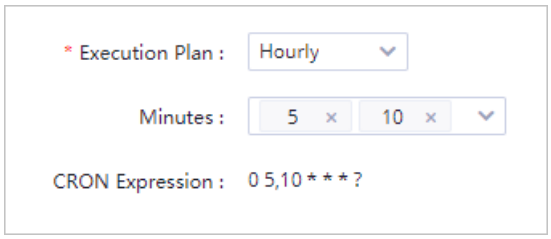
  The following figure shows that the system automatically collects metadata from the CDH Hive data source once at 01:00 each day. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.

- Hourly: The system automatically collects metadata from the CDH Hive data source once from the `N × 5` th minute of each hour.

  > **Note** For a CDH Hive metadata collection task that is run each hour, you can set the time to a multiple of 5 minutes.

  The following figure shows that the system automatically collects metadata from the CDH Hive data source from the 5th and 10th minutes of each hour. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.

- Customize: You can enter a cron expression in the field of the Cron Expression parameter. The system automatically collects metadata based on the time configuration that matches the corn expression.

iii. Click **OK**.

Manage the crawler

On the **CDH Hive Metadata Crawler** page, you can view, edit, and delete the created crawler.

Area No.	Description
1	In this area, you can enter the name of a crawler to find the crawler. > **Note** The fuzzy match is supported. If you enter keywords in the search box, crawlers whose names contain the keywords are displayed.

Area No.	Description
2	In this area, you can view the details of a crawler in the **Status**, **Execution Plan**, **Last Run At**, **Last Consumed Time**, and **Average Running Time** columns. You can also perform the following operations on the crawler: • **Details**: View the CDH cluster and the execution plan that are configured for the crawler. • **Edit**: Modify the CDH cluster and the execution plan that are configured for the crawler. • **Delete**: Delete the crawler. • **Run**: Run the crawler to collect metadata based on the configurations. • **Stop**: Stop the crawler. **Note** **Run** and **Stop** are displayed in the **Actions** column only if the **Execution Plan** parameter is set to **On-demand Execution**.

What's next

After the metadata is collected, you can view the details of the collected metadata on the **All Data** page of Data Map.

12.7.12. Collect metadata from an HBase data source

DataWorks allows you to collect metadata that includes the table schema and the linage information of tables in Data Map. You can view the schema of a table and the relationships between tables. This topic describes how to create a crawler to collect metadata from an HBase data source to DataWorks. You can view the collected metadata on the Data Map page.

Prerequisites

A Cloudera Distribution Hadoop (CDH) cluster is associated with the current DataWorks workspace. For more information, see Associate a CDH cluster with a workspace.

Limits

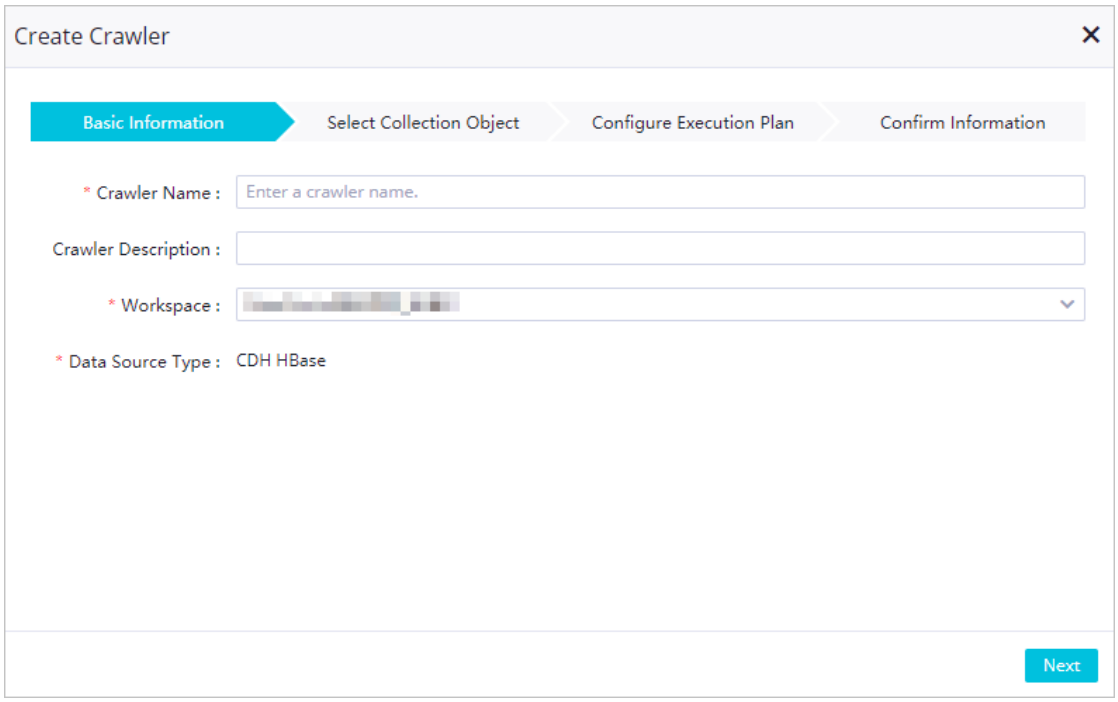
You cannot collect metadata across regions. You must create a crawler in the region where the source metadata resides to collect the metadata.

Create a crawler

1. Go to the **Data Discovery** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > DataMap(Data Management)**.
   iii. In the top navigation bar, click **Data Discovery**.

2. Create a crawler.
   i. In the left-side navigation pane, click **CDH HBase**.
   ii. On the **CDH HBaseMetadata Crawler** page, click **Create Crawler**.
3. Configure the crawler.
   i. Configure the basic information.

   In the **Basic Information** step of the **Create Crawler** dialog box, set the parameters.



Parameter	Description
Crawler Name	The name of the crawler.
Crawler Description	The description of the crawler.
Workspace	The workspace in which the crawler resides.
Data Source Type	The type of the data source from which you want to collect metadata. The default value is CDH HBase.

   ii. Click **Next**.
   iii. Select an HBase data source.

   In the **Select Collection Object** step, select a data source and a resource group from the drop-down lists, and click **Start Testing** to test the connectivity between the data source and the resource group.

   > **Note** If no data sources are available, click **Create** to add an HBase data source on the Data Source page.

   iv. Configure an execution plan.

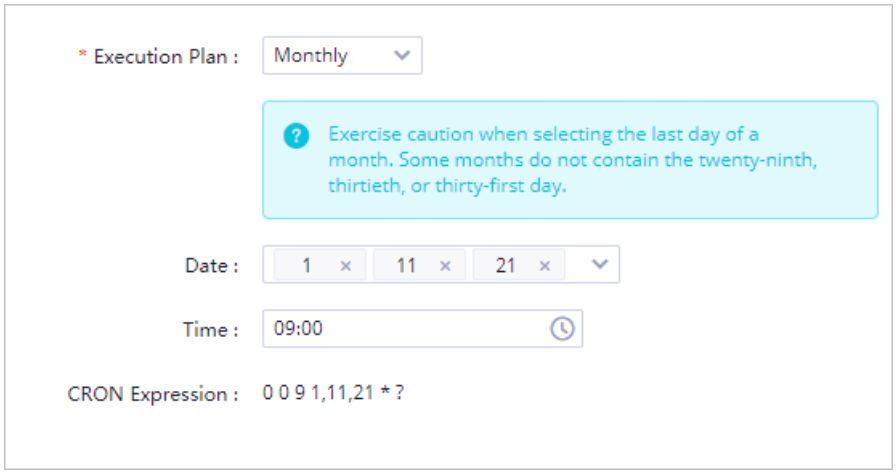
In the **Configure Execution Plan** step, configure an execution plan.

Valid values of the **Execution Plan** parameter are **On-demand Execution**, **Monthly**, **Weekly**, **Daily**, **Hourly**, and **Customize**. The execution plan that is generated varies based on the execution cycle. The system collects metadata from the HBase data source based on the execution cycle that you specify. The following descriptions explain each value and provide examples:

- On-demand Execution: You need to manually run the crawler. The system collects metadata from the HBase data source based on your business requirements.
- Monthly: The system automatically collects metadata from the HBase data source once at a specific time on several specific days of each month.

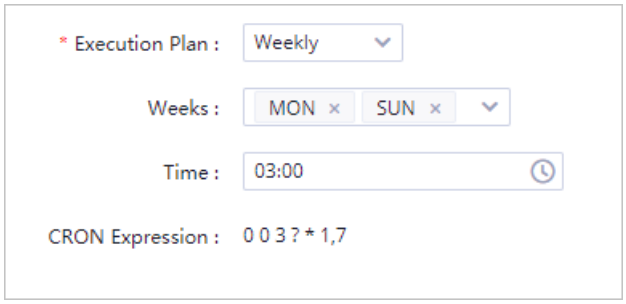
> **Notice** Some months do not have the 29th, 30th, or 31st day. In these months, the system does not collect metadata from the HBase data source on these dates. We recommend that you do not select the last days of a month.

The following figure shows that the system automatically collects metadata from the HBase data source once at 09:00 on the 1st, 11th, and 21st days of each month. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.



- Weekly: The system automatically collects metadata from the HBase data source once at a specific time on several specific days of each week.

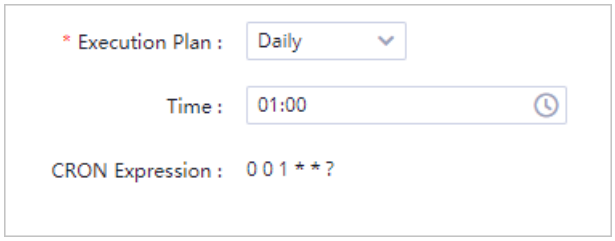
  The following figure shows that the system automatically collects metadata from the HBase data source once at 03:00 on Sunday and Monday of each week. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.

  If the **Time** parameter is not set, the system automatically collects metadata from the HBase data source once at `00:00:00` on the specific days of each week.

- Daily: The system automatically collects metadata from the HBase data source once at a specific time of each day.

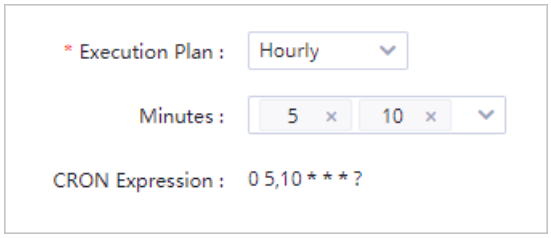
  The following figure shows that the system automatically collects metadata from the HBase data source once at 01:00 each day. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.

- Hourly: The system automatically collects metadata from the HBase data source once from the `N × 5` th minute of each hour.

  > **Note** For an HBase metadata collection task that is run each hour, you can set the time to a multiple of 5 minutes.

  The following figure shows that the system automatically collects metadata from the HBase data source from the 5th and 10th minutes of each hour. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.

- Customize: You can enter a cron expression in the field of the Cron Expression parameter. The system automatically collects metadata based on the time configuration that matches the corn expression.

v. Click **Next**.

vi. In the **Confirm Information** step, check the information that you specified and click **Confirm**.

Manage the crawler

On the **CDH HBaseMetadata Crawler** page, you can view, edit, and delete the created crawler.

Area No.	Description
1	In this area, you can enter the name of the crawler or the name of the data source in the search boxes to find the crawler. **Note** The fuzzy match is supported. If you enter keywords in the search boxes, crawlers whose names or data source names contain the keywords are displayed.
2	In this area, you can view the details of the crawler in the **Status**, **Environment**, **Network Connectivity**, **Execution Plan**, **Last Run At**, **Last Consumed Time**, and **Average Running Time** columns. You can also perform the following operations on the crawler: • **Details**: View the configurations of the crawler. • **Edit**: Modify the configurations of the crawler. For example, you can modify the resource group and the execution plan of the crawler. • **Delete**: Delete the crawler. • **Run**: Run the crawler.

What's next

After the metadata is collected, you can view the details of the collected metadata on the **All Data** page of Data Map.

12.7.13. Collect metadata from a Kudu data source

DataWorks allows you to collect metadata that includes the table schema and the linage information of tables in Data Map. You can view the schema of a table and the relationships between tables. This topic describes how to create a crawler to collect metadata from a Kudu data source to DataWorks. You can view the collected metadata on the Data Map page.

Prerequisites

A CDH cluster is associated with the current DataWorks workspace. For more information, see Associate a CDH cluster with a workspace.

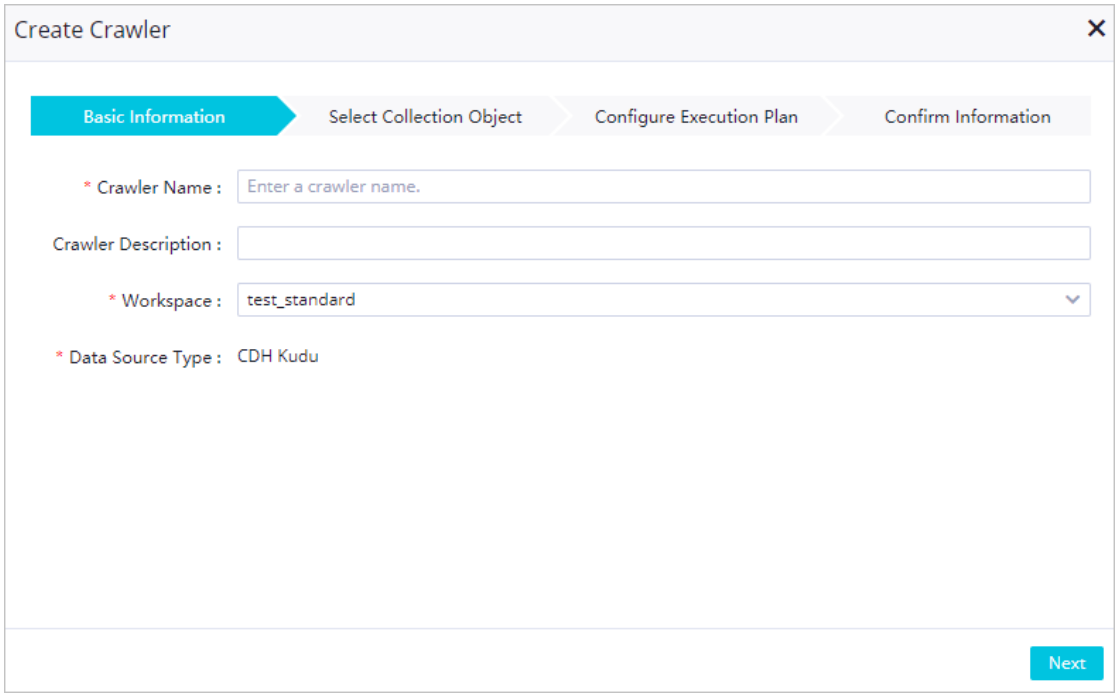
Limits

You cannot collect metadata across regions. You must create a crawler in the region where the source metadata resides to collect the metadata.

Create a crawler

1. Go to the **Data Discovery** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > DataMap(Data Management)**.
   iii. In the top navigation bar, click **Data Discovery**.
2. Create a crawler.
   i. In the left-side navigation pane, click **CDH Kudu**.
   ii. On the **CDH KuduMetadata Crawler** page, click **Create Crawler**.
3. Configure the crawler.
   i. Configure the basic information.

   In the **Basic Information** step of the **Create Crawler** dialog box, set the parameters.



Parameter	Description
Crawler Name	The name of the crawler.
Crawler Description	The description of the crawler.
Workspace	The workspace in which the crawler resides.
Data Source Type	The type of the data source from which you want to collect metadata. The default value is CDH Kudu.

ii. Click **Next**.

iii. Select a Kudu data source.

In the **Select Collection Object** step, select a data source and a resource group from the drop-down lists, and click **Start Testing** to test the connectivity between the data source and the resource group.

> **Note** If no data sources are available, click **Create** to add a Kudu data source on the Data Source page.

iv. Configure an execution plan.

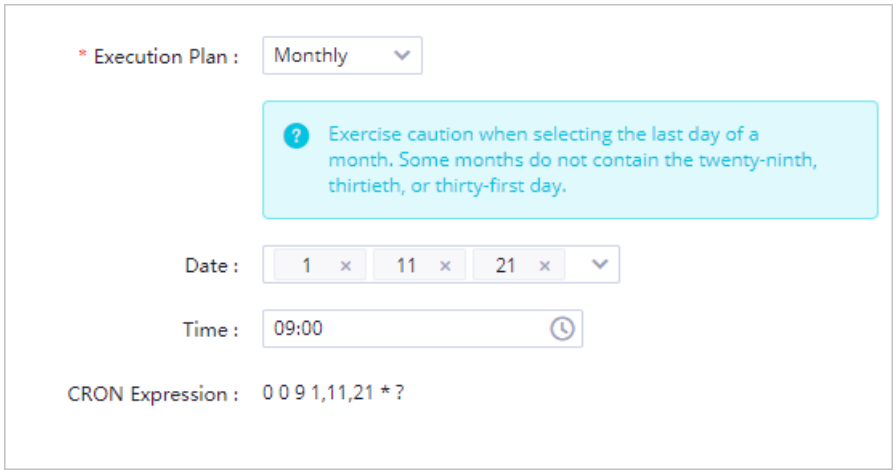
In the **Configure Execution Plan** step, configure an execution plan.

Valid values of the **Execution Plan** parameter are **On-demand Execution**, **Monthly**, **Weekly**, **Daily**, **Hourly**, and **Customize**. The execution plan that is generated varies based on the execution cycle. The system collects metadata from the Kudu data source based on the execution cycle that you specify. The following descriptions explain each value and provide examples:

- On-demand Execution: You need to manually run the crawler. The system collects metadata from the Kudu data source based on your business requirements.
- Monthly: The system automatically collects metadata from the Kudu data source once at a specific time on several specific days of each month.

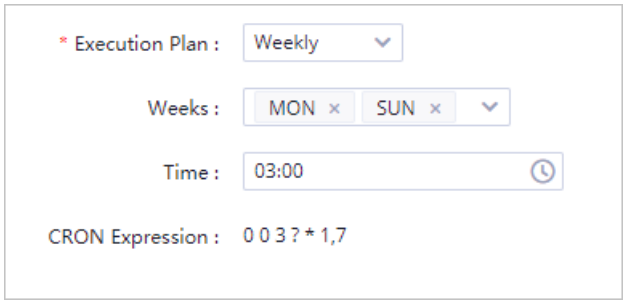
> **Notice** Some months do not have the 29th, 30th, or 31st day. In these months, the system does not collect metadata from the Kudu data source on these dates. We recommend that you do not select the last days of a month.

The following figure shows that the system automatically collects metadata from the Kudu data source once at 09:00 on the 1st, 11th, and 21st days of each month. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.



- Weekly: The system automatically collects metadata from the Kudu data source once at a specific time on several specific days of each week.

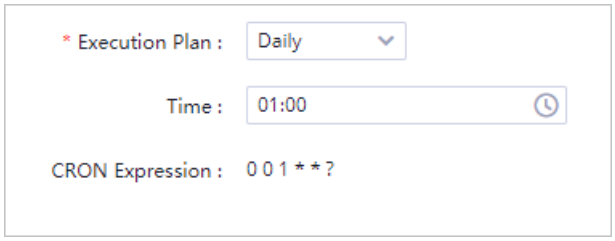
  The following figure shows that the system automatically collects metadata from the Kudu data source once at 03:00 on Sunday and Monday of each week. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.

  If the **Time** parameter is not set, the system automatically collects metadata from the Kudu data source once at `00:00:00` on the specific days of each week.

- Daily: The system automatically collects metadata from the Kudu data source once at a specific time of each day.

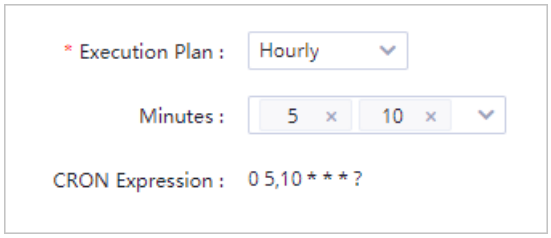
  The following figure shows that the system automatically collects metadata from the Kudu data source once at 01:00 each day. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.

- Hourly: The system automatically collects metadata from the Kudu data source once from the `N × 5` th minute of each hour.

  > **Note** For a Kudu metadata collection task that is run each hour, you can set the time to a multiple of 5 minutes.

  The following figure shows that the system automatically collects metadata from the Kudu data source from the 5th and 10th minutes of each hour. An expression is automatically generated for the **Cron Expression** parameter based on the values of the Date and Time parameters.

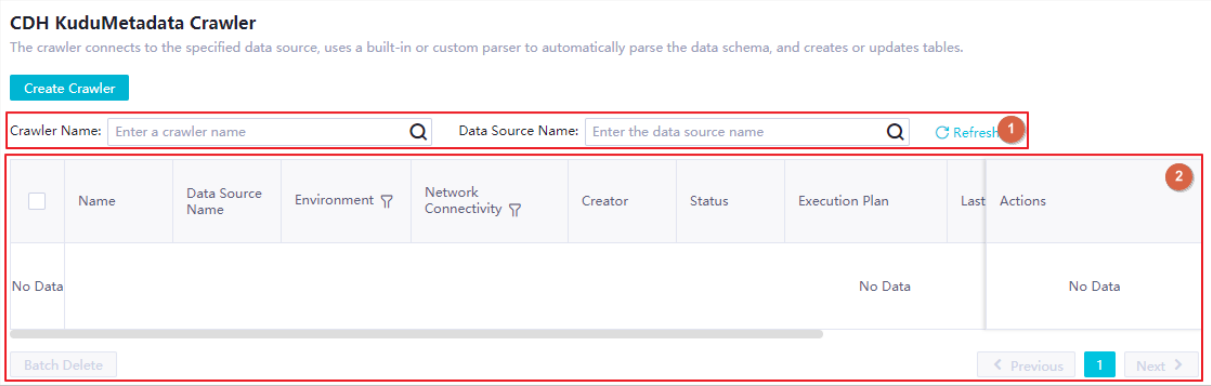
- Customize: You can enter a cron expression in the field of the Cron Expression parameter. The system automatically collects metadata based on the time configuration that matches the corn expression.

v. Click **Next**.

vi. In the **Confirm Information** step, check the information that you specified and click **Confirm**.

Manage the crawler

On the **CDH KuduMetadata Crawler** page, you can view, edit, and delete the created crawler.



Area No.	Description
1	In this area, you can enter the name of the crawler or the name of the data source in the search boxes to find the crawler. **Note** The fuzzy match is supported. If you enter keywords in the search boxes, crawlers whose names or data source names contain the keywords are displayed.
2	In this area, you can view the details of the crawler in the **Status**, **Environment**, **Network Connectivity**, **Execution Plan**, **Last Run At**, **Last Consumed Time**, and **Average Running Time** columns. You can also perform the following operations on the crawler: • **Details**: View the configurations of the crawler. • **Edit**: Modify the configurations of the crawler. For example, you can modify the resource group and the execution plan of the crawler. • **Delete**: Delete the crawler. • **Run**: Run the crawler.

What's next

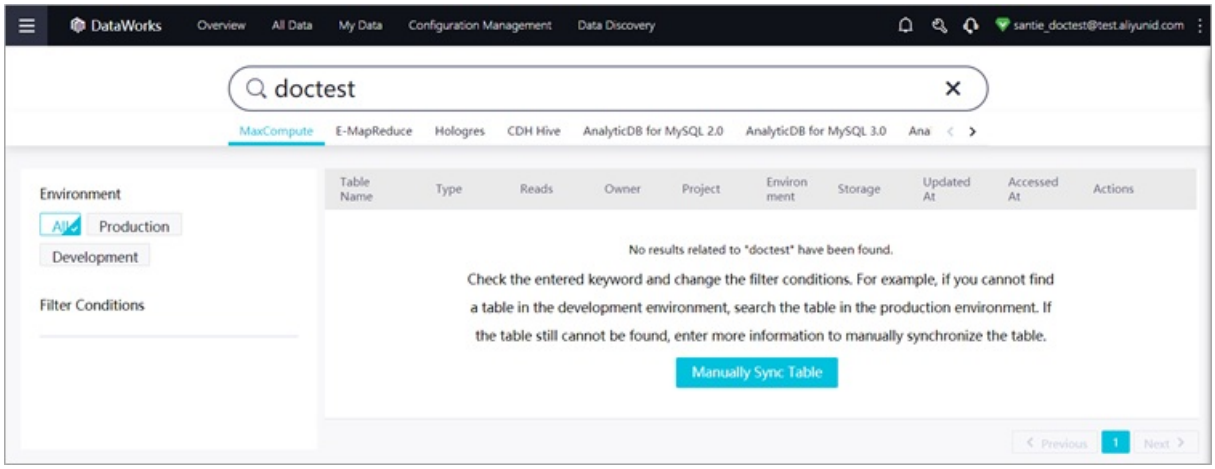
After the metadata is collected, you can view the details of the collected metadata on the **All Data** page of Data Map.

12.8. What do I do if no search results are returned when I query a newly created table in the Data Map service of DataWorks?

The Data Map service of DataWorks allows you to query tables on the homepage of Data Map by using a keyword. It also allows you to view the tables that you have recently browsed or read. If no search results are returned when you query a newly created table in the Data Map service of DataWorks, the possible cause is a latency of DataWorks in obtaining metadata. This topic describes how to address this issue.

Problem description

When I query a newly created table on the **All Data** page of Data Map by using a keyword, the **No data found.** message is displayed.



Possible causes

Possible causes:

- Possible cause 1: You entered an invalid keyword and no tables that match the keyword can be found.
- Possible cause 2: The table that you want to query has just been created, and no search results can be returned because of a latency of DataWorks in obtaining metadata.

Solution

1. Check the keyword that you have entered.

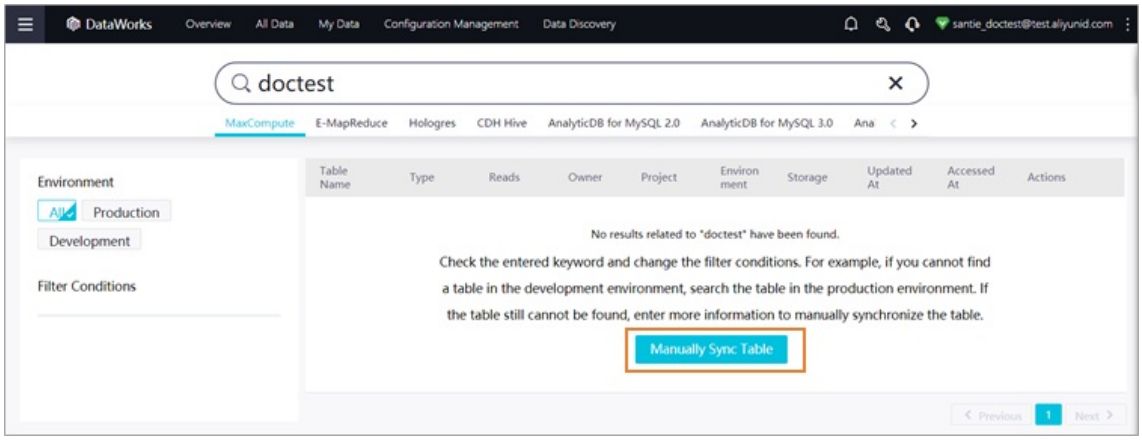
   Make sure that the keyword that you have entered is valid and matches the table that you want to query.

2. Manually synchronize the table.

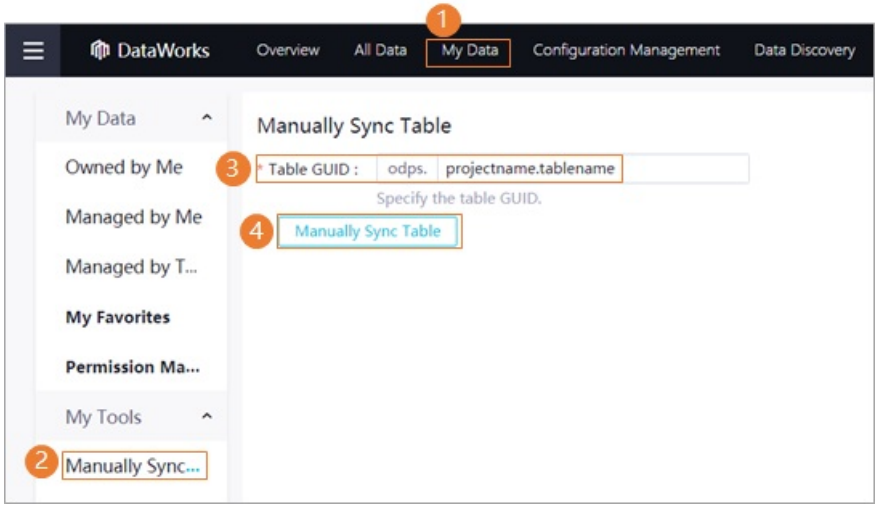
   If the table that you want to query exists and the keyword that you have entered is correct, the possible cause is that the metadata of the newly created table has not been synchronized. In this case, you need to manually synchronize the table. Use one of the following methods to

synchronize the table:

- Click **Manually Sync Table** on the query result page.



- Choose **My Data** > **Manually Sync Table** in Data Map of the DataWorks console. On the page that appears, `set` **Table GUID** `to a value in the format of Project name.Table name, and click Manually Sync Table.`



After you complete the preceding steps, query the table on the All Data page of Data Map by using the keyword again.

13.Data Asset Management

13.1. Go to the Data Asset Management page

Data Asset Management provides you with an overview of your data assets. Data Asset Management requires that data be synchronized by using Data Integration and processed by using DataStudio before you manage your tables and APIs stored in your business system and DataWorks.

Context

Data Asset Management controls the permissions of users independently. You must grant the permissions on the Project Management page because Data Asset Management is a tenant-level feature.

Data Asset Management allows you to view the metadata collected in Data Map. You can also perform basic management operations on the metadata. For example, you can change the business classes and add business descriptions for metadata tables.

Procedure

1. Log on to the DataWorks console.
2. Click ☰ in the upper-left corner and choose **All Products > Project Management**. The Project Management page appears.
3. In the left-side navigation pane, click **Member Management**.

   On the Member Management page, you can assign the following roles to members: **Data Asset Management-Asset Manager**, **Data Asset Management-Class Manager**, and **Data Asset Management-Home Visitor**.
4. Click ☰ in the upper-left corner and choose **All Products > Data Asset Management** to manage data assets in Data Asset Management.

   The following table describes the permissions of each role.

Role	Permission
Data Asset Management-Asset Manager	In the top navigation bar of the **Data Asset Management** page, click the **Assets** tab. On the Assets tab, you can manage data assets, such as adding business units and classes. You can also add data assets to a class.
Data Asset Management-Class Manager	In the top navigation bar of the **Data Asset Management** page, click the **Classes** tab. On the Classes tab, you can view the data assets of each class.
Data Asset Management-Home Visitor	In the top navigation bar of the **Data Asset Management** page, click the **Home** tab. On the Home tab, you can view the statistics of data assets of different classes and business units.

13.2. Asset manager

Asset managers can view the information about data assets.

Procedure

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Data Asset Management**.
3. On the **Home** page that appears, enter keywords in the search box and click **Search**.
4. On the search results page that appears, click the **Tables**, **File**, or **API** tab to view details and apply for permissions.

   You can click the **Classes** tab in the top navigation bar to filter data assets by class.

13.3. Asset user

Asset users can access Data Asset Management to perform operations such as searching for assets, applying for permissions, and using assets.

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Data Asset Management**.
3. On the **Home** page that appears, enter keywords in the search box and click **Search**.
4. On the search results page that appears, click the **Tables**, **File**, or **API** tab to view details and apply for permissions.

   You can click the **Classes** tab in the top navigation bar to filter data assets by class.

13.4. Asset administrator

Asset administrators can manage assets and authorizations in Data Asset Management.

> **Note** You can submit a ticket to apply for the asset administrator role.
>
> An administrator can grant the administrator role to common users. An administrator can perform any operations in Data Asset Management, and no approval is required.

Go to the Assets tab

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Data Asset Management**.
3. In the top navigation bar, click the **Assets** tab.

   Under the **Assets** tab, you can click **Data Management**, **Classes**, or **Business Management** in the left-side navigation pane to manage your data assets accordingly.

Manage data

In the left-side navigation pane, you can click **Data Management** and then **Tables**, **Files**, or **APIs** to manage tables, files, or APIs.

- **Manage tables**

  In the left-side navigation pane, choose **Data Management > Tables** to go to the **Tables** page.

  On the Tables page, you can view, edit, publish, or delete a table.

  - Click the name of a table to view table details.
  - Click **Edit** in the Actions column of a table. In the Edit dialog box that appears, you can edit the configuration of the table.
  - Move the pointer over **Publish** in the Actions column of a table. In the dialog box that appears, click **Publish**. After the request is submitted, click the **Permissions** tab in the top-navigation bar. In the left-side navigation pane, click **Submitted by Me** to view the request you submitted.

    > **Note** You can search for a published table in Data Asset Management.

  - Move the pointer over **Delete** in the Actions column of a table. In the dialog box that appears, click **Delete**. After the request is submitted, click the **Permissions** tab in the top-navigation bar. In the left-side navigation pane, click **Submitted by Me** to view the request you submitted.

    > **Note** You cannot search for a deleted table in Data Asset Management.

- **Manage files**

  In the left-side navigation pane, choose **Data Management > Files** to go to the **Files** page.

  On the **File Management** page that appears, you can upload a file. Then, you can view, edit, download, or delete the file.

  - In the upper-right corner, click **Upload File**. In the **Upload File** dialog box that appears, click **Add File**. In the Add dialog box that appears, select the file to be uploaded and click **Open**. Alternatively, you can drag and drop a file to the **Upload File** dialog box. Then, click **Next**.

    > **Note**
    > - To upload a file, make sure that the size of the file does not exceed 50 MB.
    > - You can only upload a file with one of the following file name extensions:
    >
    >   3DX, 7Z, A3D, ATX, AVI, BMP, SV, DBF, DOC, DOCX, DWG, EPS, ESP, FREELIST, GDB, GDBINDEXES, GDBTABLE, GDBTABLX, GIF, GZ, HTM, HTML, IVE, JPEG, JPG, LOCK, LSP, LST, MP3, MP4, MPJ, OSG, OSGB, PDF, PNG, PPT, PPTX, PRJ, PSD, RAR, S3C, SBN, SBX, SCP, SHP, SPX, TFW, TIF, TIFF, TTF, TXT, WAV, WL, WP, WT, XLS, XLSX, ZIP, XML, SHX, and SKP

  - Click the name of a file to view file details.
  - Click **Edit** in the Actions column of a file. In the Edit dialog box that appears, you can edit the configuration of the file.

- Move the pointer over **Publish** in the Actions column of a file. In the dialog box that appears, click **Publish**. After the request is submitted, click the **Permissions** tab in the top-navigation bar. In the left-side navigation pane, click **Submitted by Me** to view the request you submitted.

  > **Note** You can search for a published file in Data Asset Management.

- Move the pointer over **Delete** in the Actions column of a file. In the dialog box that appears, click **Delete**. After the request is submitted, click the **Permissions** tab in the top-navigation bar. In the left-side navigation pane, click **Submitted by Me** to view the request you submitted.

  > **Note** You cannot search for a deleted file in Data Asset Management.

- Click **Download** in the Actions column of a file to download the file.

  > **Note** Before downloading a file, apply for the download permission.

- **Manage APIs**

  In the left-side navigation pane, choose **Data Management > APIs** to go to the **APIs** page.

  On the **APIs** page, you can edit, publish, or delete an API.

  - Click **Edit** in the Actions column of an API. In the **Edit** dialog box that appears, you can edit the configuration of the API. Then, click **Submit**.
  - Move the pointer over **Publish** in the Actions column of an API. In the dialog box that appears, click **Publish**. After the request is submitted, click the **Permissions** tab in the top-navigation bar. In the left-side navigation pane, click **Submitted by Me** to view the request you submitted.

    > **Note** You can search for a published API in Data Asset Management.

  - Move the pointer over **Delete** in the Actions column of an API. In the dialog box that appears, click **Delete**. After the request is submitted, click the **Permissions** tab in the top-navigation bar. In the left-side navigation pane, click **Submitted by Me** to view the request you submitted.

    > **Note** You cannot search for a deleted API in Data Asset Management.

Manage classes

1. In the left-side navigation pane, click **Classes** to go to the **Classes** page.

   On the Classes page, you can import or export a class.

2. Click the icon. In the **Add Class** dialog box that appears, set relevant parameters and click **OK** to add a level-1 class

Parameter	Description
Name	The name of the class, which can be up to 128 characters in length.

Parameter	Description
Code	The code of the class. This parameter cannot be left empty.
Description	The description of the class.
Confidential	Specifies whether the class is confidential. Valid values: **Yes** and **No**.
Share	Specifies whether to share the class. Valid values: **Yes**, **Conditional**, and **No**.

To create a subclass under a class, click the icon next to the class.

3. Click a class. On the page that appears, click the **Tables** tab.
4. Click **Add Table**. In the **Add Table** dialog box that appears, select the tables to be added to the class and click **OK**.

   You can add files and APIs to a class in the same way.

   To change the class of a table, click **Modify Class** in the Actions column. In the **Change Class** dialog box that appears, change the class as needed.

Manage business

In the left-side navigation pane, you can click **Business Management** and then **Business Units**, **Business Systems**, or **Connections** to manage business units, business systems, or connections.

> **Note** Connections belong to a business system, and business systems belong to a business unit.
>
> - A business system with connections cannot be deleted.
> - A business unit with business systems cannot be deleted.

- **Manage business units**

  In the left-side navigation pane, choose **Business Management > Business Units** to go to the **Business Units** page.

  Click the icon. In the **Add Business Unit** dialog box that appears, set relevant parameters and click **OK** to add a business unit.

Parameter	Description
Name	The name of the business unit. This parameter cannot be left empty.
Code	The code of the business unit. By default, the code cannot be modified.
Description	The description of the business unit.

Parameter	Description
Confidential	Specifies whether the business unit is confidential. Valid values: **Yes** and **No**.
Share	Specifies whether to share the business unit. Valid values: **Yes**, **Conditional**, and **No**.
Business system included	Select the business systems to be added to the business unit and click the **>** icon.

To create a sub-business unit under a business unit, click the icon next to the business unit.

- **Manage business systems**

  In the left-side navigation pane, choose **Business Management > Business Systems** to go to the **Business Systems** page.

  On the Business Systems page, you can add a business system. Then, you can view, edit, or delete the business system.

  - Click **Add Business System**. In the **Basic information** dialog box that appears, set relevant parameters and click **Submit**.
  - Click **View** in the Actions column of a business system to view its details.
  - Click **Edit** in the Actions column of a business system. In the **Business System Properties** dialog box that appears, you can edit the configuration of the business system.
  - Click **Delete** in the Actions column of a business system. In the **Delete business system** dialog box that appears, click **OK** to delete the business system.

- **Manage connections**

  In the left-side navigation pane, choose **Business Management > Connections** to go to the **Connections** page.

  On the **Connections** page, you can view the information of connections. The connection information includes the connection name, the number of tables, the owner, the business system to which the connection belongs, the data type, and the update time. You can also edit the configuration of a connection.

13.5. Manage authorizations

Under the Permissions tab, you can view permissions in different states on the Submitted by Me, To Be Handled, Handled by Me, and My Permissions pages respectively.

Go to the Permissions tab

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Data Asset Management**.
3. In the top navigation bar, click the **Permissions** tab.

   Under the **Permissions** tab, you can view permissions in different states on the **Submitted by Me**, **To Be Handled**, **Handled by Me**, and **My Permissions** pages respectively.

Submitted by Me

In the left-side navigation pane, click **Submitted by Me**.

On the **Submitted by Me** page, you can view request details or cancel requests submitted by you. To resubmit a request that was not approved, find the target request and click **Reapply**.

To Be Handled

In the left-side navigation pane, click **To Be Handled**. On the To Be Handled page, you can view request details and approve or reject requests.

Handled by Me

In the left-side navigation pane, click **Handled by Me** to view requests handled by you.

My Permissions

In the left-side navigation pane, click **My Permissions**. On the My Permissions page, you can view your permissions on tables, files, and APIs respectively.

13.6. Perform cross-tenant authorization

Authorization logic

The logic of authorization within a tenant and that of cross-tenant authorization are as follows:

- Authorization within a tenant: An access control list (ACL) is used for authorization. You need to add the applicant to the corresponding workspace and then grant permissions to the applicant as requested.
- Cross-tenant authorization: A package is used for authorization. First, check whether the workspace where the requested resources reside has a package.
  - If the workspace does not have a package, create a package and add the requested resources to the package. Then, install the package in the workspace where the requested resources are used.

    After that, use an ACL to grant permissions to the applicant as requested.
  - If the workspace has a package, add the requested resources to the package.

  If the requested resources need to be shared with multiple workspaces, install the package in all these workspaces. If the workspace where the requested resources reside has multiple packages, install all of them in the workspaces where the requested resources are used.

Procedure

1. Log on to the DataWorks console. On the DataStudio page, click the DataWorks icon in the upper-left corner and choose **All Products > Data Asset Management**.
2. On the **Asset Portal** tab, enter the name of the table that belongs to another tenant in the search box and click **Search**.
3. On the **Asset Category** tab that appears, click the name of the found table on the **Tables** tab. The details page of the table appears.

On the details page, you can view details about the table, including the basic information, business information, physical information, and field information.

4. Click **Request Permission** in the upper-right corner. The **Table Permission Request** page appears.
5. Set parameters on the **Table Permission Request** page.

Parameter	Description
Target Workspace	The workspace where the table is to be used.
Environment	The type of the environment where the table is to be used. If the selected workspace is in standard mode, you need to set this parameter to Development or Production.
Home Workspace	The workspace where the table resides. This parameter is automatically set.
Home Tenant	The tenant to which the table belongs. This parameter is automatically set.
Grant To	The account for which permissions on the table are requested. Valid values: **Current Account** and **System Account for Production Environment**.
Requested Period	The period in which the requested permissions are valid.
Reason for Request	The reason why you request the permissions. To improve the review efficiency, we recommend that you describe the reason in detail.
Objects Requested	The tables you want to use. By default, the current table is selected. You can select other tables.

6. After the configuration is completed, click **Submit**.

   After you submit the request, click **Approval Management** in the top navigation bar and then click **Submitted by Me**. You can view the review progress on the page that appears.

   - If you want to revoke a request, click **Revoke** in the Actions column of the request.
   - If you want to submit a revoked or rejected request again, click **Reapply** in the Actions column of the request.
7. Log on to the DataWorks console as the table owner, go to the **Approval Management** tab and click **To Be Handled**. On the page that appears, review the request information and click **Agree**.

   After clicking **Agree**, click **Approval Management** in the top navigation bar and click **Handled by Me**. On the page that appears, you can view your authorization records and revoke specific authorizations.

14.Organization management

14.1. Member management

1. Log on to the DataWorks console as a workspace administrator.
2. Move your pointer over the DataWorks icon in the upper-left corner, and click **Project Management**.
3. Click **Member Management** in the left-side navigation bar. The **Members** page appears.
4. You can enter a member name or logon name in the search box to search for a member to be added or removed from the current organization.
   - Assign a role

     To assign a role to a member, click the **Roles** drop-down list next to the member, and select the role to be assigned.

     To unassign a role from a member, click **x** next to the role.
   - Remove a member from an organization

     Click **Delete** next to the member, and click **OK** in the **Remove from Tenant** dialog box that appears.

14.2. Resource groups

14.2.1. About scheduling resources

You can use the Scheduling Resource page to create, configure, and edit a scheduling resource.

A scheduling resource is an object within an organization. A dedicated scheduling resource may contain multiple physical machines or ECS instances that are used to implement a specific task.

1. Log on to the DataWorks console.
2. In the left-side navigation pane, choose **Organization Management > Scheduling Resources**.

> **Note** On the **Scheduling Resources** page, the tenant administrator can create a dedicated scheduling resource, and edit an existing scheduling resource.

14.2.2. Change the workspace of scheduling resources

You can change the workspace of dedicated scheduling resources that have been created and configured.

Procedure

To change the workspace of dedicated scheduling resources, the tenant administrator performs the following operations:

1. Log on to the DataWorks console as a tenant administrator.
2. Choose **Organization Management > Scheduling Resources**.

3. On the page that appears, enter a scheduling resource name for a fuzzy search to find the target scheduling resource.
4. Click **Change Workspace**.
5. Click **OK**.

14.3. Configure the compute engine

Currently, DataWorks only supports MaxCompute as its compute engine. All business flows and nodes in a workspace are run on the MaxCompute project associated to the workspace.

Example

> **Note** Tenant administrators can modify the settings for MaxCompute projects. The following settings are changeable: the project description, whether to use the MaxCompute project owner account to run MaxCompute jobs, the account used for running MaxCompute jobs, and the AccessKey of the account.

Assume that the account used for running MaxCompute jobs is no longer available, for example, because the account owner has resigned. If Run MaxCompute Task Using MaxCompute Owner Account is not selected, the tenant administrator needs to immediately modify the account used for running MaxCompute jobs and its AccessKey so that tasks can properly run in the workspace that uses the corresponding MaxCompute project.

Procedure

You can modify the account used for running MaxCompute jobs and its AccessKey as follows:

1. Log on to the DataWorks console as a tenant administrator. For more information, see Log on to the DataWorks console.
2. Choose **Project Management > Compute Engine**.
3. In the search box on the **Project Management > Compute Engine** page, enter the compute engine name. Fuzzy search is supported.
4. Find the target compute engine, and click **Configure** in the Actions column.
5. In the **Configure Compute Engine** dialog box, specify the Alibaba Cloud Account and the AccessKey.

   > **Note** You can also select Run MaxCompute Task Using MaxCompute Owner Account or create a new Alibaba Cloud account.

6. Click **Submit**.

15.Data Service

15.1. Overview

DataService Studio aims to build a data service bus to help enterprises centrally manage private and public APIs.

DataService Studio allows you to create APIs based on data tables. You can also register existing APIs to DataService Studio for centralized management. DataService Studio and API Gateway are interconnected. This allows you to publish APIs to API Gateway with ease. DataService Studio works together with API Gateway to provide a secure, stable, cost-effective, and easy-to-use data sharing service.

DataService Studio adopts a serverless architecture. This allows you to focus on the query logic of the API without worrying about the infrastructure, such as compute resources. DataService Studio supports automatic scaling for compute resources. This way, you can significantly reduce O&M costs.

Create an API

DataService Studio allows you to create APIs based on tables in relational databases and NoSQL databases. You can create an API in the codeless user interface (UI) within a few minutes without the need to write code. For more information, see Create an API in the codeless UI.

DataService Studio also allows you to create an API in the code editor. You can write SQL statements to customize the query logic of the API. In the code editor, you can specify multi-table join queries, complex query criteria, and aggregate functions. For more information, see Create an API in the code editor.

Register an API

You can register existing RESTful APIs to DataService Studio to manage them together with the APIs that are created in DataService Studio based on tables. Four request methods and three data formats are supported. The four request methods are GET, POST, PUT, and DELETE. The three data formats are tables, JSON, and XML.

API Gateway

API Gateway provides API lifecycle management services, including API publishing, management, maintenance, and monetization. API Gateway provides a simple, fast, cost-effective, and low-risk service for you to aggregate microservices, separate the frontend from the backend, integrate systems, and provide features and data to partners and developers.

Being integrated with API Gateway, DataService Studio allows you to publish APIs to API Gateway conveniently. Both APIs that you create based on data tables and APIs that you register to DataService Studio can be published to API Gateway for management, for example, for authorization, authentication, throttling, and billing.

Sell APIs in Alibaba Cloud Marketplace

The API sector of Alibaba Cloud Marketplace provides thousands of API products in the following categories: finance, AI, e-commerce, transportation and geography, living services, corporate management, and public affairs. Alibaba Cloud Marketplace provides a platform where you can monetize your data.

After you publish APIs that are created or registered in DataService Studio to API Gateway, you can publish them to Alibaba Cloud Marketplace with a few clicks. This is an easy way to achieve financial gains for your enterprise.

15.2. Terms

This section introduces terms of Data Service.

Name	Description
Data source	Indicates database links. Data Service accesses data through data sources. Data sources are configured in Data Integration.
Create an API	Creates APIs based on data tables.
Register an API	Registers existing APIs to Data Service for unified management.
Wizard mode	Guides you through the procedure of API creation. This method is suitable for beginners who want to create simple APIs. You do not need to write any code.
Script mode	Allows you to create APIs by writing SQL scripts. This method supports associative tables, complex queries, and aggregate functions. This method is suitable for experienced developers who want to create complex APIs.
API group	Indicates a set of APIs for a specific scenario or for consuming a specific service. An API group is the smallest group unit in Data Service, and the smallest unit for API Gateway management. API groups are published in Alibaba Cloud API Marketplace as API products.
API Gateway	Indicates a hosted service provided by Alibaba Cloud to manage APIs. API Gateway supports API lifecycle management, permission management, access management, and traffic control.

15.3. Manage tags

This topic describes how to create, add, view, and remove tags for an API.

DataService Studio allows you to add tags to APIs when you manage workflows and create, register, and deploy APIs. In this topic, the scenario of creating an API in the codeless user interface (UI) is used. Tags allow you to efficiently classify and search for APIs. You can maintain only one tag list in a workspace, and cannot use the tag list of a workspace in another workspace.

> Note

- Each API supports zero to five tags. You can add no tag to an API at all or add up to five tags to an API.
- A tag can be up to 20 characters in length and can contain letters, digits, and underscores (_).

Create a tag for an API

1. Log on to the DataWorks console, click the DataWorks icon in the upper-left corner, and then choose **All Products > Data Service**.

2. Move the pointer over the +Create icon and choose **Create API > Generate API**.
3. In the **Generate API** dialog box, set the API Mode parameter to Wizard Mode and enter a tag in the **Label** field.

   If the tag you entered does not exist in the current workspace, Add <Tag> appears in the drop-down list. Click Add <Tag> to create the tag for the API.
4. Configure parameters as required and click **OK**. The tag is created for the API and appears in the tag list of the current workspace. For more information about other parameters in the Generate API dialog box, see Generate APIs in wizard mode.

Add an existing tag to an API

1. Log on to the DataWorks console, click the DataWorks icon in the upper-left corner, and then choose **All Products > Data Service**.
2. Move the pointer over the +Create icon and choose **Create API > Generate API**.
3. In the **Generate API** dialog box, set the API mode parameter to Wizard Mode and click a blank area or the downward arrow in the **Label** field. Tags in the current workspace appear in a drop-down list.
4. Click the required tag, configure parameters, and then click **OK**. The tag is added to the API. For more information about other parameters in the Generate API dialog box, see Generate APIs in wizard mode.

View tags of an API

1. Log on to the DataWorks console, click the DataWorks icon in the upper-left corner, and then choose **All Products > Data Service**.
2. On the **Service Development** tab, double-click the API that you want to view in the API list.
3. In the right-side navigation pane, click **Properties**. Then, you can view the tags of the API in the **Label** column.

   > **Note** You can also create, add, and remove tags for the API in the Properties pane.

   If an API is published, you can also perform the following steps to view the tags of the API:

   i. In DataService Studio, click **Service Management** in the top navigation bar.

   ii. Click **Manage APIs** in the left-side navigation pane. On the page that appears, click the **APIs of Published** tab and click the API whose tags you want to view. The **API Details** page appears.

   iii. View the tags of the API under **Label** in the **API Basic Information** section.

Remove a tag from an API

1. Log on to the DataWorks console, click the DataWorks icon in the upper-left corner, and then choose **All Products > Data Service**.
2. On the **Service Development** tab, double-click the API that you want to publish in the API list.
3. In the right-side navigation pane, click **Properties**. Then, you can view the tags of the API in the **Label** column.
4. Click x next to the tag to remove.

5. Click **Publish** in the upper-right corner. The tag is removed from the API.

Search for APIs by tags

1. Log on to the DataWorks console, click the DataWorks icon in the upper-left corner, and then choose **All Products > Data Service**.
2. On the **Service Development** tab, click **Service Management** in the top navigation bar.
3. Click **Manage APIs** in the left-side navigation pane. On the page that appears, click the **APIs of Published** tab and view the tags of each API in the **Label** column.

   If an API has multiple tags and some of them are hidden, click **...** to show all the tags.
4. On the **APIs of Published** tab, enter a tag in the **Label** field to search for all APIs associated with the tag.

   > **Note** You can search for APIs based on multiple tags.

15.4. Manage business processes and objects under business processes

15.4.1. Manage business processes

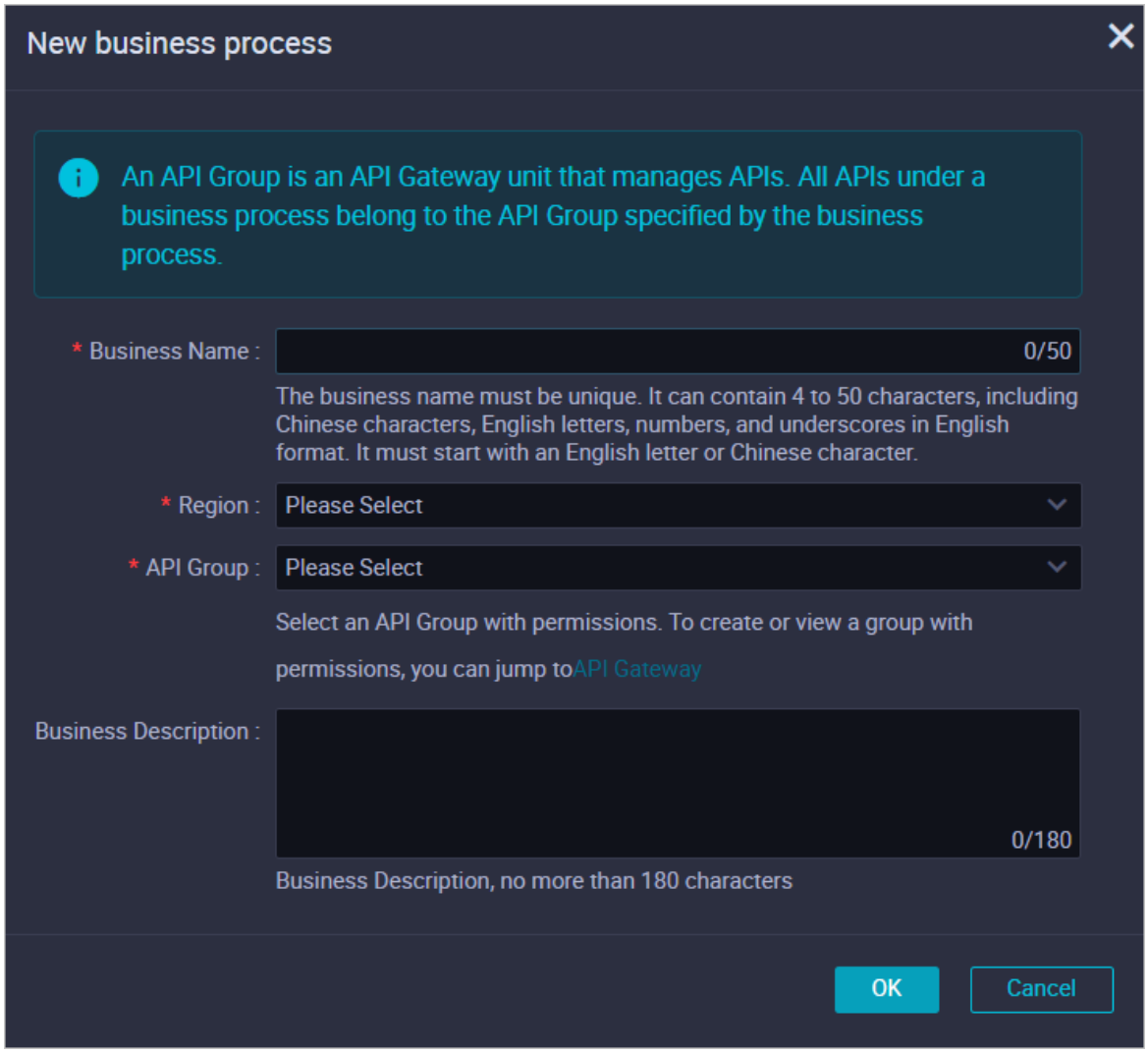
This topic describes how to create, modify, and delete a business process.

Context

DataWorks allows you to organize different types of resources in a business process. This helps you analyze data by business. Each business process contains APIs, functions, and workflows.

Create a business process

1. Log on to the DataWorks console.
2. Click the icon in the upper-left corner and choose **All Products > Data Service**.
3. In the **Service Development** pane, move the pointer over the **+Create** icon and click **Create Business Process**.
4. In the **Create Business Process** dialog box, configure the parameters.



Parameter	Description
Name	The name of the business process. ◦ The name can contain letters, digits, and underscores (_). ◦ The name must start with a letter. ◦ The name must be 4 to 50 characters in length. ◦ The name must be unique in the workspace to which the business process belongs.
Region	The region to which you want to publish APIs. After you select a region, you can publish APIs to this region. You can set this parameter to a region that is associated with the level-1 organization to which your account belongs in the Apsara Uni-manager Management Console.

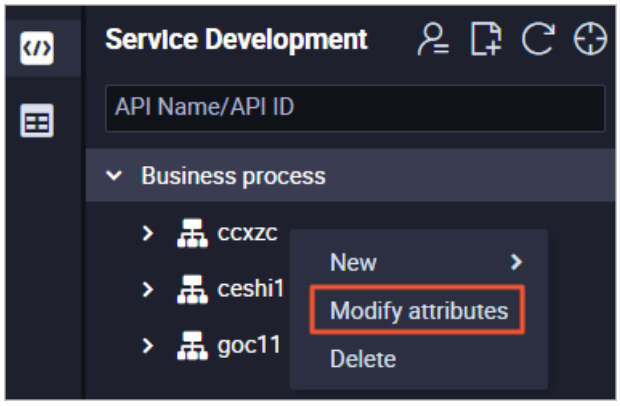
Parameter	Description
API Group	The API group to which the APIs in the business process belong. An API group is the API management unit of API Gateway. You can select an API group from the **API Group** drop-down list. We recommend that you select an API group on which you have permissions in the Apsara Uni-manager Management Console. If you want to view existing API groups or create an API group, go to the API Gateway console.
Description	The description for the business process. The description can be up to 180 characters in length.

5. Click **OK**.

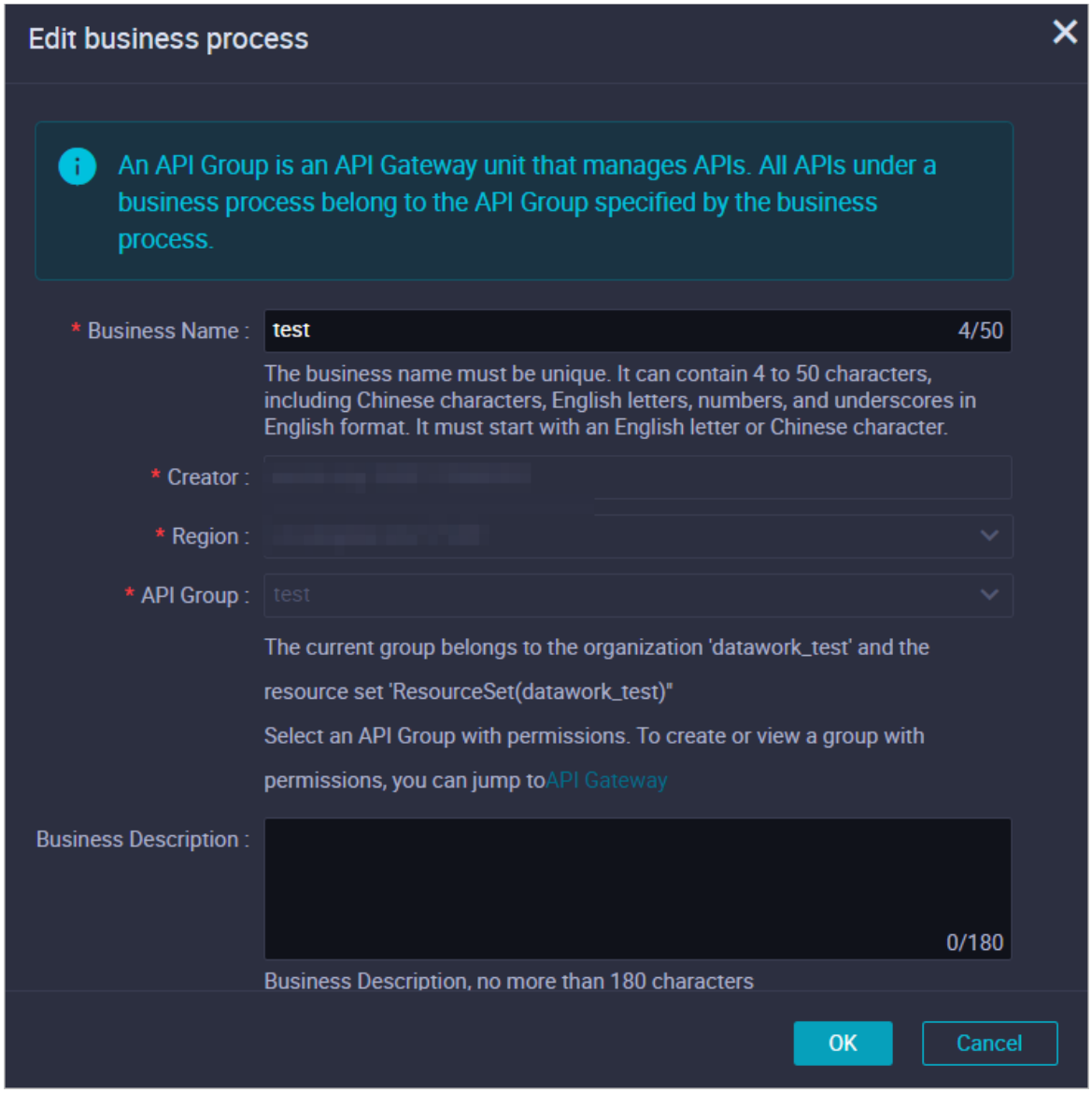
   After the business process is created, you can view the business process in the business process list.

Modify a business process

1. On the **Service Development** page, right-click the name of the business process that you want to modify and select **Modify attributes**.



2. In the **Edit business process** dialog box, modify the **Name** and **Description** parameters as required.

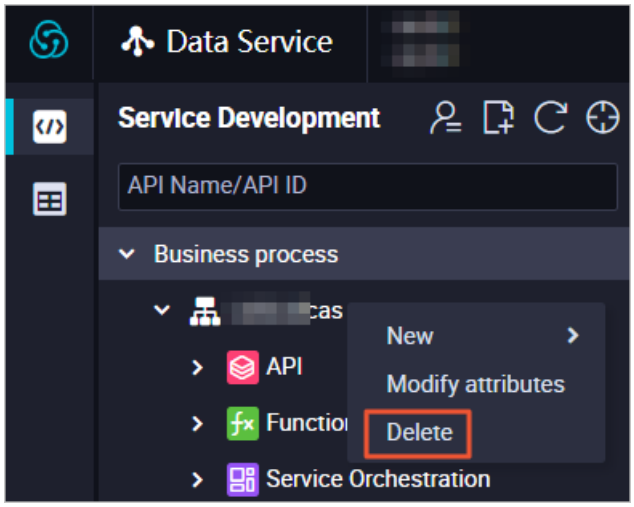


Note You cannot modify the **Creator** or **API Group** parameter of a business process.

3. Click **OK**.

Delete a business process

1. On the **Service Development** page, right-click the name of the business process that you want to delete and select **Delete**.



2. In the **Notes** message, click **OK**.

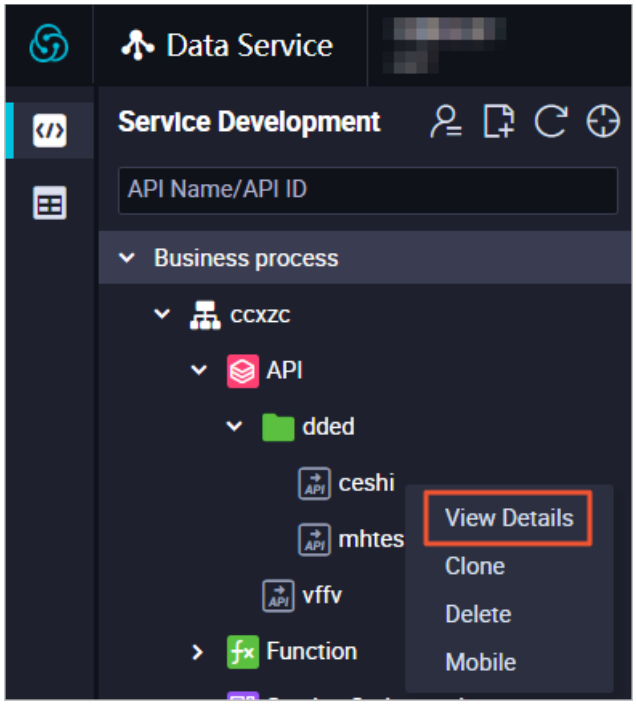
> **Note**
> - You can delete only business processes that do not contain objects such as folders, APIs, functions, or workflows.
> - If you want to delete a business process that contains such objects, delete the objects before you delete the business process.

15.4.2. Manage APIs

This topic describes how to view, clone, delete, and move APIs.

View an API

1. Log on to the DataWorks console.
2. Click ☰ in the upper-left corner and choose **All Products > Data Service**.
3. On the **Service Development** tab, right-click the name of the target API and select **View Details**.

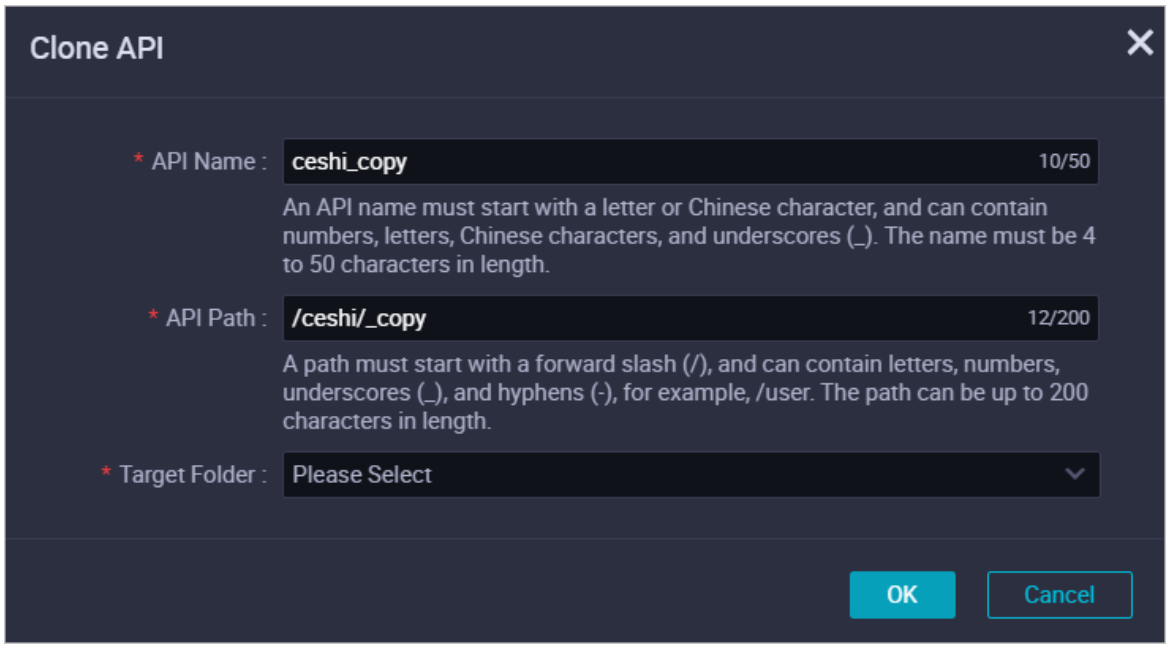


Note The **View Details** option appears only in the shortcut menu of an API that has been published. If an API has not been published, double-click the API to go to the configuration tab of the API. Then, click **Properties** in the right-side navigation pane to view its basic information.

Clone an API

You can clone an API to a specified directory in the directory tree.

1. On the **Service Development** tab, right-click the name of the target API and select **Clone**.
2. In the **Clone API** dialog box, set the parameters as required.



Parameter	Description
API Name	The name of the cloned API. It must be 4 to 50 characters in length and can contain letters, digits, and underscores (_). It must start with a letter.
API Path	The path for storing the cloned API, for example, */user*. The path can contain letters, digits, underscores (_), and hyphens (-). It must start with a forward slash (/) and can be up to 200 characters in length.
Target Folder	The directory for storing the cloned API.

3. Click **OK**.

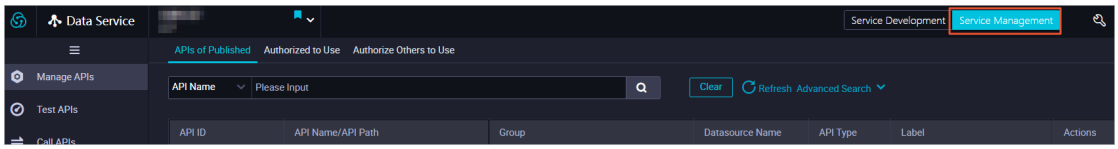
Delete an API

You can delete only APIs that have not been published. To delete APIs that have been published, you must unpublish them first.

1. Optional. Unpublish the target API.

   If the API to be deleted is in the Unpublished state, skip this step.

   i. Go to the **Service Development** tab and click **Service Management** in the upper-right corner.

   ii. On the page that appears, click the **APIs of Published** tab, find the target API, and then click **Unpublish** in the Actions column.

   iii. In the **Unpublish API** message, click **OK**.

   iv. Click **Service Development** in the upper-right corner to return to the **Service Development** tab.

2. On the **Service Development** tab, right-click the name of the target API and select **Delete**.
3. In the **Delete API** message, click **OK**.

> **Note** Deleted APIs cannot be recovered. Use caution when you delete an API.

Move an API to another directory

You can move only APIs that have not been published. To move APIs that have been published, you must unpublish them first.

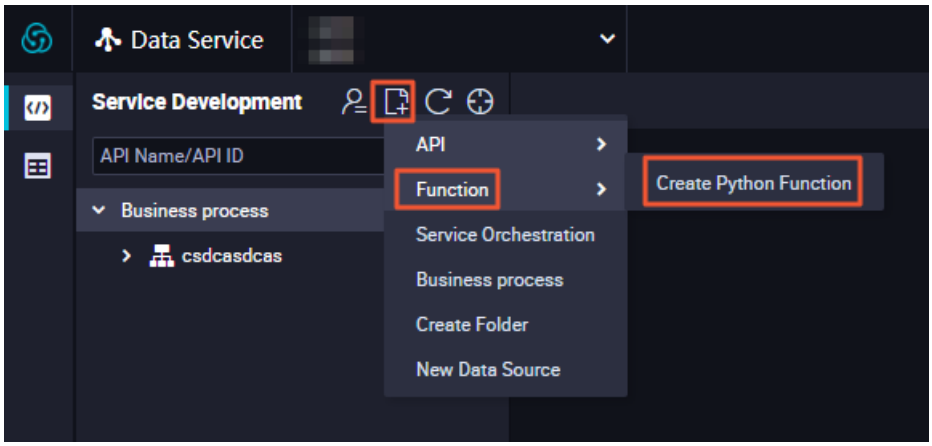
1. On the **Service Development** tab, right-click the name of the target API and select **Mobile**.
2. In the **Modify file path** dialog box, set the **Target Folder** parameter.
3. Click **OK**.

15.4.3. Manage functions

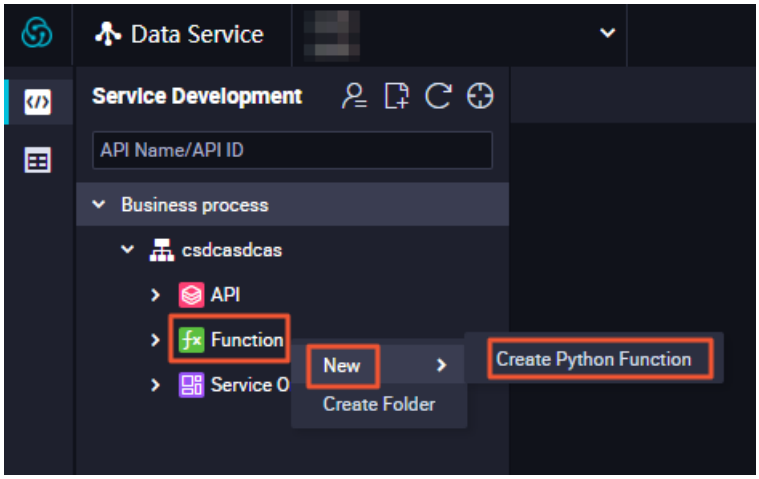
This topic describes how to create, clone, delete, and move Python functions.

Create a Python function

1. Log on to the DataWorks console.
2. Click the icon in the upper-left corner and choose **All Products > Data Service**.
3. Move the pointer over the **+Create** icon and choose **Create Function > Create Python Function**.

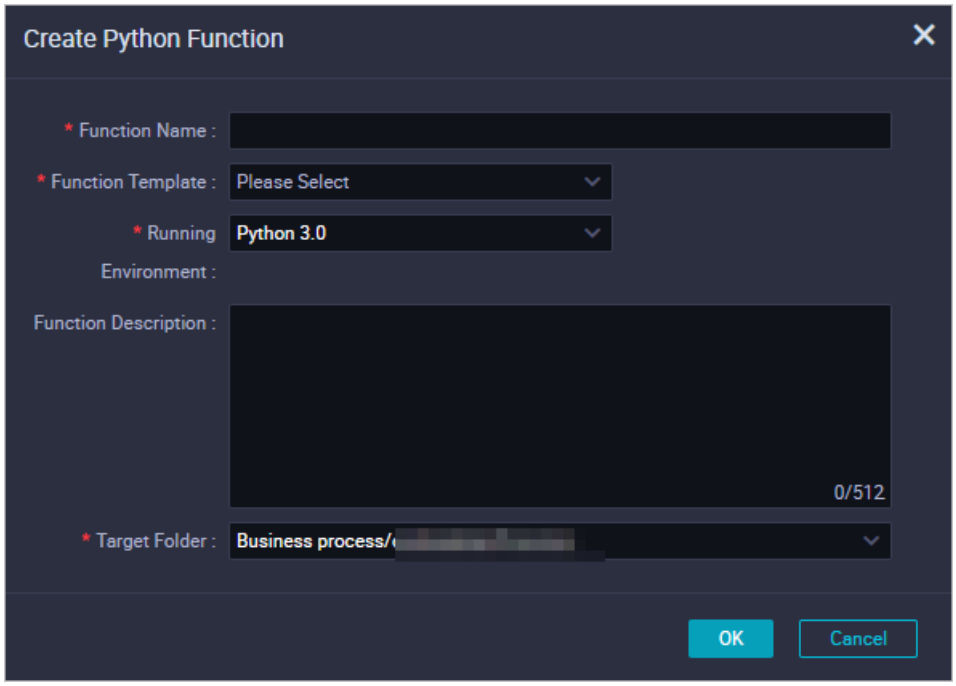


You can alternatively click the required business process, right-click **Function**, and then choose **Create Function > Create Python Function**.



> **Notice** DataService Studio allows you to create only Python functions.

4. In the **Create Python Function** dialog box, set the parameters as required.

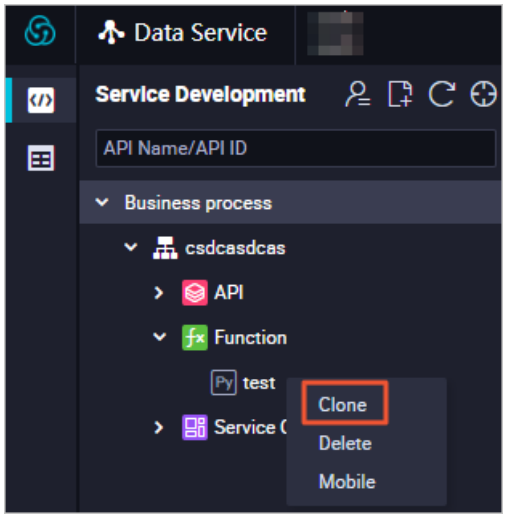


Parameter	Description
Function Name	The name of the function. The name can be up to 256 characters in length.
Function Template	The template that is used to create the function. Set the value to Python3 Standard v1.
Running Environment	The runtime environment of the function. Set the value to Python 3.0.
Function Description	The description of the function. The description can be up to 512 characters in length.
Destination Folder	The folder for storing the function.

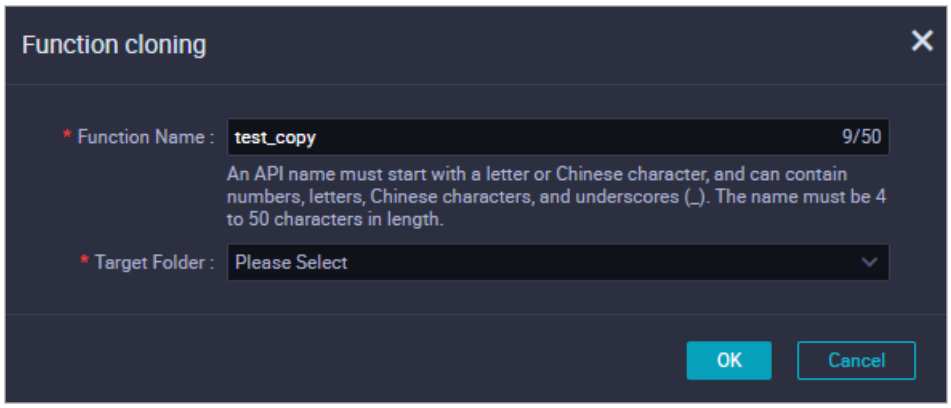
5. Click **OK**.
6. Configure the function on the configuration tab.
   i. In the **Edit Code** section, enter the function code.
   ii. In the **Environment Configuration** section, set the **Memory** and **Function Timeout** parameters.
7. Click the icon in the top toolbar.

Clone a function

1. On the **Service Development** tab, right-click the name of the function that you want to clone and select **Clone**.



2. In the **Clone Function** dialog box, set the **Function Name** and **Destination Folder** parameters.

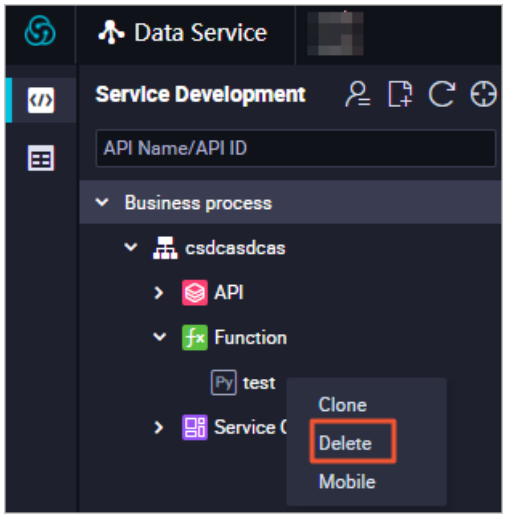


> **Note** The name of the function must be 4 to 50 characters in length and can contain letters, digits, and underscores (_). The name must start with a letter.

3. Click **OK**.

Delete a function

1. On the **Service Development** tab, right-click the name of the function that you want to delete and select **Delete**.

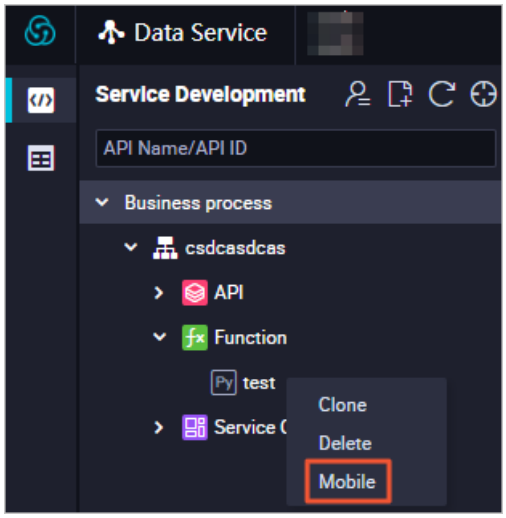


2. In the **Delete Function** message, click **OK**.

> **Note** You can delete only functions that are not referenced by APIs. You must remove the function from the filters of the APIs that reference the function before you can delete the function.

Move a function to another folder

1. On the **Service Development** tab, right-click the name of the function that you want to move and select **Mobile**.



2. In the **Modify file path** dialog box, set the **Destination Folder** parameter.
3. Click **OK**.

15.4.4. Manage workflows

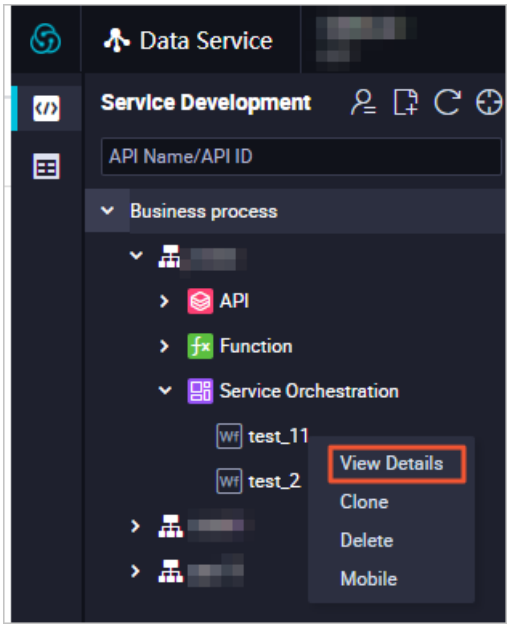
This topic describes how to view, clone, delete, and move a workflow.

Prerequisites

Workflows are created and published. For more information, see Use workflows.

View a workflow

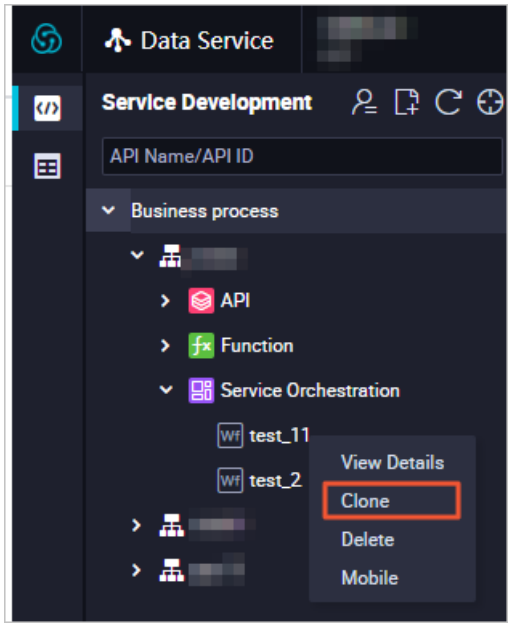
1. Log on to the DataWorks console.
2. Click the icon in the upper-left corner and choose **All Products > Data Service**.
3. On the Service Development tab, right-click the name of the workflow that you want to view and select **View Details**.



> **Note** The View Details option appears only in the shortcut menu of a workflow that has been published. If a workflow has not been published, double-click the workflow to go to the configuration tab of the workflow. Then, click **Properties** in the right-side navigation pane to view the basic information of the workflow.

Clone a workflow

1. On the **Service Development** tab, right-click the name of the workflow that you want to clone and select **Clone**.



2. In the **Clone API** dialog box, configure the parameters.

Parameter	Description
API Name	The name of the cloned API. The name must be 4 to 50 characters in length and can contain letters, digits, and underscores (_). The name must start with a letter.
API Path	The path in which the API is stored. Example: */user*. The path can be up to 200 characters in length and can contain letters, digits, underscores (_), and hyphens (-). The path must start with a forward slash (/).
Destination Folder	The folder that stores the API.

3. Click **OK**.

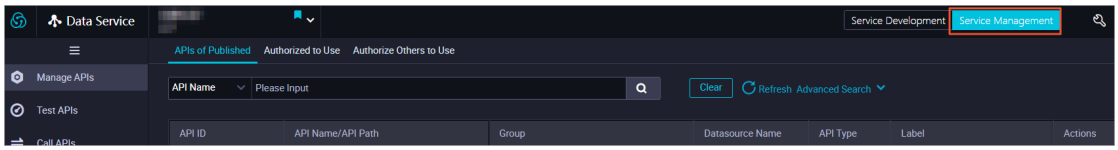
Delete a workflow

You can delete only workflows that have not been published. If you want to delete workflows that have been published, you must unpublish the workflows first.

1. Unpublish a workflow.

   If the workflow to be deleted is in the Unpublished state, skip this step.

   i. Go to the **Service Development** tab and click **Service Management** in the upper-right corner.



ii. On the **APIs of Published** tab, find the API that you want to unpublish and click **Unpublish** in the Actions column.

iii. In the **Unpublish API** message, click **OK**.

iv. Click **Service Development** in the upper-right corner to return to the **Service Development** tab.

2. On the **Service Development** tab, right-click the name of the workflow that you want to delete and select **Delete**.
3. In the **Delete API** message, click **OK**.

> **Note** Deleted APIs cannot be recovered. Exercise caution when you delete an API.

Move a workflow to another folder

You can move only workflows that have not been published. To move workflows that have been published, you must unpublish the workflows first.

1. On the **Service Development** tab, right-click the name of the workflow that you want to move and select **Mobile**.
2. In the **Modify file path** dialog box, set the **Destination Folder** parameter.
3. Click **OK**.

15.5. Create an API

In Data Service, you can quickly create APIs based on tables in relational databases or NoSQL databases using a visual wizard. It takes only a few minutes to configure a data API, and coding is not required.

You can also create APIs by specifying SQL scripts. The script mode supports advanced functions such as associative tables, complex criteria, and aggregate functions.

The differences between the wizard mode and script mode are described as follows:

Differences between the wizard mode and script mode

Category	Description	Wizard mode	Script mode
Query object	Queries a single table from one data source	Supported	Supported
	Queries associative tables from one data source	Not supported	Supported
Search condition	Searches for an exact number	Supported	Supported
	Searches for a range of numbers	Not supported	Supported
	Matches an exact string	Supported	Supported

Category	Description	Wizard mode	Script mode
	Performs fuzzy search for strings	Supported	Supported
	Sets required and optional parameters	Supported	Supported
Query result	Returns the field value	Supported	Supported
	Performs a mathematical calculation for field values	Not supported	Supported
	Performs an aggregate operation on field values	Not supported	Supported
	Displays results with pagination	Supported	Supported

15.5.1. Configure connections

DataService Studio allows you to obtain table schemas and query data through APIs from connections.

> **Note** Before generating an API, make sure that you have configured the relevant connections.

You can click **Connections** on the **Data Integration** page to configure connections. The following table lists the available connection types and supported configuration modes.

Connection name	Create an API in the codeless UI	Create an API in the code editor
ApsaraDB for RDS	Supported	Supported
DRDS	Supported	Supported
ApsaraDB for MySQL	Supported	Supported
ApsaraDB for PostgreSQL	Supported	Supported
ApsaraDB for SQL Server	Supported	Supported
Oracle	Supported	Supported
AnalyticDB	Supported	Supported
Table Store	Supported	Not supported
MongoDB	Supported	Not supported

15.5.2. Create an API in the codeless UI

DataWorks allows you to create APIs by setting parameters in the codeless UI without the need to write code. This topic describes how to create an API in the codeless UI.

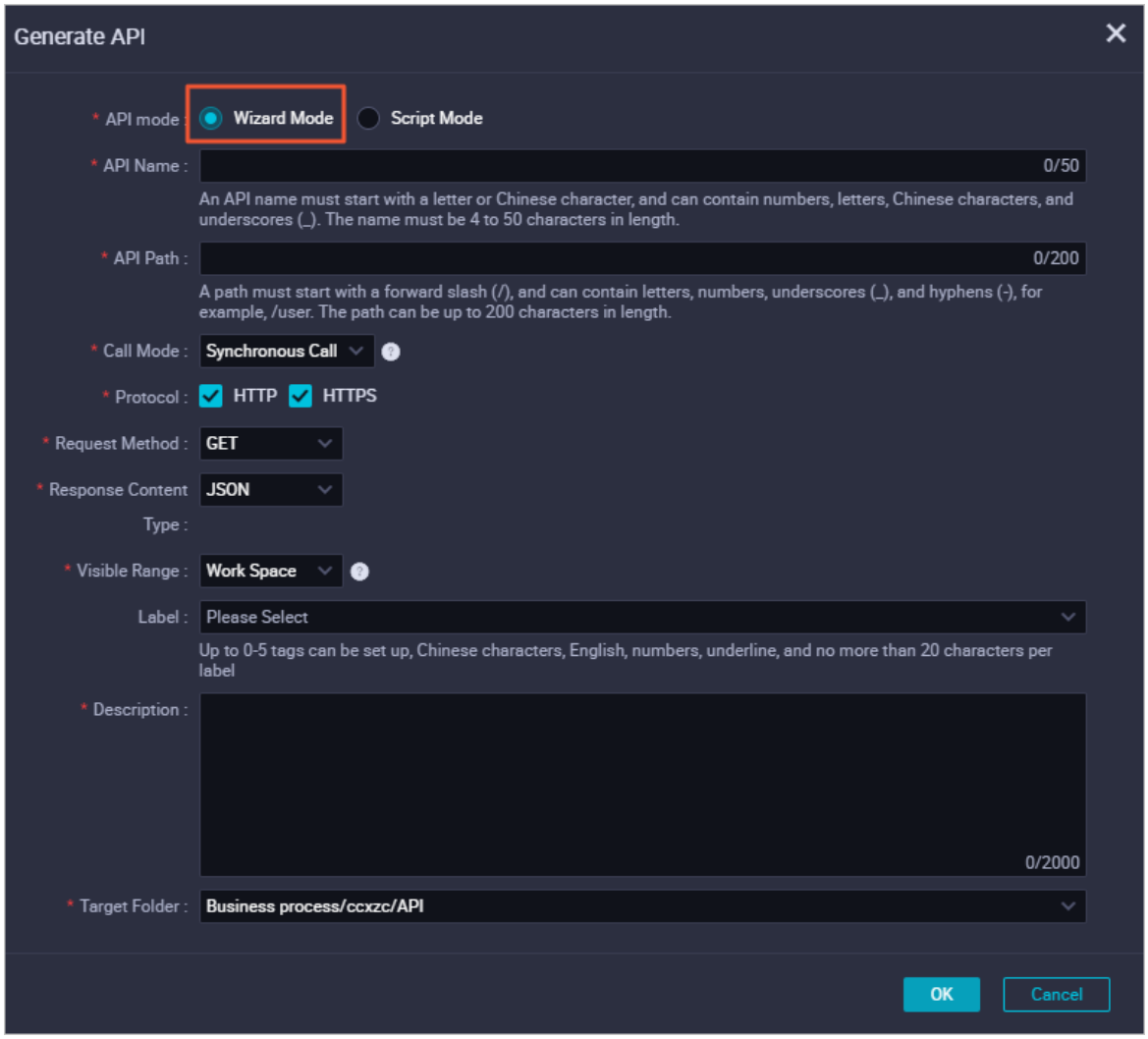
Prerequisites

Connections are configured on the **Data Source** page. For more information, see Configure data sources.

Create an API

1. Log on to the DataWorks console.
2. Click ☰ in the upper-left corner and choose **All Products > Data Service**.
3. On the **Service Development** tab, move the pointer over **+Create** and choose **API > Generate API**.

   You can also click a business process, right-click **API**, and then choose **New > Generate API**.
4. In the **Generate API** dialog box, set the parameters as required.



Parameter	Description
API mode	The mode for creating the API. Valid values: **Wizard Mode** and **Script Mode**. In this example, select **Wizard Mode**.
API Name	The name of the API. The name must be 4 to 50 characters in length and can contain letters, digits, and underscores (_). It must start with a letter.
API Path	The path for storing the API, for example, */user*.
Call Mode	The mode for calling the API. Valid values: **Synchronous Call** and **Asynchronous Call**. ○ If you set this parameter to **Synchronous Mode**, the API returns results immediately after it is called. The synchronous mode is most commonly used. ○ If you set this parameter to **Asynchronous Mode**, the API returns the RequestID parameter immediately after it is called. The API caller can then obtain the call result from a message queue based on the request ID.
Protocol	The protocol used by the API. Valid values: **HTTP** and **HTTPS**.
Request Method	The request method used by the API. Valid values: **GET** and **POST**.
Response Content Type	The return type of the API. Set the value to **JSON**.
Visible Range	The visibility of the API. Valid values: ○ **Work Space**: The API is visible to all members in the current workspace. ○ **Private**: The API is visible only to its owner and permissions on the API cannot be granted to other users. **Note** If you set the Visible Range parameter to Private, the API is visible only to you in the directory tree. It is hidden to other members of the workspace.
Label	The tag of the API. Select one or more tags from the drop-down list. For more information, see Manage tags. **Note** You can set at most five tags for an API.
Description	The description of the API, which can be up to 2,000 characters in length.
Target Folder	The directory for storing the API.

5. Click **OK**.

Configure the API

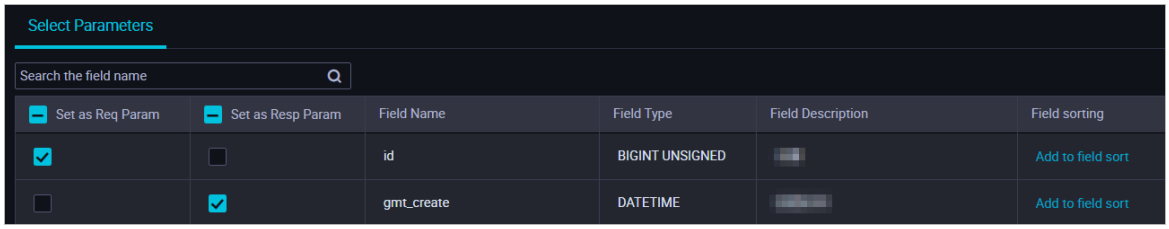
1. Double-click the API in the directory tree. On the configuration tab that appears, set the **Datasource Type**, **Datasource Name**, and **Table Name** parameters in the **Select Table**

section.

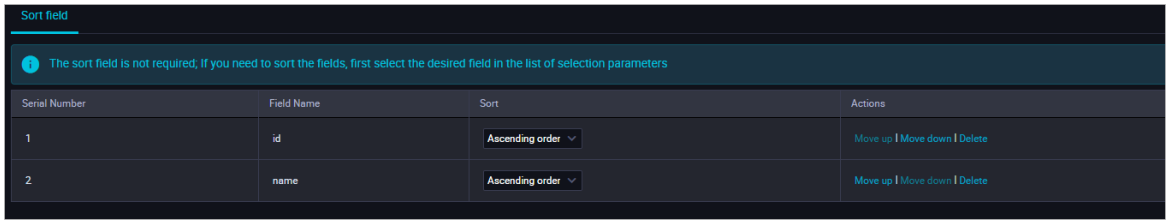
> **Note**
>
> - Before you select a table for an API, you must configure a connection in Data Integration. You can enter a table name in the Table Name field to search for the desired table.
> - After you create an API, the table configuration tab automatically appears for you to select a table for the API.

2. In the **Environment Configuration** section, set the **Memory** and **Function Timeout** parameters.
3. In the **Select Parameters** section, set the request and response parameters for the API.

   After you select a table in the Select Table section, all fields in the table appear in the **Select Parameters** section. Select the required fields and select the check boxes in the **Set as Req Param** and **Set as Resp Param** columns as required.

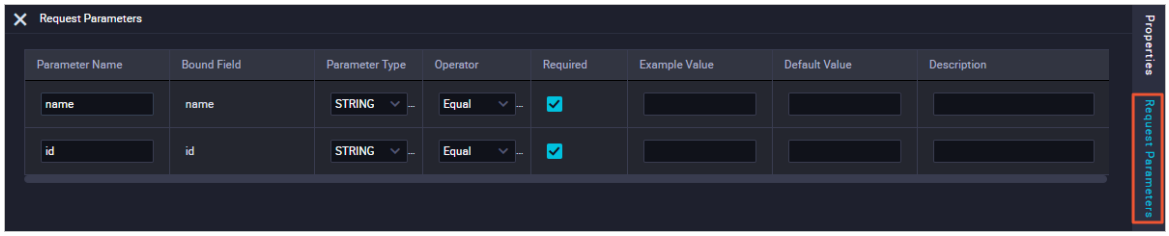


   To sort the data returned by the API based on a field, click **Add to field sort** in the Actions column of the field to add it to the **Sort field** section.



   The sorting feature allows you to specify the fields based on which the parameters returned by the API are sorted. A field with a smaller sequence number in the Sort field section has a higher priority in sorting. You can click **Move up** or **Move down** to adjust the sequence of a field. You can specify the sorting mode for each field by selecting **Ascending order** or **Descending order** in the Sort column.

4. In the right-side navigation pane, click **Request Parameters**. In the Request Parameters pane, set the parameters as required.



Parameter	Description

Parameter	Description
Parameter Name	The name of the request parameter. The name can contain letters, digits, underscores (_), and hyphens (-). It must start with a letter and can be up to 64 characters in length.
Bound Field	The field to be bound to the request parameter. You cannot change the value.
Parameter Type	The type of the request parameter. Valid values: **STRING**, **INT**, **LONG**, **FLOAT**, **DOUBLE**, and **BOOLEAN**.
Operator	The operator that is used to associate or compare the value of the request parameter with the specified value. You can select one of the following operators: ○ **Equal**: The value of the request parameter is equal to the specified value. ○ **LIKE**: The value of the request parameter matches the specified pattern. ○ **IN**: The value of the request parameter is in the specified range. ○ **NOT IN**: The value of the request parameter is out of the specified range. ○ **NOT LIKE**: The value of the request parameter does not match the specified pattern. ○ **! =**: The value of the request parameter is not equal to the specified value. ○ **>**: The value of the request parameter is greater than the specified value. ○ **<**: The value of the request parameter is less than the specified value. ○ **>=**: The value of the request parameter is greater than or equal to the specified value. ○ **<=**: The value of the request parameter is less than or equal to the specified value.
Required	Specifies whether the request parameter is required.
Example Value	The sample value of the request parameter.
Default Value	The default value of the request parameter.
Description	The description of the request parameter.

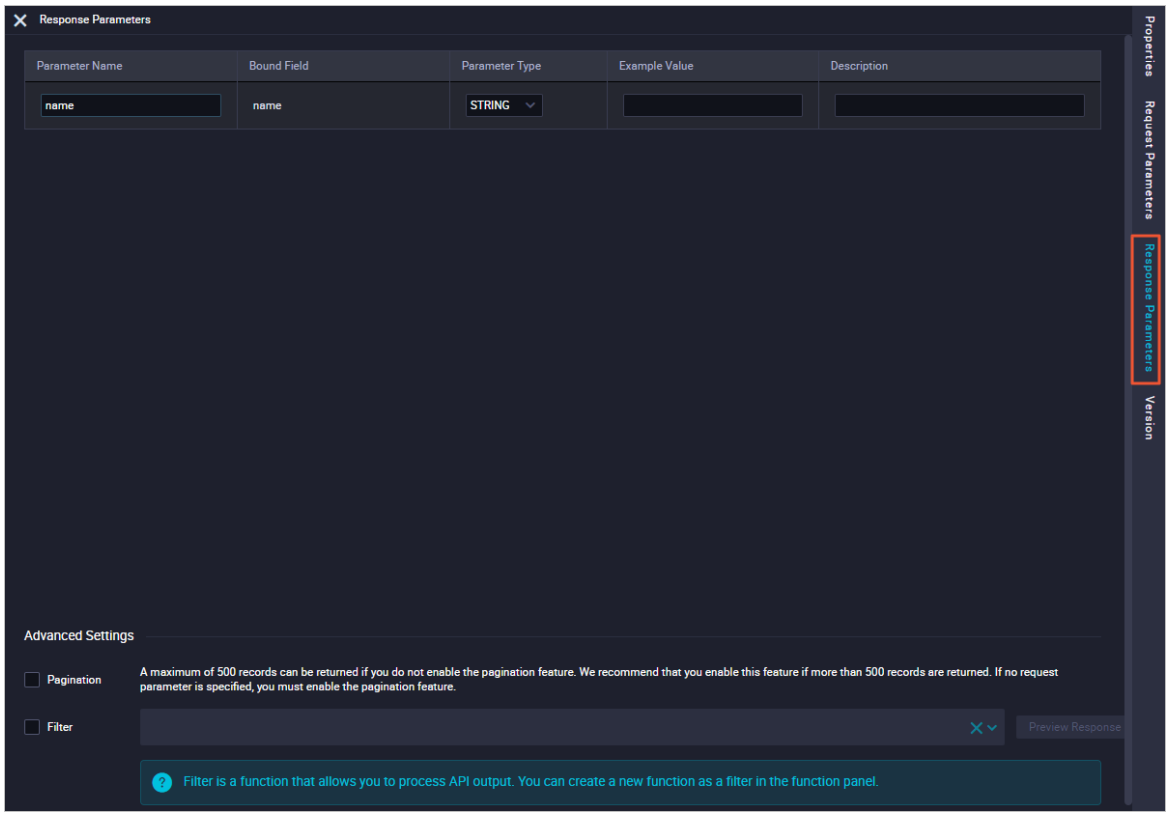
To preprocess the request parameters of the API, select **Use prefilter** in the **Advanced Settings** section. For more information, see Use prefilters.

Note

- To enhance the matching efficiency, set an indexed field as a request parameter.

- To make it easier for API callers to know the details about the API, we recommend that you specify information such as the sample value, default value, and description for each parameter of the API.

5. In the right-side navigation pane, click **Response Parameters**. In the Response Parameters pane, set the parameters as required.



Parameter	Description
Parameter Name	The name of the response parameter. The name can contain letters, digits, underscores (_), and hyphens (-). It must start with a letter and can be up to 64 characters in length.
Bound Field	The field to be bound to the response parameter. You cannot change the value.
Parameter Type	The type of the response parameter. Valid values: **STRING**, **INT**, **LONG**, **FLOAT**, **DOUBLE**, and **BOOLEAN**.
Example Value	The sample value of the response parameter.
Description	The description of the response parameter.

You can select **Pagination** and **Filter** in the **Advanced Settings** section.

Select Pagination based on your needs.

- If you do not select **Pagination**, the API returns a maximum of 2,000 records by default.
- If the API may return more than 2,000 records, we recommend that you select **Pagination**.

The following common parameters are available when Pagination is selected:

- Common request parameters
  - pageNum: the number of the page to return.
  - pageSize: the number of entries to return on each page.
- Common response parameters
  - pageNum: the page number of the returned page.
  - pageSize: the number of entries returned per page.
  - totalNum: the total number of returned entries.

If you need to process the query results returned by the API, select **Filter**.

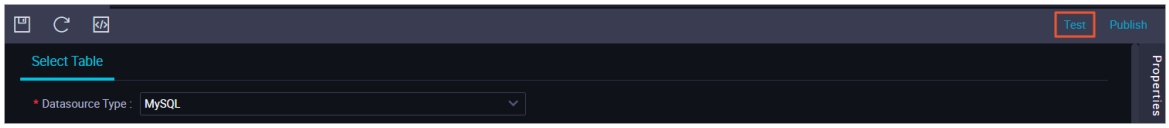
> Note

- Field values are returned in the response as they are in the table.
- Request parameters are optional for an API. If you do not specify any request parameters for an API, you must select **Pagination**.

6. Click Save icon in the toolbar.

Test the API

1. After you save the settings of the API, click **Test** in the upper-right corner.



2. In the **Test APIs** dialog box, click **Test** to send an API request.

   The request and response details appear on the right. If the API fails the test, check the error message, modify the API settings accordingly, and test the API again.

   You can select **Save the correct response example automatically** as required.

> Note

- The system automatically generates sample failure responses and error codes when it tests an API. However, the system does not automatically generate sample success responses.

  To allow the system to save the success test result as a sample success response, you must select **Save the correct response example automatically** before you perform the test. If the response contains sensitive data that must be de-identified, you can manually edit the response.
- The sample success response is an important reference for API callers, and therefore must be configured.
- The Call Latency value is the latency of the current API request, which is used to evaluate the API performance. If the latency is long, consider optimizing the database.

3. After the API is tested, close the **Test APIs** dialog box and click **Publish** in the upper-right corner of the configuration tab.

Switch from the codeless UI to the code editor

On the configuration tab of an API, you can switch from the codeless UI to the code editor.

1. Go to the **Service Development** tab and double-click the target API. The configuration tab of the API appears.
2. Click  in the toolbar.
3. In the message that appears, click **OK**. Then, you can view the SQL statements of the API in the **Edit query SQL** section.

> **Notice**
>
> - DataService Studio allows you to switch only from the codeless UI to the code editor.
> - After you switch from the codeless UI to the code editor, you cannot switch back to the codeless UI.

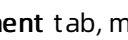
15.5.3. Create an API in the code editor

To meet the requirements of advanced data query, DataService Studio allows you to create an API by writing an SQL script in the code editor. DataService Studio supports table join queries, complex queries, and aggregate functions. This topic describes how to create an API in the code editor.

Prerequisites

Connections are configured on the **Data Source** page. For more information, see Configure data sources.

Create an API

1. Log on to the DataWorks console.
2. Click in the upper-left corner and choose **All Products > Data Service**.
3. On the **Service Development** tab, move the pointer over  and choose **API > Generate API**.

   You can also click a workflow, right-click **API**, and then choose **New > Generate API**.
4. In the **Generate API** dialog box, set the parameters as required.

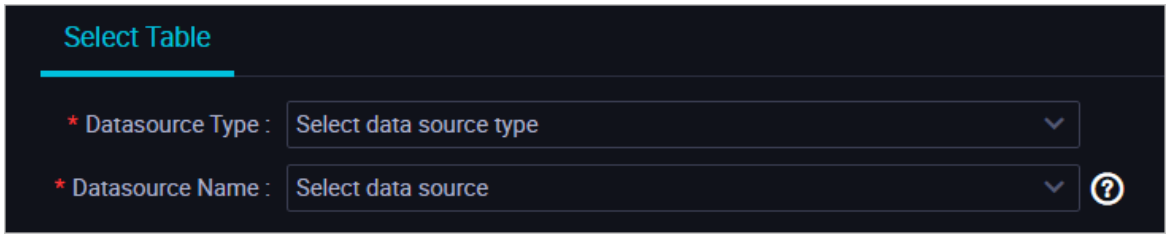
Parameter	Description
API mode	The mode for creating the API. Valid values: **Wizard Mode** and **Script Mode**. In this example, select **Script Mode**.
API Name	The name of the API. The name must be 4 to 50 characters in length and can contain letters, digits, and underscores (_). It must start with a letter.
API Path	The path for storing the API, for example, */user*.

Parameter	Description
Call Mode	The mode for calling the API. Valid values: **Synchronous Call** and **Asynchronous Call**. ○ If you set this parameter to **Synchronous Mode**, the API returns results immediately after it is called. The synchronous mode is most commonly used. ○ If you set this parameter to **Asynchronous Mode**, the API returns the RequestID parameter immediately after it is called. The API caller can then obtain the call result from a message queue based on the request ID.
Protocol	The protocol used by the API. Valid values: **HTTP** and **HTTPS**.
Request Method	The request method used by the API. Valid values: **GET** and **POST**.
Response Content Type	The return type of the API. Set the value to **JSON**.
Visible Range	The visibility of the API. Valid values: ○ **Work Space**: The API is visible to all members in the current workspace. ○ **Private**: The API is visible only to its owner and permissions on the API cannot be granted to other users. **Note** If you set the Visible Range parameter to Private, the API is visible only to you in the directory tree. It is hidden to other members of the workspace.
Label	The tag of the API. Select one or more tags from the drop-down list. For more information, see Manage tags. **Note** You can set at most five tags for an API.
Description	The description of the API, which can be up to 2,000 characters in length.
Target Folder	The directory for storing the API.

5. Click **OK**.

Configure the API

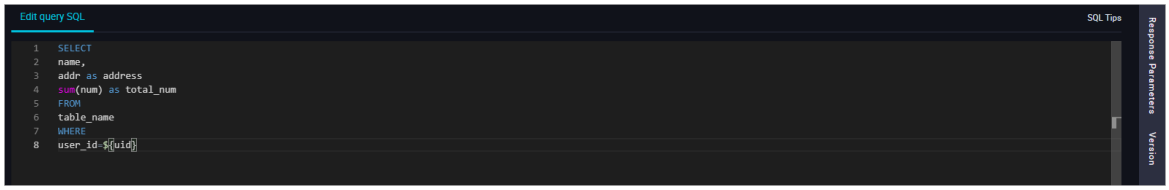
1. Double-click the API. On the configuration tab that appears, set the **Datasource Type** and **Datasource Name** parameters in the **Select Table** section.



> **Note** You must configure a connection in Data Integration in advance. You can enter a keyword in the Table Name field to search for the desired table.

2. In the **Environment Configuration** section, set the **Memory** and **Function Timeout** parameters.

3. In the **Edit query SQL** section, enter the SQL statement for querying data.

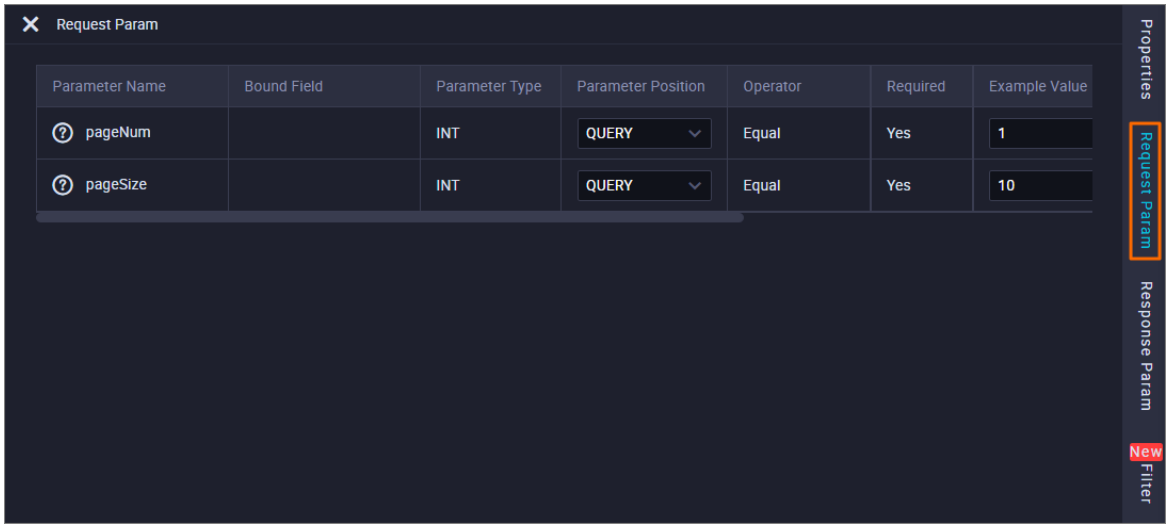


> **Note** The SELECT clause specifies the parameters that the API returns. The WHERE clause specifies the request parameters of the API. You must use ${} to interpolate a request parameter.

Follow these rules when you enter the SQL statement:

- You can enter only one SQL statement in the script editor.
- Only the SELECT clause is supported. Other clauses such as INSERT, UPDATE, and DELETE are not supported.
- The clause `SELECT \*` is not supported. You must specify the columns to be queried.
- Single-table queries, table join queries, and nested queries under the same connection are supported.
- If the name of the column that the SELECT clause specifies has a table prefix, for example, t.name, you must create an alias for the corresponding response parameter, for example, t.name as name.
- If you use an aggregate function, such as min, max, sum, or count, you must create an alias for the corresponding response parameter, for example, `sum(num) as total\_num`.
- ${param} in the SQL statement and that in character strings are regarded as request parameters and replaced. If an escape character \ is placed before ${param}, ${param} is processed as a common string.
- ${param} cannot be enclosed in single quotation marks (' '). For example, `'${id}'` and `'abc${xyz}123'` are not allowed. If necessary, you can use `concat('abc', ${xyz}, '123')` instead.
- Parameters cannot be configured as optional.
- ${param} is not allowed in comments. For example, --${id} is not allowed.

4. In the right-side navigation pane, click **Request Parameters**. In the Request Parameters pane, set the parameters as required.



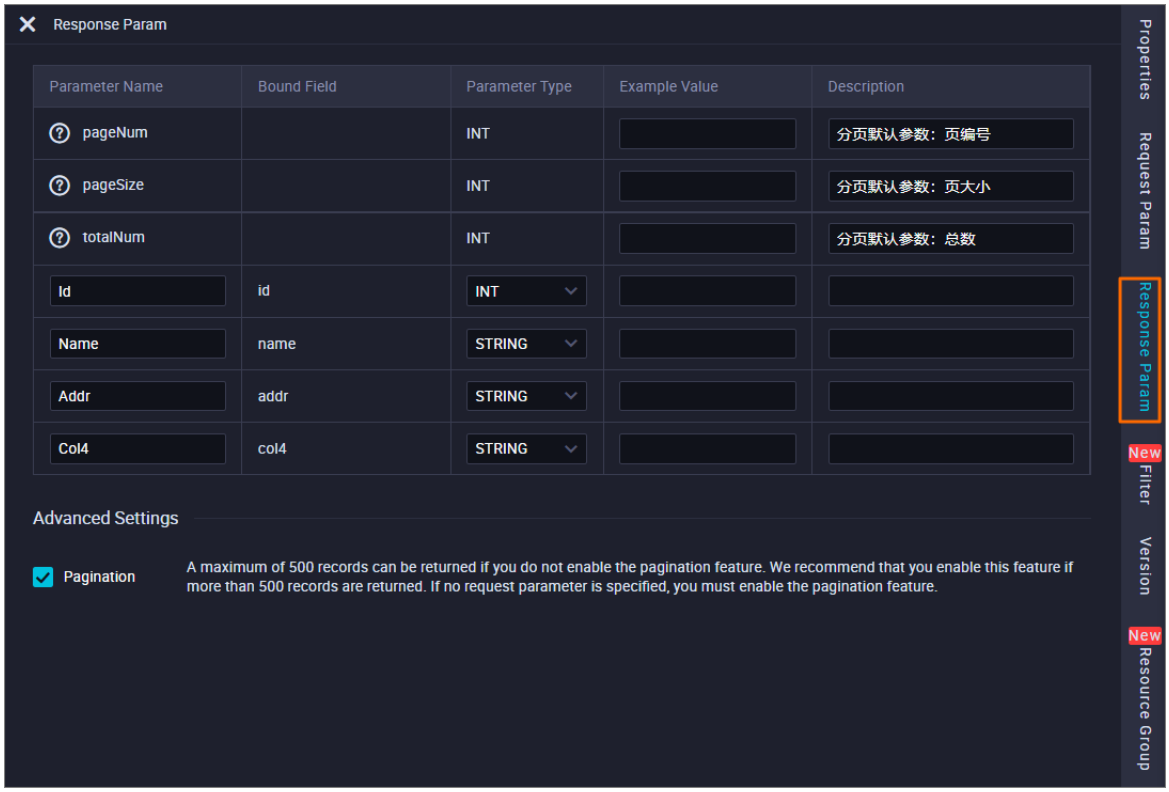
Parameter	Description
Parameter Name	The name of the request parameter. The name can contain letters, digits, underscores (_), and hyphens (-). It must start with a letter and can be up to 64 characters in length.
Parameter Type	The type of the request parameter. Valid values: **STRING**, **INT**, **LONG**, **FLOAT**, **DOUBLE**, and **BOOLEAN**.
Required	Specifies whether the request parameter is required.
Example Value	The sample value of the request parameter.
Default Value	The default value of the request parameter.
Description	The description of the request parameter.

To preprocess the request parameters of the API, select **Use prefilter** in the **Advanced Settings** section. For more information, see Use prefilters.

Note

- To enhance the matching efficiency, set an indexed field as a request parameter.
- To make it easier for API callers to know the details about the API, we recommend that you specify information such as the sample value, default value, and description for each parameter of the API.

5. In the right-side navigation pane, click **Response Parameters**. In the Response Parameters pane, set the parameters as required.



Parameter	Description
Parameter Name	The name of the response parameter. The name can contain letters, digits, underscores (_), and hyphens (-). It must start with a letter and can be up to 64 characters in length.
Parameter Type	The type of the response parameter. Valid values: **STRING**, **INT**, **LONG**, **FLOAT**, **DOUBLE**, and **BOOLEAN**.
Example Value	The sample value of the response parameter.
Description	The description of the response parameter.

You can select **Pagination** and **Filter** in the **Advanced Settings** section.

Select Pagination based on your needs.

- If you do not select **Pagination**, the API returns a maximum of 2,000 records by default.
- If the API may return more than 2,000 records, we recommend that you select **Pagination**.

The following common parameters are available when Pagination is selected:

- Common request parameters
  - pageNum: the number of the page to return.
  - pageSize: the number of entries to return on each page.
- Common response parameters
  - pageNum: the page number of the returned page.
  - pageSize: the number of entries returned per page.

- totalNum: the total number of returned entries.

If you need to process the query results returned by the API, select **Filter**.

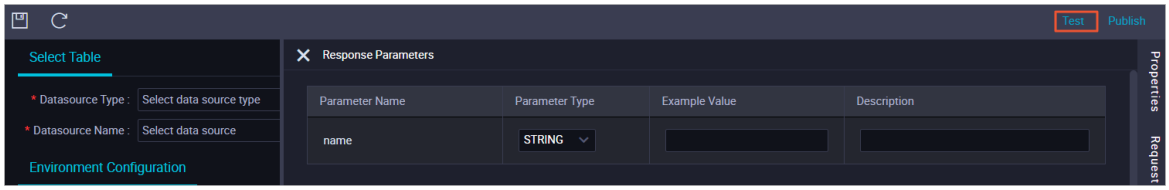
Note

- Field values are returned in the response as they are in the table.
- Request parameters are optional for an API. If you do not specify any request parameters for an API, you must select **Pagination**.

6. Click Save icon in the toolbar.

Test the API

1. After you save the settings of the API, click **Test** in the upper-right corner.



2. In the **Test APIs** dialog box, click **Test** to send an API request.

   The request and response details appear on the right. If the API fails the test, check the error message, modify the API settings accordingly, and test the API again.

   You can select **Save the correct response example automatically** as required.

   Note

   - The system automatically generates sample failure responses and error codes when it tests an API. However, the system does not automatically generate sample success responses.

     To allow the system to save the success test result as a sample success response, you must select **Save the correct response example automatically** before you perform the test. If the response contains sensitive data that must be de-identified, you can manually edit the response.
   - The sample success response is an important reference for API callers, and therefore must be configured.
   - The Call Latency value is the latency of the current API request, which is used to evaluate the API performance. If the latency is long, consider optimizing the database.

3. After the API is tested, close the **Test APIs** dialog box and click **Publish** in the upper-right corner of the configuration tab.

15.5.4. Use filters

15.5.4.1. Use prefilters

A prefilter is a function that is used to process request parameters of APIs. You can specify one or more prefilters to customize the request content for APIs. This topic describes the limits of prefilters, the built-in function template provided by the system, and how to create functions and use them as prefilters.

Context

Prefilters have the following limits:

- Only Python 3.0 functions can be used as prefilters.
- Prefilters support importing only the following modules: json, time, random, pickle, re, and math.
- The function name of a prefilter must be `def handler(event,context):`.

Function template

The system provides the following built-in function template:

```
# -*- coding: utf-8 -*-
# event (str) : in filter it is the API result, in other cases, it is your param
# context : some environment information, temporarily useless
# import module limit: json,time,random,pickle,re,math
# do not modify function name
import json
def handler(event,context):
# load str to json object
obj = json.loads(event) # Convert the string specified by the event parameter to a JSON obj
ect.
# add your code here
# end add
return obj
```

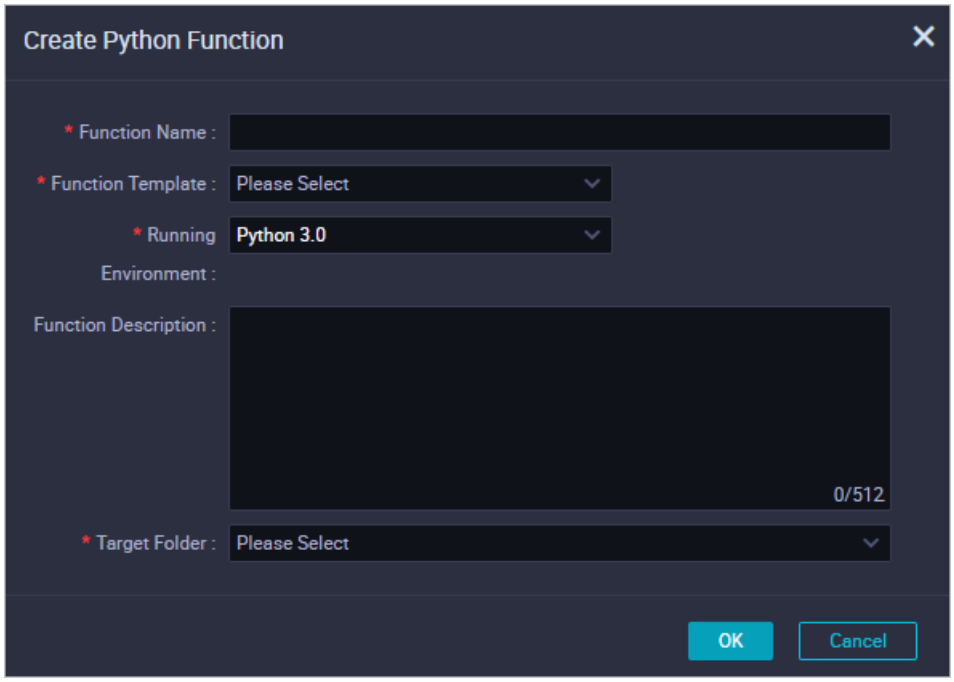
You can modify the function template to write your own function. You can modify the names of the input parameters as needed.

```
Parameter 1 [context]: the context of calling APIs. The value is of the STRING type. This p
arameter is not in use and is left empty.
Parameter 2 [event]: the result data returned by APIs or the preceding filter. The value is
of the STRING type.
```

Create a Python function

1. Log on to the DataWorks console.
2. Click ☰ in the upper-left corner and choose **All Products > Data Service**.
3. On the **Service Development** tab, move the pointer over **+Create** and choose **Function > Create Python Function**.

   You can also click a workflow, right-click **Function**, and then choose **New > Create Python Function**.
4. In the **Create Python Function** dialog box, set the parameters as required.

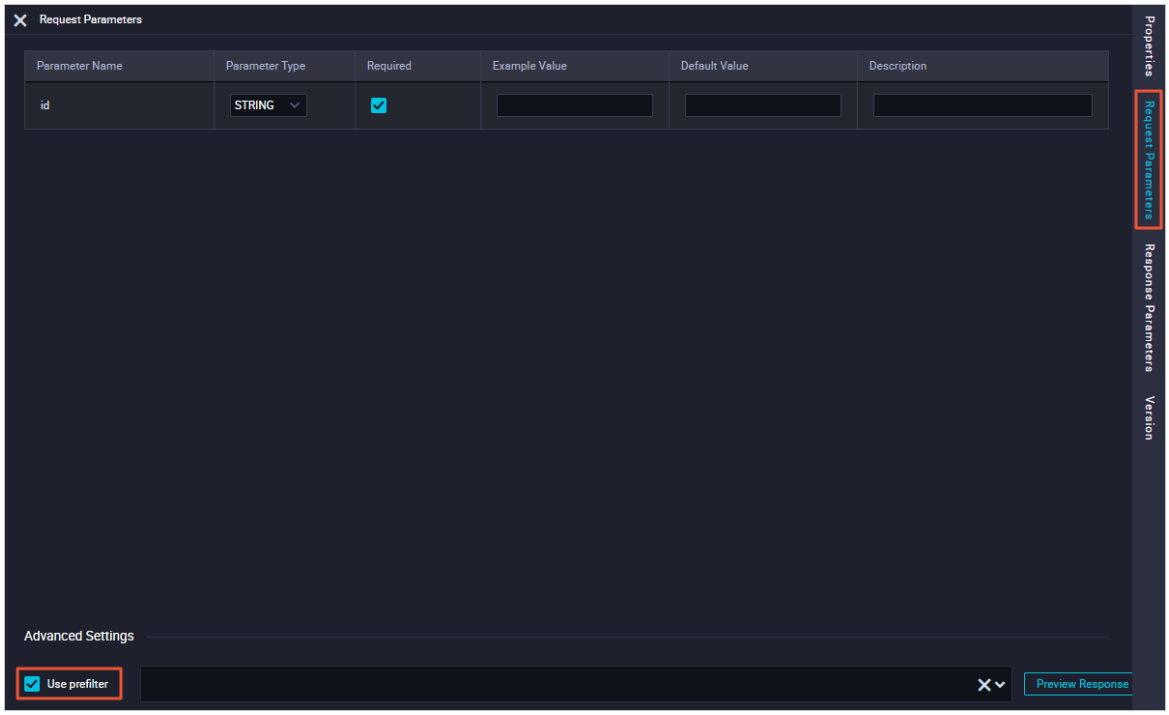


Parameter	Description
Function Name	The name of the function to create, which can be up to 256 characters in length.
Function Template	The template used to create the function. Set the value to Python3 Standard v1.
Running Environment	The runtime environment of the function. Set the value to Python 3.0.
Function Description	The description of the function.
Target Folder	The directory for storing the function.

5. Click **OK**.

Use prefilters

1. On the **Service Development** tab, double-click the target API.
2. On the configuration tab that appears, click **Request Parameters** in the right-side navigation pane.
3. In the **Request Parameters** pane, select **Use prefilter** in the **Advanced Settings** section.



4. Select functions from the **Use prefilter** drop-down list.

   > **Note** A prefilter is a function that is used to process request parameters of APIs. You can create a function and use it as a prefilter.

5. Click **Preview Response** to view the processing results of the prefilters.

15.5.4.2. Use post filters

A post filter is a function that is used to process the results returned by APIs. You can specify one or more post filters to process the results returned by APIs. This topic describes the limits of post filters, the built-in function template provided by the system, and how to create functions and use them as post filters.

Context

Post filters have the following limits:

- Only Python 3.0 functions can be used as post filters.
- Post filters support importing only the following modules: json, time, random, pickle, re, and math.
- The function name of a post filter must be `def handler(event,context):` .

Function template

The system provides the following built-in function template:

```
# -*- coding: utf-8 -*-
# event (str) : in filter it is the API result, in other cases, it is your param
# context : some environment information, temporarily useless
# import module limit: json,time,random,pickle,re,math
# do not modify function name
import json
def handler(event,context):
# load str to json object
obj = json.loads(event) # Convert the string specified by the event parameter to a JSON obj
ect.
# add your code here
# end add
return obj
```

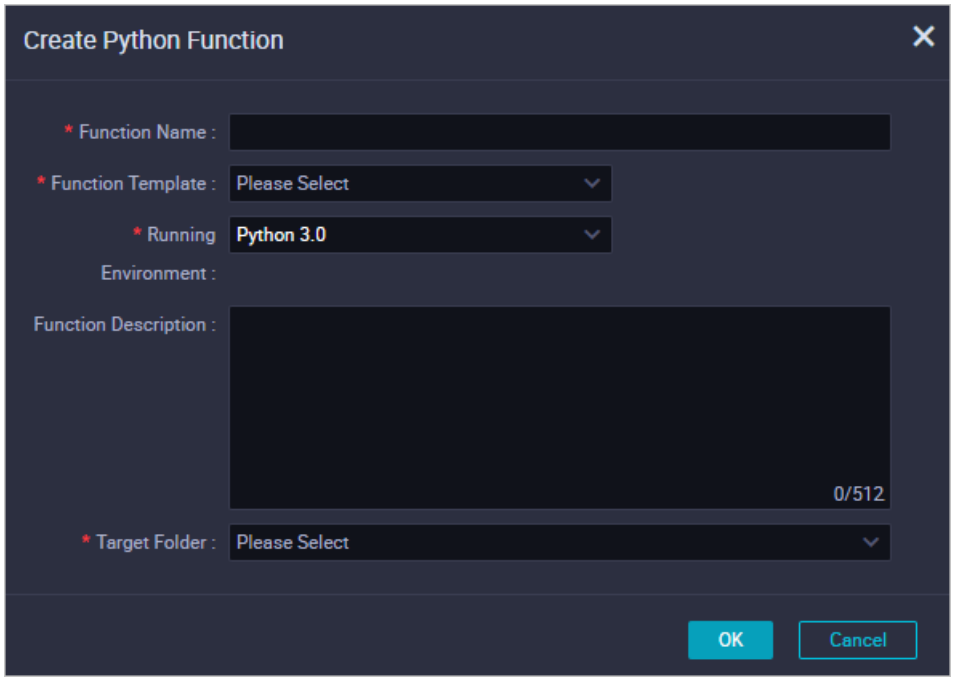
You can modify the function template to write your own function. You can modify the names of the input parameters as needed.

```
Parameter 1 [context]: the context of calling APIs. The value is of the STRING type. This p
arameter is not in use and is left empty.
Parameter 2 [event]: the result data returned by APIs or the preceding filter. The value is
of the STRING type.
```

Create a Python function

1. Log on to the DataWorks console.
2. Click ☰ in the upper-left corner and choose **All Products > Data Service**.
3. On the **Service Development** tab, move the pointer over **+Create** and choose **Function > Create Python Function**.

   You can also click a workflow, right-click **Function**, and then choose **New > Create Python Function**.
4. In the **Create Python Function** dialog box, set the parameters as required.

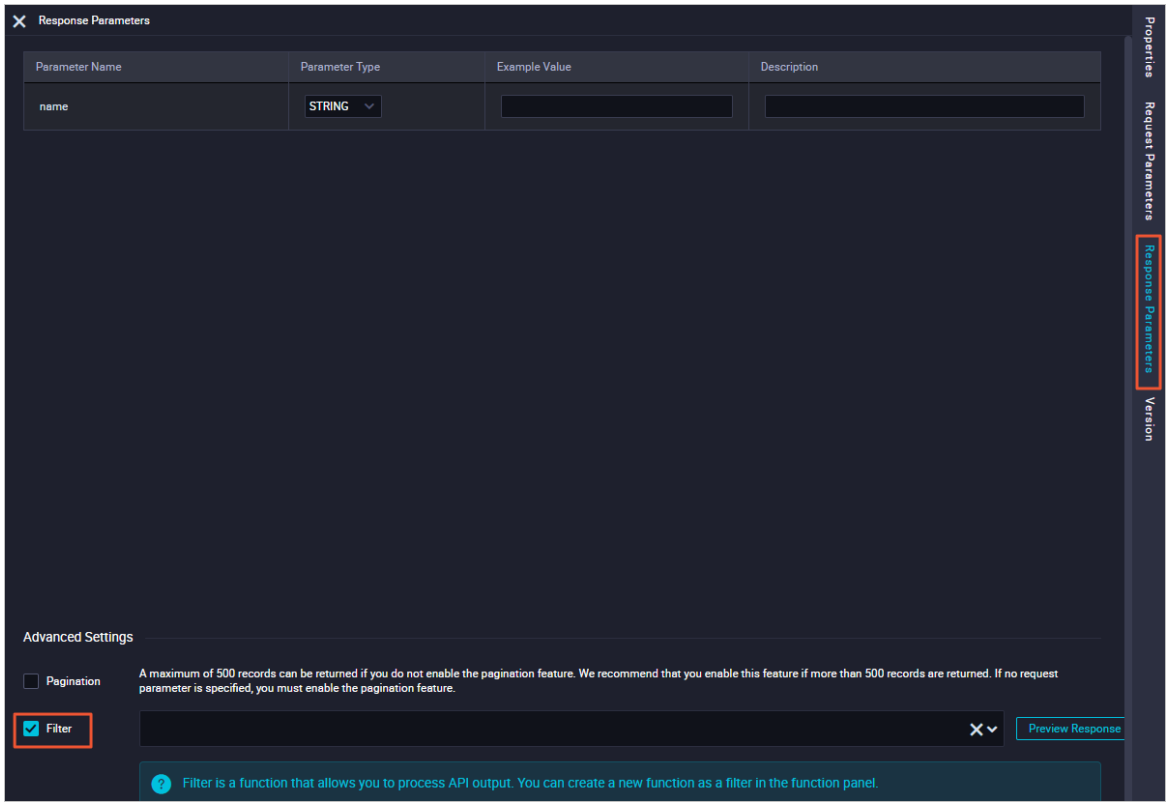


Parameter	Description
Function Name	The name of the function to create, which can be up to 256 characters in length.
Function Template	The template used to create the function. Set the value to Python3 Standard v1.
Running Environment	The runtime environment of the function. Set the value to Python 3.0.
Function Description	The description of the function.
Target Folder	The directory for storing the function.

5. Click **OK**.

Use post filters

1. On the **Service Development** tab, double-click the target API.
2. On the configuration tab that appears, click **Response Parameters** in the right-side navigation pane.
3. In the **Response Parameters** pane, select **Filter** in the **Advanced Settings** section.



4. Select functions from the **Filter** drop-down list.

   > **Note** A post filter is a function that is used to process the results returned by APIs. You can create a function and use it as a post filter.

5. Click **Preview Response** to view the processing results of the post filters.

15.6. Register an API

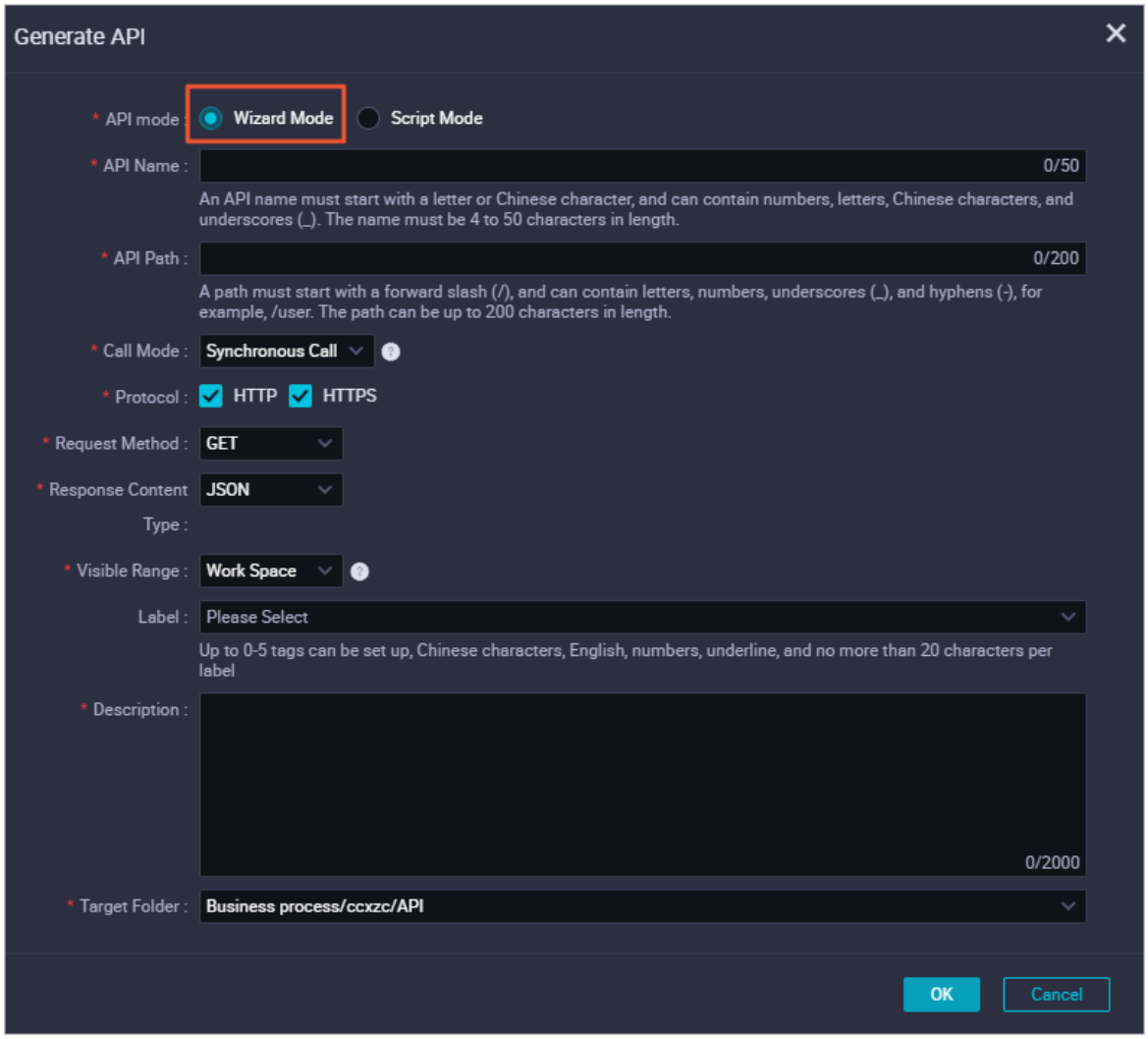
This topic describes how to register APIs, manage APIs, and publish APIs to API Gateway together with APIs created based on data tables.

Four request methods and three data formats are supported. The four request methods are GET, POST, PUT, and DELETE. The three data formats are tables, JSON, and XML.

Register an API

1. Log on to the DataWorks console
2. Click the icon in the upper-left corner and choose **All Products > Data Service**.
3. In the **Service Development** pane, move the pointer over the **+Create** icon and choose **Create API > Register API**.

   You can also click a business process, right-click **API**, and then choose **Create API > Register API**.
4. In the **Register API** dialog box, set the parameters as required.



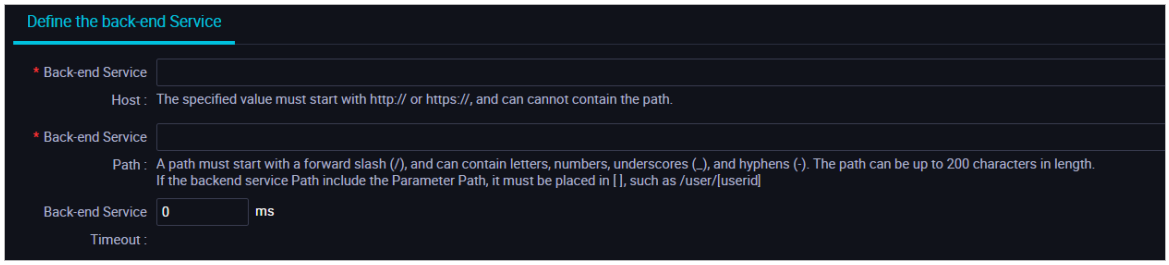
Parameter	Description
API Name	The name of the API. The name must be 4 to 50 characters in length, and can contain letters, digits, and underscores (_). The name must start with a letter.
API Path	The path in which the API is stored. Example: */user*. **Note** The path can be up to 200 characters in length and can contain letters, digits, underscores (_), and hyphens (-). The path must start with a forward slash (/).

Parameter	Description
Call Mode	The mode used to call the API. Valid values: **Synchronous Call** and **Asynchronous Call**. ○ If you set this parameter to **Synchronous Call**, the API returns results immediately after it is called. The synchronous mode is most commonly used. ○ If you set this parameter to **Asynchronous Call**, the API returns the request ID immediately after it is called. The API caller can then obtain the call result from a message queue based on the request ID.
Protocol	The protocol used by the API. Valid values: **HTTP** and **HTTPS**.
Request Method	The request method used by the API. Valid values: **GET**, **POST**, **PUT**, and **DELETE**.
Response Content Type	The format of the data returned by the API. Valid values: **JSON** and **XML**.
Visible Range	The range of users to whom the API is visible. Valid values: ○ **Work Space**: The API is visible to all members in the current workspace. ○ **Private**: The API is visible only to its owner, and permissions on the API cannot be granted to other members. **Note** If you set this parameter to Private, other members in the workspace cannot view the API in the API list.
Label	Select tags from the **Label** drop-down list. For more information, see Manage tags. **Note** You can set a maximum of five tags for an API.
Description	The description for the API. The description can be up to 2,000 characters in length.
Destination Folder	The folder that stores the API.

5. Click **OK**.

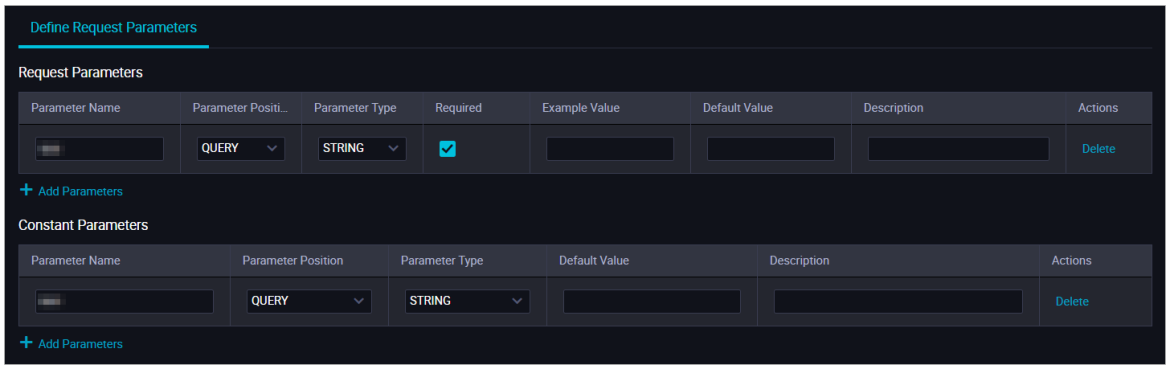
Configure the API

1. Double-click the registered API. On the configuration tab of the API, set the parameters as required in the **Define Backend Service** section.



Parameter	Description
Host	The host of the registered API. The hostname must start with http:// or https://, and cannot contain the path.
Path	The path of the registered API. The path can contain parameters that are enclosed in brackets []. Example: /user/[userid]. In the next step, parameters that are defined in the Path parameter are automatically added to the request parameter list.
Back-end Service Timeout	The timeout period of the backend service.

2. In the **Define Request Parameters** section, set the parameters as required.



Parameter	Description
Request Parameters	You can click **Add Parameter** to add a request parameter for the API. For each request parameter, you can set the **Parameter Position** field to **QUERY**, **HEAD**, or **BODY**. The valid values of the Parameter Position field vary based on the request method that is used by the API. Select one from the valid values that are displayed.
Constant Parameters	Constant parameters have fixed values and are invisible to API callers. The constant parameters do not need to be specified during an API call. However, the backend service receives the defined constant parameters and their values in each API call. The constant parameters are applicable when you need to fix the value of a parameter or hide the parameter from API callers.

Parameter	Description
Request Body Description	This parameter is displayed only if you set the Request Method parameter to **POST** or **PUT**. In the Request Body Description parameter, you can enter the body description in the format of JSON or XML. This way, you can provide an example of the request body for API callers to determine the format of the request body.

3. In the **Define Response Content** section, set the **Correct Response Example** and **Error Response Example** parameters. These examples are references for API callers to write the code for parsing the results of the API.
4. In the **Define Error Codes** section, set the **Error Code**, **Error Message**, and **Solution** parameters to define an error code. The information helps API callers diagnose the causes of the error.
5. Click the icon in the top toolbar.

   After the API is configured, you can test it. For more information, see Test an API.

   After the API passes the test, click **Submission** in the upper-right corner.

   In the right-side navigation pane of the configuration tab of the API, click **Version**. Find the version that you want to publish and click **Publish** in the Actions column.

15.7. Manage APIs

After an API is published, you can perform operations to manage the API. For example, you can authorize access to, unpublish, or test the API. This topic describes the operations that you can perform to manage APIs.

Prerequisites

One or more APIs are published.

You can unpublish, authorize access to, or change the protocols of APIs only after the APIs are published. For more information, see Publish APIs.

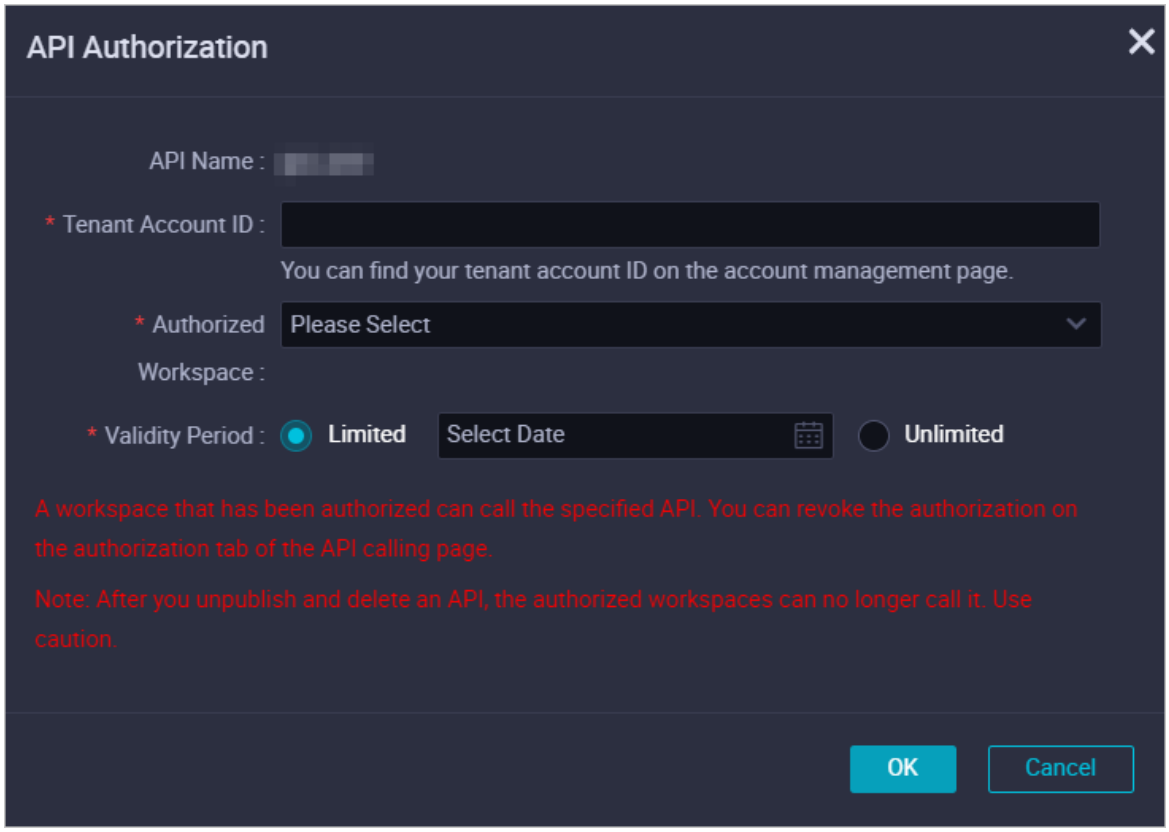
Go to the Manage APIs page

1. Log on to the DataWorks console
2. Click the More icon in the upper-left corner and choose **All Products > Data Service**. The Data Service page appears.
3. Click in the upper-left corner and choose **All Products > Data Service**.
4. Click the **Service Management** tab in the upper-right corner of the Data Service page. The Manage APIs page appears.

   On the APIs of Published tab of the Manage APIs page, you can search for an API, and then authorize access to, unpublish, test, change the protocol of, or view the statistics of the API. On the Authorized to Use tab, you can view the APIs that you are authorized to access. On the Authorize Others to Use tab, you can view the APIs that you authorize others to access.

Authorize access to an API

1. On the **APIs of Published** tab, find the API to which you want to authorize access and click **Authorize** in the Actions column.
2. In the **API Authorization** dialog box, set the parameters based on your business requirements.



Parameter	Description
API Name	The name of the API to which you want to authorize access. You cannot change the value.
Tenant Account ID	The ID of the Alibaba Cloud account to which you want to grant the permissions to call the API. You can go to the **Account Management** page to view the account ID.
Authorized Workspace	Enter the ID of the workspace to which you want to grant the permissions to call the API in the field to search for the workspace, and select the workspace from the drop-down list. **Note** You can view the workspace ID in the Basic properties section on the Workspace Management page.
Validity Period	The validity period of the permissions to call the API. Valid values: **Limited** and **Unlimited**. ○ **Limited**: specifies that the authorized user has the permissions to call the API before the specified expiration date. ○ **Unlimited**: specifies that the authorized user have permanent permissions to call the API.

3. Click **OK**.

Unpublish an API

> **Note**
> - If you unpublish or delete an API after you authorize a workspace to access the API, the workspace can no longer call the API.
> - If you publish an API again after you unpublish or modify the API, you must authorize access to the API again.

1. On the **APIs of Published** tab, find the API that you want to unpublish and click **Unpublish** in the Actions column.
2. In the **Unpublish API** message, click **OK**.

Test an API

On the **APIs of Published** tab, find the API that you want to test and click **Test** in the Actions column. The **Test APIs** page appears. For more information, see Test an API.

Change the protocol of an API

> **Note**
> - If you deselect a protocol, you can no longer call the API by using this protocol. Use caution when you perform this operation.
> - The protocol change takes effect immediately.

1. On the **APIs of Published** tab, find the API of which you want to change the protocol and choose **More** > **Change Protocol** in the Actions column.
2. In the **Change Protocol** dialog box, change the protocol used by the API and click **OK**.

View the statistics of an API

On the **APIs of Published** tab, find the API of which you want to view the statistics and choose **More** > **View Statistics** in the Actions column. The page for viewing the statistics of the API appears.

View the APIs that you are authorized to access

On the **Manage APIs** page, click the **Authorized to Use** tab to view the APIs that you are authorized to access.

You can perform the following operations on the APIs that you are authorized to access:

- Find the API that you want to test and click **Test** in the Actions column to test the API on the **Test APIs** page.
- Find the API on which you want to remove access permissions and click **Delete**. In the **Delete authorized** message, click **OK** to remove access permissions on the API.

> **Note** If you remove access permissions on an API, the workspace to which you belong can no longer call the API.

View the APIs that you authorize others to access

On the **Manage APIs** page, click the **Authorize Others to Use** tab to view the APIs that you authorize others to access.

You can perform the following operations on the APIs that you authorize others to access:

- Find the API that you want to test and click **Test** in the Actions column to test the API on the **Test APIs** page.
- Find the API on which you want to manage access permissions and click **Manage** in the Actions column. In the **Authorization** dialog box, revoke or modify the permissions of a workspace on the API.

15.8. View API statistics

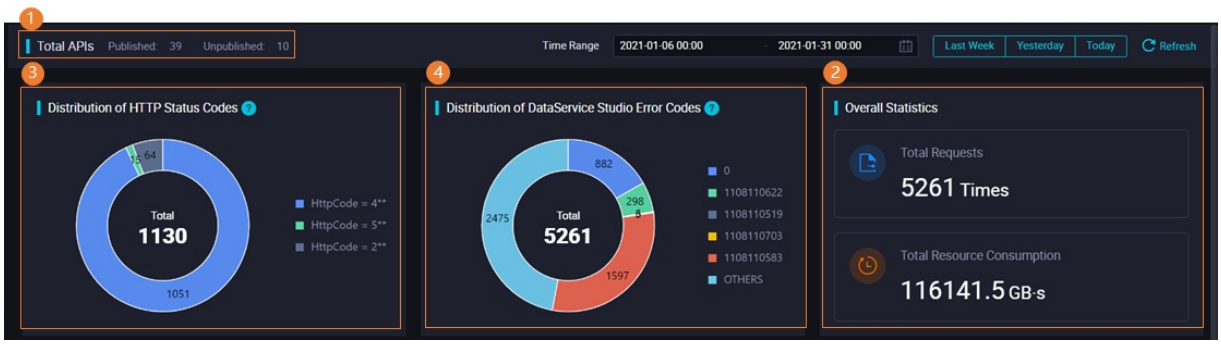
15.8.1. View the summary information about API statistics

DataWorks uses dashboards to provide various visual charts and statistics of APIs. You can view the total number of APIs in a workspace, total number of API calls, total resource consumption, and information such as HTTP status codes, data service error codes, resource allocation, and rankings. This helps you obtain information about API calls from a global perspective.

Go to the Statistics Dashboard page

1. Go to the **Data Service** page.
2. Click the **Service Management** tab in the upper-right corner of the Data Service page. The **Manage APIs** page appears.
3. In the left-side navigation pane, choose **API Statistics > Statistics Dashboard**. The Statistics Dashboard page appears.

View overall statistics



- Total APIs section

  In this section, you can view the number of published APIs and the number of unpublished APIs as of the current time in the current workspace.

- Overall Statistics section

  In this section, you can view the statistics of all the APIs in the current workspace in the specified period of time, including the total number of API calls and total resource consumption.

The maximum statistical period for overall statistics is seven days.

- Distribution of HTTP Status Codes section

  In this section, you can view the distribution of HTTP status codes that are returned by all published APIs in the current workspace after the APIs are called in the specified period of time. Different sectors in the pie chart represent different HTTP status codes. You can click a sector in the pie chart to view the corresponding status code. The maximum statistical period for HTTP status codes is seven days.

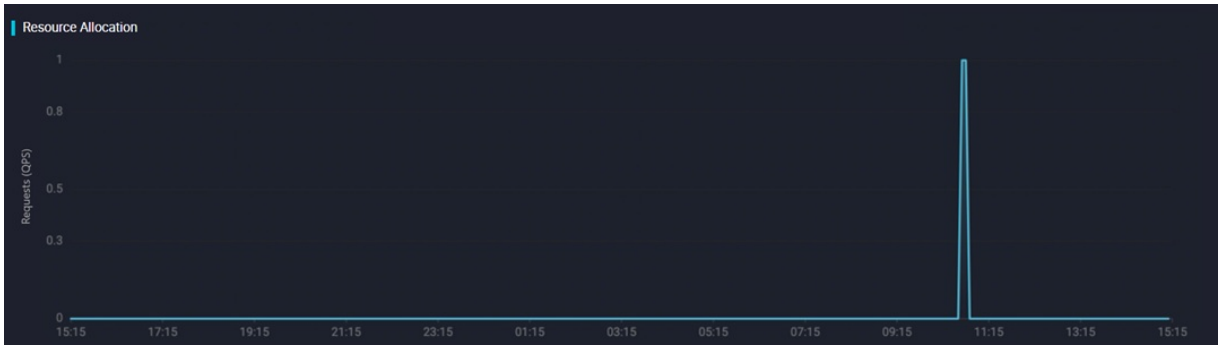
- Distribution of DataService Studio Error Codes section

  In this section, you can view the distribution of data service error codes that are returned by all published APIs in the current workspace after the APIs are called in the specified period of time. Different sectors in the pie chart represent different data service error codes. You can click a sector in the pie chart to view the corresponding error code. For more information about data service error codes, see .

  The maximum statistical period for data service error codes is seven days.

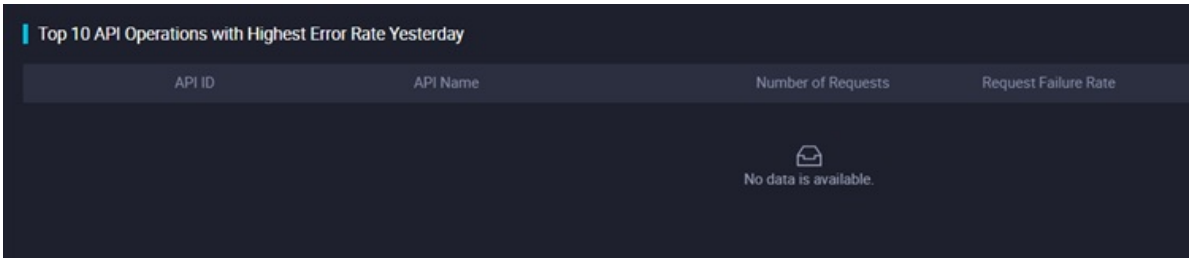
View resource allocation

The chart in the Resource Allocation section displays the trend of the number of API calls that are requested in the past 24 hours.



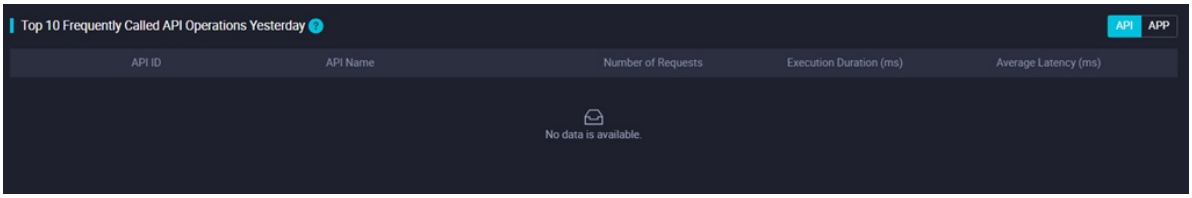
View rankings

- Top 10 API Operations with Highest Error Rate Yesterday section



  In this section, you can view the top 10 APIs with the highest call error rate in the past 24 hours. You can view the ID, name, number of calls, and call error rate of each API.

- Top 10 Frequently Called API Operations Yesterday section



In this section, you can view the top 10 APIs that are most frequently called or the top 10 applications that most frequently call APIs in the past 24 hours. You can click API or APP in the upper-right corner of the section to view rankings in different dimensions. In the API dimension, you can view the following information: API ID, API Name, Number of Requests, Execution Duration (ms), and Average Latency (ms). In the APP dimension, you can view the following information: APP ID, Application Name, Workspace, Number of Requests, Execution Duration (ms), and Average Latency (ms).

15.8.2. View the details of API statistics

DataWorks provides various visual charts and statistics for each API. You can view monitoring charts of an API to obtain statistics details of the API, including HTTP status codes, data service error codes, the number of requests from applications, traffic bandwidth, and average response time.

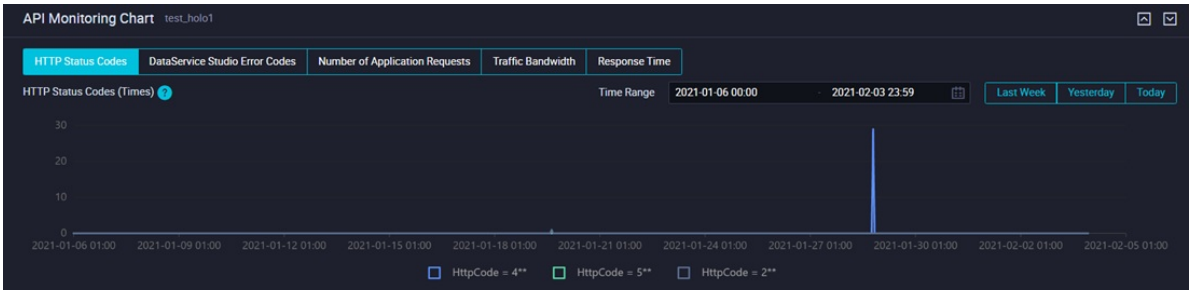
Go to the Statistics Details page

1. Go to the **Data Service** page.
2. Click the **Service Management** tab in the upper-right corner of the Data Service page. The **Manage APIs** page appears.
3. In the left-side navigation pane, choose **API Statistics > Statistics Details**. The Statistics Details page appears.

Find the API of which you want to view the statistics details and click **Monitoring Chart** in the Actions column. Then, you can view monitoring charts of the API.

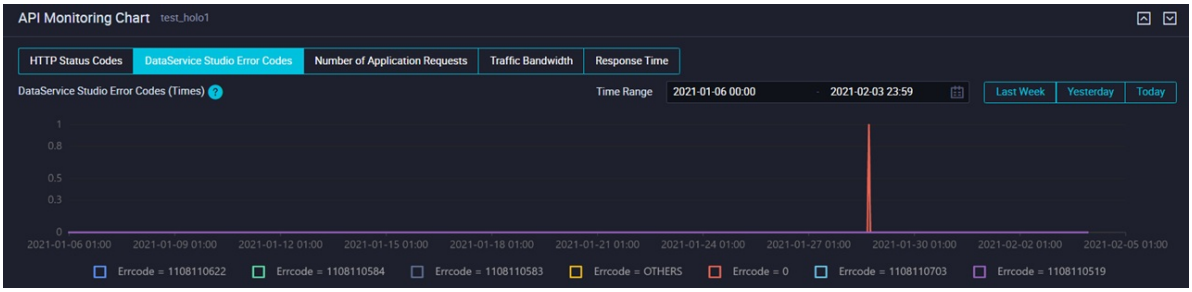
View monitoring charts

- HTTP Status Codes chart



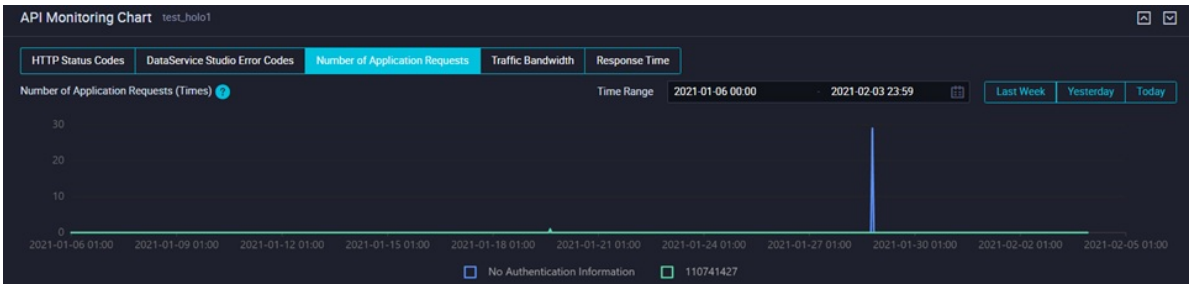
The HTTP Status Codes chart displays the trends of the numbers of different HTTP status codes that are returned after the API is called in the specified period of time.

- DataService Studio Error Codes chart



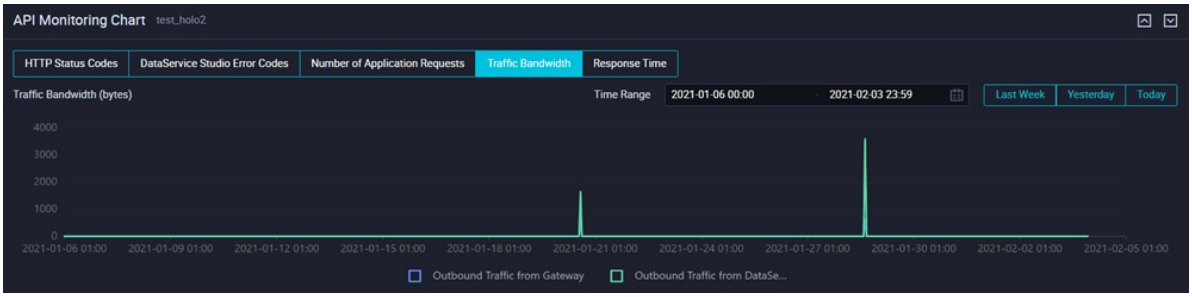
The DataService Studio Error Codes chart displays the trends of the numbers of different data service error codes that are returned after the API is called in the specified period of time. This helps you troubleshoot errors in an efficient manner.

- Number of Application Requests chart



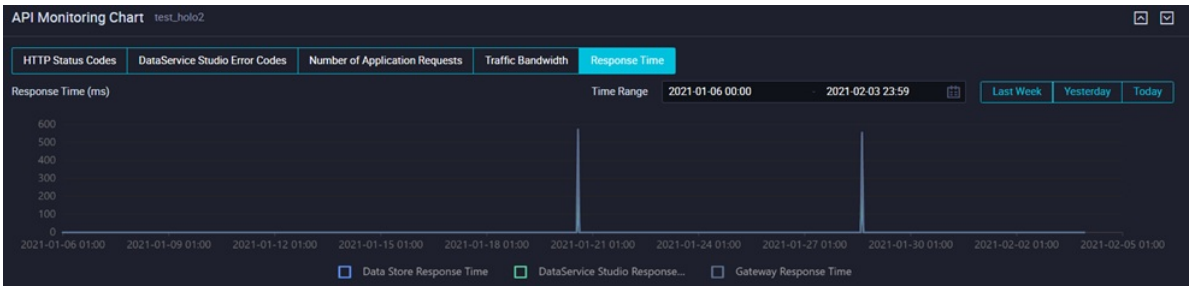
The Number of Application Requests chart displays the trends of the numbers of API calls from different applications in the specified period of time.

- Traffic Bandwidth chart



The Traffic Bandwidth chart displays the trends of the outbound traffic from DataService Studio and outbound traffic from API Gateway for API calls in the specified period of time.

- Average Response Time chart



The Average Response Time chart displays the trends of the execution durations in API Gateway, DataService Studio, and the data source after API calls are requested in the specified period of time.

15.9. Test an API

This topic describes how to test an API.

DataWorks allows you to test an API when you create or register the API. DataWorks also provides an independent API test feature, which allows you to test an API online.

Test an API that is being developed

You can test an API that is being developed on the **Service Development** tab. Before you perform such a test, create or register an API. For more information, see Create an API in the codeless UI and Register APIs.

1. Log on to the DataWorks console
2. Click the icon in the upper-left corner and choose **All Products > Data Service**.

3. On the **Service Development** tab, double-click the API under development that you want to test in the API list.
4. On the configuration tab of the API, click **Test** in the upper-right corner.
5. In the **Test APIs** dialog box, click **Test** to send an API request online.

   The request and response details appear on the right. If the API fails the test, check the error message, modify the API settings, and then test the API again.

   You can select **Save the correct response example automatically** based on your business requirements.

   - DataWorks automatically generates sample failure responses and error codes when it tests APIs. However, DataWorks does not automatically generate sample success responses.

     To allow DataWorks to save a success test result as a sample success response, you must select **Save the correct response example automatically** before you perform the test. If the response contains sensitive data that must be masked, you can manually edit the response.
   - A sample success response must be configured because it is an important reference for API callers.
   - The API call latency is the latency of the current API request. The latency is used to evaluate the API performance. If the latency is long, consider whether to optimize the database.

Test an API that has been published

1. Go to the **Data Service** page and click the **Service Management** tab in the upper-right corner.
2. In the left-side navigation pane, click **Test APIs**.
3. Select the API to be tested, set the request parameters, and then click **Test**.

> **Note**
>
> - On the Test APIs page, you can only test published APIs online, but not update sample success responses for APIs. To update the sample success response for an API, click the API name in the API list to go to the API Details page. Then, update the sample success response for the API in the Successful Response Example section.
> - You must test an API before you publish it.

15.10. Publish an API

API Gateway supports API lifecycle management, including API publishing, management, maintenance, and monetization. API Gateway provides a simple, fast, cost-effective, and low-risk service for you to aggregate microservices, separate the frontend from the backend, integrate systems, and provide features and data to partners and developers.

For security reasons, you must publish APIs that are created or registered in DataService Studio to API Gateway before the APIs are called by other users or in your own applications. In API Gateway, you can perform a variety of management operations on APIs. For example, you can manage permissions, configure bandwidth throttling, configure access control, and view statistics on APIs. DataService Studio is integrated with API Gateway to allow you to publish APIs to API Gateway with one click.

Publish an API from DataService Studio to API Gateway

Before you publish an API, you must activate API Gateway and register and test the API.

If the API passes the test, click **Submit** in the upper-right corner of the configuration tab of the API. After the API is submitted, click the **Version** tab in the right-side navigation pane. Find the version that you want to publish and click **Publish** in the Actions column. This way, you can publish the API to API Gateway with one click.

The system automatically registers the API with API Gateway during the publish process. The system also creates a group in API Gateway with the same name as the API group to which the API belongs in DataService Studio and publishes the API in this group. After the API is published, you can log on to the API Gateway console to view details of the API and configure features such as bandwidth throttling and access control on the API.

If you need to call the API in your own application, create the application and authorize the application to use the API in the API Gateway console. Then, call the API in your application by signing the API request with the AppKey and AppSecret of the application. For more information, see *API Gateway Documentation*. API Gateway also provides SDKs for mainstream programming languages. You can use the SDKs to integrate APIs with your application.

15.11. Call an API

This topic describes how to call an API after the API is published to API Gateway.

Prerequisites

An API is published to API Gateway. For more information, see Publish APIs.

The following conditions are met:

- The parameter definition of the API is obtained.
- The AppKey and AppSecret of the application that needs to call the API are obtained.
- The application is authorized to call the API.

Context

API Gateway allows you to authorize access to APIs and use SDKs to integrate APIs with applications. You can authorize your own account, a user in your enterprise, or a third party to call APIs.

Procedure

1. Obtain the API documentation.

   The method of obtaining the API documentation varies based on how you obtain an API. You can use one of the following methods to obtain an API:

   - Purchase the API in Alibaba Cloud Marketplace.
   - Obtain the API authorization from the API provider.

2. Create an application.

   In API Gateway, applications define the identities that you use to call APIs. Each application has a key pair that consists of an AppKey and an AppSecret, which are equivalent to an account and its password.

3. Obtain the permissions to call the API.

   Authorization is to grant an application the permissions to call an API. Your application must be authorized before it can be used to call an API. The authorization method varies based on how you

obtain an API.

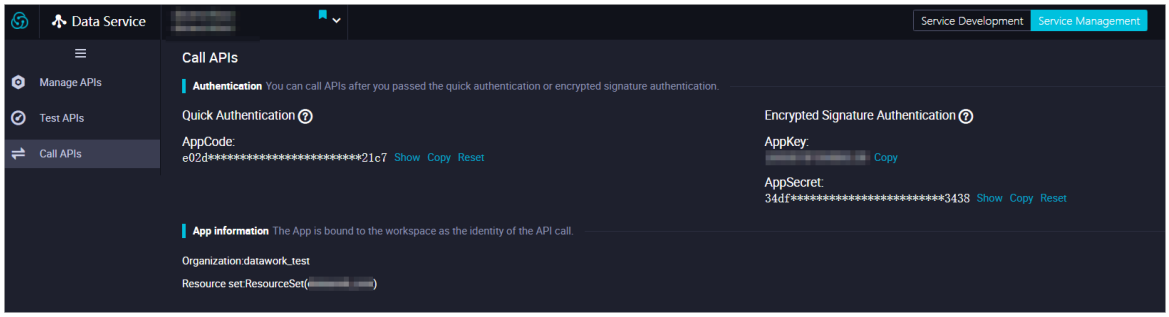
4. Call the API.

   You can send an HTTP or HTTPS request to call the API. Before you call the API, you can test the call by using the API calling examples provided in the API Gateway console. The examples are provided in multiple programming languages.

View the authentication information for calling APIs

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Development > Data Service**.
3. On the **Data Service** page, click the **Service Management** tab in the upper-right corner.
4. In the left-side navigation pane, click **Call APIs**.

   On the **Call APIs** page, you can view or copy the authentication information that is required to call APIs. You can also view the information about the applications that are bound to the current workspace. This helps you find the applications in the API Gateway console.



15.12. Service orchestration

The workflow feature, also called service orchestration, provides a composite API service. This topic describes the benefits of workflows and how to use workflows.

In DataService Studio, a workflow is represented as a directed acyclic graph (DAG). By dragging and dropping nodes to a DAG, you can arrange APIs and functions in a serial, parallel, or branch structure based on the business logic.

When you run a workflow to call APIs, DataWorks runs the nodes in the workflow in sequence, passes parameters among the nodes, and automatically changes the status of each node. The workflow feature simplifies the process of calling multiple APIs or functions and reduces the cost of O&M. This allows you to focus on business development.

Benefits

- Reduced cost of developing APIs

  By dragging nodes to a DAG, you can arrange APIs and functions in a serial, parallel, or branch structure without writing code. This reduces the cost of developing APIs.

- Higher performance in calling APIs and functions

  A workflow allows you to call multiple APIs and functions in a container. Compared with writing code to call APIs and functions, the workflow feature reduces the latency of calling APIs and functions and greatly improves the calling performance.

- Serverless architecture

  The workflow feature adopts a serverless architecture. A serverless architecture supports automatic resource scaling based on business needs. You can only focus on the business logic.

Values of request and response parameters

DataService Studio uses JSONPath to obtain parameter values. JSONPath is a query language that allows you to extract data from JSON files.

For example, three nodes are run in the following order: A, B, and then C. Node C needs to use the response parameters of Node A and Node B.

- Response parameter of Node A: {"namea":"valuea"}

  Expression for obtaining the value of the response parameter of Node A: ${A.namea}

- Response parameter of Node B: {"nameb":"valueb"}

  Expression for obtaining the value of the response parameter of Node B: $.nameb or ${B.nameb}

The built-in start node provides request parameters for the whole workflow. Assume that a request parameter of a workflow is {"namewf":"valuewf"}. All nodes of the workflow can obtain the value of the request parameter by using the ${START.namewf} expression.

Parameters

Request parameters:

- If you do not specify a value for a request parameter of a node, DataService Studio obtains the value of the same parameter from the first layer of the JSON string that is returned by the parent node. Then, DataService Studio assigns the value to the request parameter. If no value is specified for a request parameter of the first node, DataService Studio obtains the value of the same parameter from the request parameters of the workflow.
- If you specify a value for a request parameter of a node, DataService Studio uses the value that you specify.
- If you need to use the value of the specified parameter that is returned by the specified ancestor node, obtain the value by using a JSONPath expression.

Common JSONPath expressions for obtaining parameter values:

- $.: obtains the response parameters of the parent node.
- $.param: obtains the value of the param parameter that is returned by the parent node. DataService Studio enhances JSONPath expressions to help you obtain the response parameters of an ancestor node.
- ${NODEID1}: obtains response parameters of the node whose ID is NODEID1.
- ${START}: obtains the request parameters of the workflow, which are the response parameters of the start node.
- ${NODEID1.param}: obtains the value of the param parameter returned by the node whose ID is NODEID1.

Settings of the response parameters of a node:

- $.: sets the response parameters of the current node.
- $.param: sets the param parameter to be returned by the current node.
- ${NODEID1.param}: obtains the value of the param parameter returned by the node whose ID is NODEID1.

Example

Add a connection before you create and use a workflow. In this example, a MySQL connection is used.

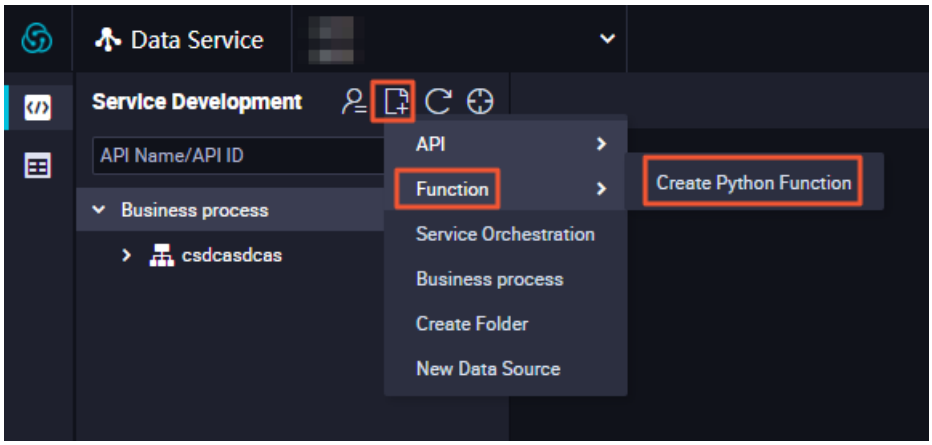
1. Register an API.

   In this example, create an API by using the registration method. For more information, see Register an API.

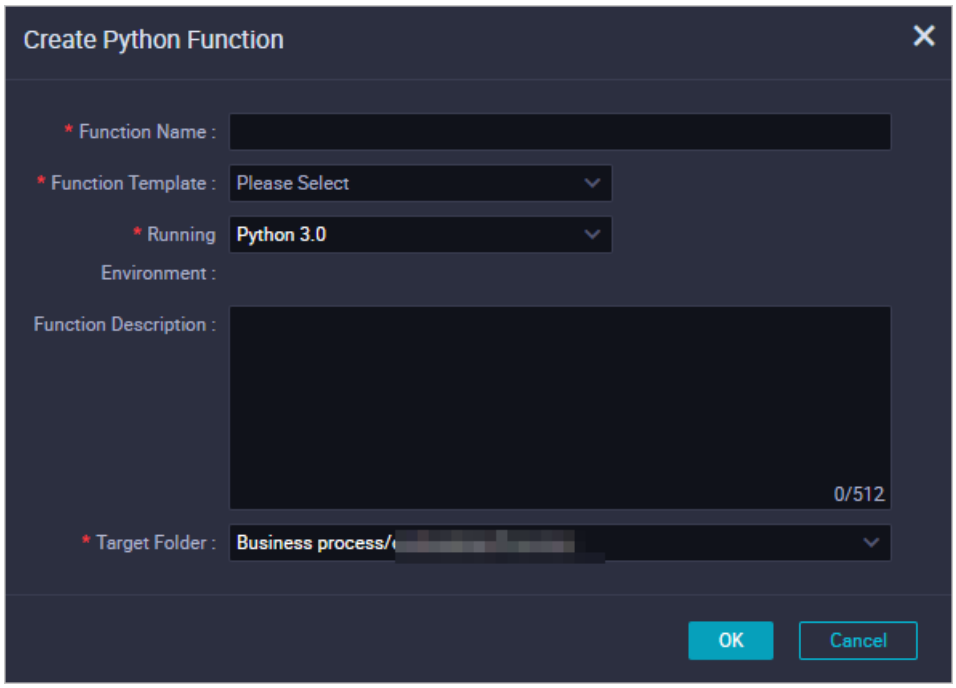
2. Register a function. For more information, see Manage functions.

   In this example, create a Python function as a branch node to process the result data of the parent node.

   i. Go to the **Service Development** tab, move the pointer over the **+Create** icon and choose **Create Function > Create Python Function**.



ii. In the **Create Python Function** dialog box, set the parameters as required.



Parameter	Description
Function Name	The name of the function. The name can be up to 256 characters in length.
Function Template	The template that is used to create the function. Set the value to Python3 Standard v1.
Running Environment	The runtime environment of the function. Set the value to Python 3.0.
Function Description	The description for the function. The description can be up to 512 characters in length.
Destination Folder	The folder for storing the function.

iii. Click **OK**.

iv. Configure the function on its configuration tab.

a. In the **Edit Code** section, enter the function code.

```
# -*- coding: utf-8 -*-
# event (str) : in filter it is the API result, in other cases, it is your para
m
# context : some environment information, temporarily useless
# import module limit: json,time,random,pickle,re,math
import json
def handler(event,context):
    # load str to json object
    obj = json.loads(event)
    # add your code here
    # end add
    return obj
```

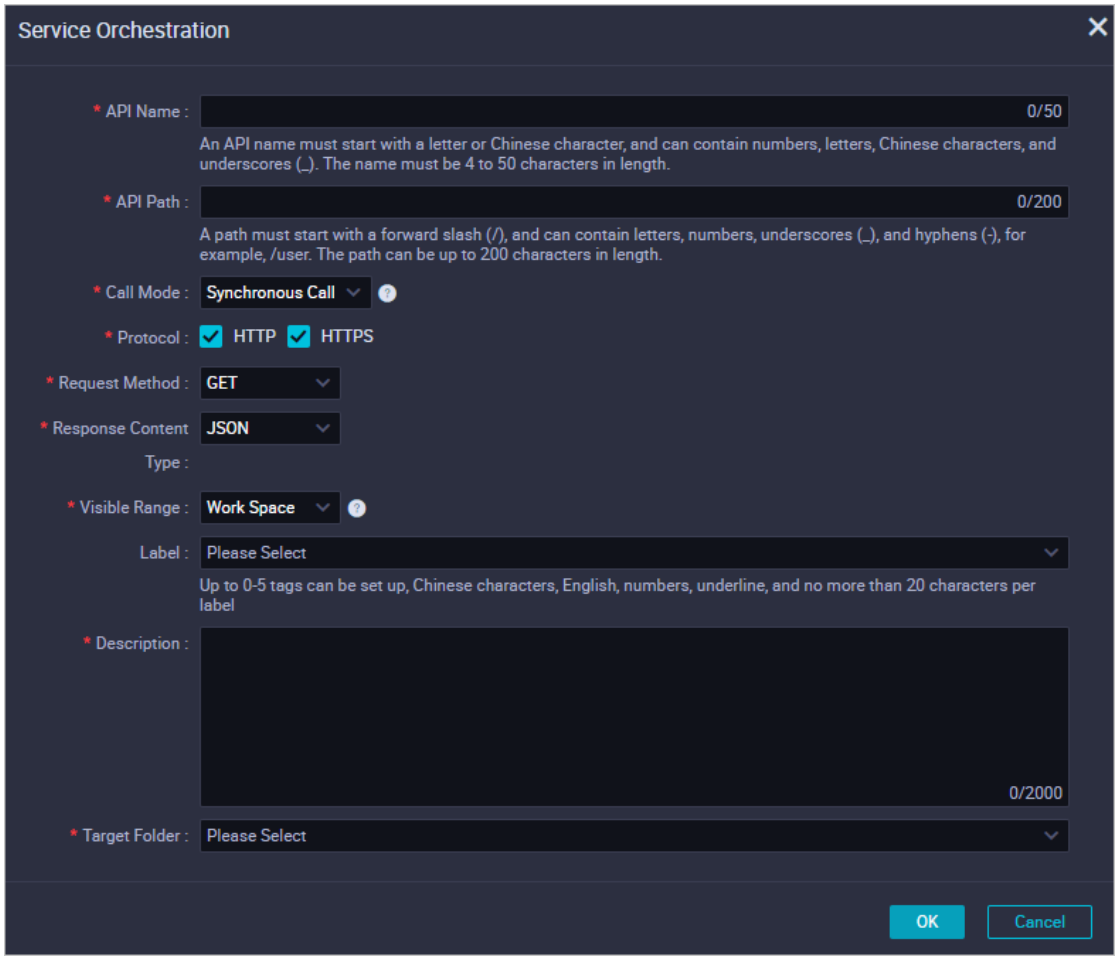
b. In the **Environment Configuration** section, set the **Memory** and **Function Timeout** parameters.

v. Click the icon in the top toolbar.

3. Create a workflow.

i. On the **Service Development** tab, move the pointer over the +Create icon and select **New Service Orchestration**.

ii. In the **Service Orchestration** dialog box, set the parameters as required.



Parameter	Description
API Name	The name of the API. The name must be 4 to 50 characters in length, and can contain letters, digits, and underscores (_). The name must start with a letter.
API Path	The path in which the API is stored. Example: */user*.
Call Mode	The mode used to call the API. Valid values: **Synchronous Call** and **Asynchronous Call**. ■ If you set this parameter to **Synchronous Call**, the API returns results immediately after it is called. The synchronous mode is most commonly used. ■ If you set this parameter to **Asynchronous Call**, the API returns the request ID immediately after it is called. The API caller can then obtain the call result from a message queue based on the request ID.
Protocol	The protocol used by the API. Valid values: **HTTP** and **HTTPS**.
Request Method	The request method. Valid values: **GET** and **POST**.
Response Content Type	The format of the data returned by the API. Set the value to **JSON**.

Parameter	Description
Visible Range	The range of users to whom the API is visible. Valid values: ■ **Work Space**: The API is visible to all members in the current workspace. ■ **Private**: The API is visible only to its owner, and permissions on the API cannot be granted to other members. **Note** If you set this parameter to Private, other members in the workspace cannot view the API in the API list.
Label	Select tags from the **Label** drop-down list. For more information, see Manage tags. **Note** You can set a maximum of five tags for an API.
Description	The description of the API. The description can be up to 2,000 characters in length.
Destination Folder	The folder that stores the API.

iii. Click **OK**.

4. Configure the workflow.

i. On the configuration tab of the workflow, drag nodes to the DAG and connect them as required.

ii. Double-click the **API1** node to edit the node. Select the API that you registered earlier as the API to be called in the node.

Select **set output results** and enter `{"user_id":"$.data[0].id"}`.

Use JSONPath expressions to configure response parameters. The syntax for obtaining the value of a parameter is ${NodeA.namea}, which is the same as that for configuring request parameters. `user_id":"$.data[0].id"}` assigns the value of the id parameter of the first element in the data array to the user_id parameter. Then, the API1 node returns {"user_id":"value"} in JSON format.

iii. Double-click the **PYTHON1** node to edit the node. Select the function that you created earlier as the function to be called in the node.

iv. Double-click the **SWITCH2** node to edit the node. In the right-side pane that appears, click **Set branch conditions**. You can enter conditional expressions based on the response parameter of the parent node. For example, you can enter expressions in the format of `${Node ID.Parameter}>1` or `$.Parameter>1`. Conditional expressions support the following operators: ==, !=, >=, >, <=, <, &&, !, (), +, -, *, /, and %.

In this example, the user_id parameter is the response parameter of the API1 node and is used as the request parameter of the SWITCH1 node.

```
Branch Node 1: $.user_id != 1, indicating that Branch Node 1 is run if the value of
the user_id parameter is not 1.
Branch Node 2: $.user_id == 1, indicating that Branch Node 2 is run if the value of
the user_id parameter is 1.
```

v. Double-click the end node and then click the **Response Parameters** tab on the right side to set response parameters.

5. Click **Test** in the upper-right corner.
6. In the **Test APIs** dialog box, set the parameters as required and click **OK**.

   You can view the test result after the workflow is tested.

15.13. Version management

This topic describes how to manage versions of APIs, workflows, and functions in Data Service.

You can view and compare historical versions of APIs, workflows, and functions. Data Service generates a version record each time an API, a workflow, or a function is published.

View Versions

1. Log on to the DataWorks console, click the DataWorks icon in the upper-left corner, and then choose **All Products > Data Service**.
2. On the **Service Development** tab, double-click the API that you want to publish in the API list.

   > **Note** You can also click Function or Table in the left-side navigation pane and double-click the name of a workflow or function to manage its versions.

3. In the right-side navigation pane, click **Version**. In the **Version** dialog box, view historical versions of the API.

Parameter	Description
API ID	The ID of the API. Each API ID is unique.
Version	The version of the node. A version is generated each time the node is published. V1 indicates version 1 and V2 indicates version 2. The version number increases by 1 each time an additional version is generated.
Submitted By	The user who published the version.

Parameter	Description
Submitted At	The time when the version was published. The time is accurate to second.
Status	The status of the version. Value values: ○ **Release**: indicates that the version of the API is the latest version. ○ **Off-Line**: indicates that the version of the API is a historical version. ○ **Can Be Published**: indicates that the version of the API can be published.
Actions	The actions that you can perform on the API. Valid values: Publish, Version Details, and Roll Back. ○ Click **Version Details** in the Actions column to view the details of the API. ○ Click **Roll Back** in the Actions column to roll back to the specified version. After you click this button, the **Are you sure you want to roll back the current version?** message appears. Click **OK**. ○ Click **Publish** in the Actions column to publish the API.

View Versions

In the **Version** dialog box, select two versions to compare, and click **Contrast**. In the **History Version Contrast** dialog box, compare the code and parameters of the two versions.

> **Note** The information appears in the History Version Contrast message varies with the mode in which the API was created:
>
> - If the API was created in the codeless user interface (UI), the request parameters and response parameters of the two versions are compared.
> - If the API was created in the code editor, the SQL statements of the two versions are compared.

15.14. FAQ

This topic provides answers to commonly asked questions about DataService Studio.

- Q: Do I need to activate the API Gateway service?

  A: API Gateway provides you with high-performance and highly available API hosting services. If you need to make your APIs available to others, activate the API Gateway service first.

- Q: Where can I add and change connections?

A: After you log on to the DataWorks console, click the DataWorks icon in the upper-left corner and choose **All Products > Data Integration** to go to the **Data Integration** page. In the left-side navigation pane, choose **Sync Resources > Connections**. On the page that appears, perform the relevant configuration. DataService Studio automatically reads data from the connections that you have configured.

- Q: What is the role of an API group in DataService Studio? What is the relationship between an API group in DataService Studio and an API group in API Gateway?

  A: An API group is a set of APIs specific to a feature or scenario. It is the smallest organization unit in DataService Studio, which is similar to an API group in API Gateway. An API group in DataService Studio is equivalent to an API group in API Gateway. After you publish an API from DataService Studio to API Gateway, API Gateway automatically creates an API group with the same name.

- Q: How can I configure an API group appropriately?

  A: Typically, an API group includes APIs that provide similar features or resolve a specific issue. For example, a weather API group can include APIs that are used to check the weather by city and by longitude and latitude.

- Q: How many API groups can I create?

  A: An Alibaba Cloud account can create up to 100 API groups.

- Q: When do I need to enable the pagination feature for an API call so that its return results can be displayed on multiple pages?

  A: By default, an API call returns a maximum of 2,000 records. If an API call may return more than 2,000 records, enable the pagination feature. If you do not specify any request parameters, the API call usually returns a large number of records and the pagination feature is automatically enabled.

- Q: Do APIs created by DataService Studio support POST requests?

  A: APIs created by DataService Studio support GET and POST requests.

- Q: Do APIs created by DataService Studio support the HTTPS protocol?

  A: APIs created by DataService Studio support both HTTP and HTTPS protocols.

15.15. Appendix: DataService Studio error codes

After DataService Studio receives an API request, it returns a response that contains an error code. You can locate and troubleshoot issues based on the error code. This topic describes common error codes that are returned by DataService Studio.

Error code	Error message	Description
0	success	The error message returned because the data query has succeeded.
1108110583	query timeout	The error message returned because the query has timed out. The timeout occurs because the total runtime of the API in DataService Studio and the database exceeds the timeout period that is configured for the API.

Error code	Error message	Description
1108110519	param miss	The error message returned because specific required request parameters are not specified.
1108110584	api context failed	The error message returned because the system has failed to obtain context information based on the third party. The information includes the connection information of the data source, AccessKey information of the data source, and tenant information.
1108110622	datasource query error	The error message returned because the system has failed to query the data source. The failure may occur because the SQL syntax is invalid, the data source does not respond within the configured timeout period (10s), or the number of connections to the data source exceeds the upper limit.
1108110703	database connection error	The error message returned because the data source has failed to be connected.
Other error codes	Other error messages	If the response contains an error code other than the preceding error codes, you can consult technical support personnel.

16.Stream Studio

16.1. Overview

Built on Alibaba Cloud Realtime Compute, which is based on Apache Flink, Stream Studio allows you to develop real-time computing nodes in directed acyclic graph (DAG) mode or SQL mode. You can switch between the two modes to edit the code or drag and drop components and configure them in a visual way.

As an ideal platform for developing real-time computing nodes, Stream Studio has the following features:

- Supports developing nodes in DAG mode. You can perform drag and drop components to configure real-time computing nodes.
- Supports developing nodes in SQL mode. You can edit the code to configure real-time computing nodes.
- Supports switching between the DAG mode and the SQL mode for you to easily check SQL operators.
- Supports using Function Studio to create and publish user-defined functions (UDFs) online in exclusive mode.
- Supports smart diagnosis for real-time computing nodes to facilitate online troubleshooting.

16.2. Bind a Realtime Compute project

1. Log on to the DataWorks console.
2. Click the **Project Manage** icon in the upper-right corner. The **Project Management** page appears.
3. On the **Project Management** page, click **Add Compute Engine** in the **Compute Engine** section, and select **Add engine service**.
4. Enter the name of the Blink engine to be added and click **Bind**.

16.3. Create a real-time computing node

This topic describes how to create a real-time computing node and develop data in Stream Studio.

Prerequisites

A workflow is created. You can create real-time computing nodes and develop data under an existing workflow.

Procedure

After a workflow is created, you can create real-time computing nodes under the workflow. By default, data is developed for a real-time computing node in directed acyclic graph (DAG) mode.

1. Right-click the workflow that you have created and select **Create task**.
2. In the **Create Node** dialog box that appears, set the parameters and click **Submit**.
3. Develop data in DAG mode on the **Components** page.

The Components page includes the following four sections:

- Component list section: In this section, you can view the list of available components. You can click **Components** on the left-side navigation submenu to go to the Components page and view the list.
- DAG section: In this section, you can drag and drop components to the DAG and connect them. To configure the dependency between two components, click and hold the highlighted dot at the bottom of a component and move the pointer to link this component with a descendant component. A DAG corresponds to a real-time computing node.
- Parameter configuration section: Double-click a component in the DAG. Then, you can set the related parameters in this section.
- Toolbar section: In this section, you can click the icons to perform the save, submit, steal lock, pre-compilation, test, stop, reload, and format operations respectively.

When you configure the DAG, you can right-click a component and select an operation from the menu that appears to perform it on the selected component. Available operations include **Rename**, **View schema**, **Delete node**, **View error message**, **New component group**, and **Copy**.

16.4. Get started with Stream Studio

This topic uses an example to describe how to use Stream Studio to develop and manage a real-time computing node. Stream Studio allows you to create, configure, publish, run, stop, and unpublish a real-time computing node.

Prerequisites

A Realtime Compute project is bound to the current DataWorks workspace.

Context

- Data store: a Datahub table with a created topic, which contains the `m_name, id, m_type, and tag` fields.

  > **Note** The Datahub topic must be created in advance.

- Data processing: splits the tag field by using the `semicolon (;)` as the delimiter to the color, mode, and weight fields.
- Output data: writes the `id, m_type, and weight` fields to a Log Service table.

  > **Note** The Log Service project and Logstore must be created in advance.

Procedure

1. Log on to the DataWorks console. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Stream Studio**.
2. Create a workflow.
   i. On the Stream Studio homepage, click **New business process**.

   You can also move the pointer over the **Create** icon and click **Business process**.

ii. In the **Create business process** dialog box that appears, set the **Business Name** and **Description** parameters.

iii. Click **New**.

3. Create a real-time computing node.

i. Right-click the workflow that you have created and select **Create task**.

ii. In the **Create node** dialog box that appears, set the relevant parameters.

iii. Click **Submit**. The **Components** page appears.

iv. On the left-side navigation submenu, click **Resource Reference** and select **PUBLIC_COMMON**.

> **Note** If this option is not selected, the message shown when you use the **FixedFieldsSplit** component.
>
> You can also go to the Resource Reference page to configure the reference resources after this message appears.

4. Configure the created real-time computing node.

On the left-side navigation submenu, click **Components**. The **Components** page appears.

i. Configure the data store of the node on the **Components** page.

a. Drag and drop the Datahub component in the Data Source section to the directed acyclic graph (DAG).

b. Click the Datahub component and set the parameters as needed on the **Parameter configuration** tab that appears.

Parameter	Description
oriTableName	The table name used in the CREATE TABLE statement. It must be a globally unique name. If this parameter is not specified, the real table name is used.
table schema	The custom fields and attribute fields to be returned. Click **Custom**. In the **Select field** dialog box that appears, click **+ Add** and enter the name and type of the output field. Then, click **OK**.
endPoint	The endpoint used to access Datahub. It corresponds to the endPoint parameter in the WITH clause of the CREATE TABLE statement.
accessId	The AccessKey ID used to read data from Datahub. It corresponds to the accessId parameter in the WITH clause of the CREATE TABLE statement.
accessKey	The AccessKey secret used to read data from Datahub. It corresponds to the accessKey parameter in the WITH clause of the CREATE TABLE statement.

Parameter	Description
project	The name of the Datahub project from which data is to be read. It corresponds to the project parameter in the WITH clause of the CREATE TABLE statement.
topic	The name of the Datahub topic from which data is to be read. It corresponds to the topic parameter in the WITH clause of the CREATE TABLE statement.
startTime	The beginning of the time range when data is read. It corresponds to the startTime parameter in the WITH clause of the CREATE TABLE statement.
maxRetryTimes	The maximum number of retries for reading data from Datahub. It corresponds to the maxRetryTimes parameter in the WITH clause of the CREATE TABLE statement. Default value: *20*.
retryIntervalMs	The retry interval at which data is read. It corresponds to the retryIntervalMs parameter in the WITH clause of the CREATE TABLE statement. Unit: milliseconds. Default value: *1,000*.
batchReadSize	The number of data records that are read at a time. It corresponds to the batchReadSize parameter in the WITH clause of the CREATE TABLE statement. Default value: *10*.
lengthCheck	The rule for checking the number of fields parsed from a row of data. It corresponds to the lengthCheck parameter in the WITH clause of the CREATE TABLE statement. Default value: *NONE*.
columnErrorDebug	Specifies whether to enable debugging. It corresponds to the columnErrorDebug parameter in the WITH clause of the CREATE TABLE statement. If you turn on this switch, logs about parsing errors are returned. You can view the node details Operation Center.

ii. Configure the data operator.

a. Drag and drop the **FixedFieldsSplit** component to the DAG to split the tag field.

b. Click and hold the highlighted dot at the bottom of the **Datahub** component and move the pointer to link this component with the **FixedFieldsSplit** component.

c. Click the **FixedFieldsSplit** component and set the field to tag and the column separator to semicolon (;) on the Parameter configuration tab that appears.

d. Click **Custom** for the Add column parameter. In the **Select field** dialog box that appears, click + Add and enter the name and type of the output field. Then, click **OK**.

e. Drag and drop the **Select** component to the DAG. Click and hold the highlighted dot at the bottom of the **FixedFieldsSplit** component and move the pointer to link this component with the **Select** component.

f. Click the **Select** component and click **0 Field has been selected** on the **Parameter configuration** tab that appears.

g. In the dialog box that appears, select the fields to be returned and click **OK**.

iii. Configure the result table.

This example uses the LogService component as the destination.

a. Drag and drop the **LogService** component to the DAG. Click and hold the highlighted dot at the bottom of the **Select** component and move the pointer to link this component with the **LogService** component.

b. Click the **LogService** component and set the parameters as needed on the **Parameter configuration** tab that appears.

Parameter	Description
oriTableName	The table name used in the CREATE TABLE statement. It must be a globally unique name. If this parameter is not specified, the real table name is used.
Output Field	The fields to be returned. Click **0 Field has been selected** for the Output Field parameter. In the dialog box that appears, select the fields to be returned and click **OK**.
endPoint	The endpoint used to access Log Service. It corresponds to the endPoint parameter in the WITH clause of the CREATE TABLE statement.
project	The name of the Log Service project to which data is to be written. It corresponds to the project parameter in the WITH clause of the CREATE TABLE statement.
topic	The name of the Log Service topic to which data is to be written. It corresponds to the topic parameter in the WITH clause of the CREATE TABLE statement.
source	The name of the Log Service table to which data is to be written. It corresponds to the source parameter in the WITH clause of the CREATE TABLE statement.
accessId	The AccessKey ID used to access Log Service. It corresponds to the accessId parameter in the WITH clause of the CREATE TABLE statement.
accessKey	The AccessKey secret used to access Log Service. It corresponds to the accessKey parameter in the WITH clause of the CREATE TABLE statement.
mode	The mode of data writing. It corresponds to the mode parameter in the WITH clause of the CREATE TABLE statement. Default value: *random*.
logStore	The name of the Logstore in the Log Service project to which the data is to be written. It corresponds to the logStore parameter in the WITH clause of the CREATE TABLE statement.

iv. Switch between the DAG mode and SQL mode.

Stream Studio allows you to configure a real-time computing node in both DAG mode and SQL mode. You can switch between these two modes.

By default, you configure a node in DAG mode. You can click **Switch to SQL mode** in the upper-right corner to switch to the SQL mode.

In SQL mode, you can click **Switch to DAG mode** in the upper-right corner to switch back to the DAG mode.

v. Configure the execution plan.

a. Click **Execution Plan** on the right-side navigation submenu to generate an execution plan.

b. Click **Save execution plan**.

5. Publish the real-time computing node.

You can publish the real-time computing node that you have configured. Click **Save** and then click **Submit** to publish the node.

i. Click **Save** and then click **Submit**. If you have not saved the node, a message appears, indicating that you must save it.

ii. In the **Submit New version** dialog box that appears, enter the remarks for the node and click **OK**.

iii. After you publish the node, you can go to the **OAM** page to view the node status and manage the node.

6. Perform O&M on the real-time computing node.

Click **OAM** in the upper-right corner to perform O&M on the real-time computing node.

i. Start the real-time computing node.

Find the real-time computing node that you have created in the node list and click **Start** to start the node.

You can set a custom start time for the real-time computing node based on your business requirement.

After starting the real-time computing node, you can click the node name to view its running status. If the real-time computing node is started properly, it enters the **Run** state.

ii. Stop and unpublish the real-time computing node.

a. Click **Stop** to stop the real-time computing node.

b. After the real-time computing node is stopped, click **Offline** to unpublish it.

Result

Now you have created, configured, published, run, stopped, and unpublished a real-time streaming node.

16.5. Configure components

16.5.1. Source tables

16.5.1.1. Datahub

Datahub is a real-time data distribution platform that is designed to process streaming data. It provides a channel for the Apsara Stack DTplus platform to process big data.

Realtime Compute typically uses Datahub to store source and result tables for streaming data processing. Data Transmission Services (DTS) and the Internet of Things (IoT) also use Datahub to access big data platforms. Datahub stores streaming data that can be used as input data for Realtime Compute.

Parameter configuration

Parameter	Description	Remarks
oriTableName	The table name used in the CREATE TABLE statement. It must be a globally unique name.	None
table schema	The custom fields and attribute fields to be read from Datahub.	None
endPoint	The consumer endpoint.	None
accessId	The AccessKey ID used to read data from Datahub.	None
accessKey	The AccessKey secret used to read data from Datahub.	None
project	The name of the Datahub project from which data is to be read.	None
topic	The name of the Datahub topic from which data is to be read.	None
startTime	The beginning of the time range when data is read.	The format is `yyyy-MM-dd hh:mm:ss`.
maxRetryTimes	The maximum number of retries for reading data from Datahub.	None
retryIntervalMs	The retry interval at which data is read. Unit: milliseconds.	None
batchReadSize	The number of data records that are read at a time.	None

Parameter	Description	Remarks
lengthCheck	The rule for checking the number of fields parsed from a row of data.	Valid values: *SKIP*, *EXCEPTION*, and *PAD*. Default value: SKIP. • *SKIP*: skips a data record when the number of fields in the data record is not the specified one. • *EXCEPTION*: throws an exception when the number of fields in the data record is not the specified one. • *PAD*: pads fields in sequence. Pad a field with null when the field does not exist.
columnErrorDebug	Specifies whether to enable debugging. If you turn on this switch, logs about parsing errors are returned.	None
BLOB	Specifies whether the type of data read from Datahub is BLOB.	None
Data Quality	Specifies whether to open the Data Quality page to view related monitoring nodes.	None

Field type mapping

The following table lists the mapping between Datahub and Realtime Compute data types. We recommend that you declare the type mapping in the DDL statement.

Datahub data type	Realtime Compute data type
BIGINT	BIGINT
DOUBLE	DOUBLE
TIMESTAMP	BIGINT
BOOLEAN	BOOLEAN
DECIMAL	DECIMAL

Attribute fields

You can obtain the attribute field indicating the system time at which each data record is written to Datahub.

Field	Description

Field	Description
System Time	The system time at which each data record is written to Datahub.

16.5.1.2. Log Service

As an all-in-one real-time data logging service, Log Service allows you to quickly finish tasks such as data ingestion, consumption, delivery, query, and analysis without any extra development work. This can help you improve O&M and operational efficiency, and build up the capability to process large amounts of logs in the data technology era.

Log Service stores streaming data that can be used as input data for Realtime Compute.

The data format of Log Service is consistent with JSON. Example:

```
{
    "a": 1000,
    "b": 1234,
    "c": "li"
}
```

Parameter configuration

Parameter	Description	Remarks
oriTableName	The table name used in the CREATE TABLE statement. It must be a globally unique name.	None
table schema	The custom fields and attribute fields to be read from Log Service.	None
endPoint	The consumer endpoint.	Log Service endpoints
accessId	The AccessKey ID used to read data from Log Service.	None
accessKey	The AccessKey secret used to read data from Log Service.	None
project	The name of the Log Service project from which data is to be read.	None
logStore	The name of the Logstore under the Log Service project.	None
consumerGroup	The name of the consumer group.	You can specify a custom consumer group name. The format of the name is not fixed.

Parameter	Description	Remarks
startTime	The beginning of the time range when the log data is consumed.	None
heartBeatIntervalMills	Optional. The heartbeat interval at which the client sends heartbeat messages. Unit: milliseconds.	None
maxRetryTimes	The maximum number of retries for reading data from Log Service.	None
columnErrorDebug	Specifies whether to enable debugging. If you turn on this switch, logs about parsing errors are returned.	None

Field type mapping

The following table lists the mapping between Log Service and Realtime Compute data types. We recommend that you declare the type mapping in the DDL statement.

Log Service data type	Realtime Compute data type
STRING	VARCHAR

Attribute fields

Currently, Log Service supports the following three attribute fields by default. You can also specify other custom fields.

Field	Description
`__source__`	Specifies a log source.
`__topic__`	Specifies a log topic.
`__timestamp__`	Specifies the time when a logged event occurs.

Note

- Currently, Log Service does not support the MAP type.
- We recommend that you define the fields in the same order as the fields in the preceding table. Unordered fields are also supported.
- If the input data is in JSON format, define the delimiter and use the built-in function JSON_VALUE to parse the JSON value. Otherwise, the parsing fails and the following error is returned:

```
2017-12-25 15:24:43,467 WARN [Topology-0 (1/1)] com.alibaba.blink.streaming.connectors.common.source.parse.DefaultSourceCollector - Field missing error, table column number: 3, data column number: 3, data field number: 1, data: [{"lg_order_code":"LP00000005","activity_code":"TEST_CODE1","occur_time":"2017-12-10 00:00:01"}]
```

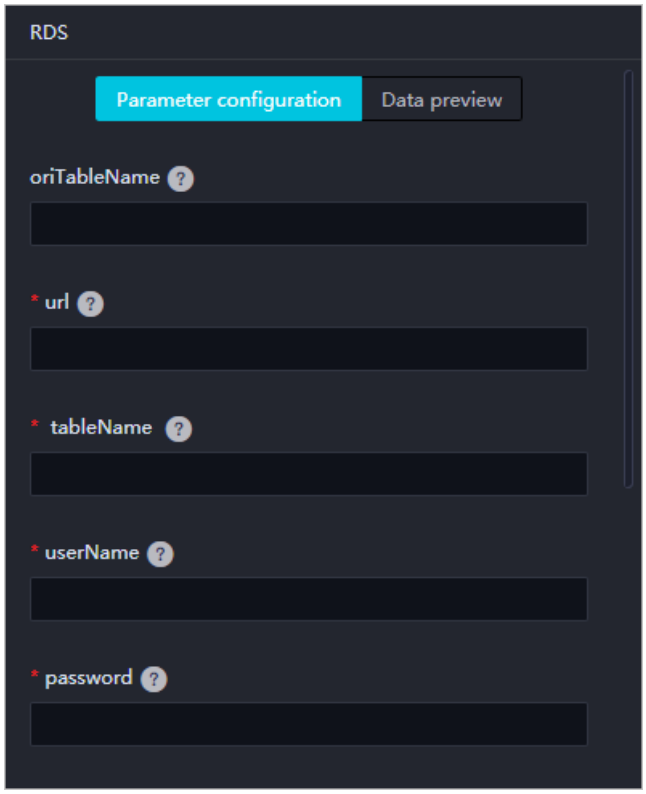
- The batchGetSize value must not exceed 1,000. Otherwise, an error occurs.
- The batchGetSize parameter specifies the number of log items read at a time in a log group. If both the size of a single log item and the batchGetSize value are too large, frequent garbage collection (GC) may be triggered. To avoid this, you must set batchGetSize parameter to a smaller value.

16.5.2. Dimension tables

16.5.2.1. ApsaraDB RDS

ApsaraDB RDS is a stable, reliable, and scalable cloud database service.

Parameter configuration



Parameter	Description	Remarks
oriTableName	The table name used in the CREATE TABLE statement. It must be a globally unique name.	None
url	The URL of your ApsaraDB RDS instance.	None
tableName	The name of the table in your database.	None
userName	The username that is used to connect to the database.	None
password	The password that is used to connect to the database.	None
Output Field	The fields that you want to return to the descendant component.	None
maxRetryTimes	The maximum number of retries for reading data from the table.	None
Cache Policy	The policy that is used to cache data.	Valid values: *None*, *LRU*, and *ALL*.
primaryKey	The primary key field in the output fields.	• You must specify a primary key when you declare a dimension table. • When you join a dimension table with another table, the ON clause must contain the equivalent (=) conditions for all the primary key fields. • The primary key in ApsaraDB RDS or PolarDB-X is the primary key or unique index column of an ApsaraDB RDS or PolarDB-X dimension table.

Additional information

- ApsaraDB RDS and PolarDB-X provide the following cache policies:
  - **None**: indicates that data is not cached.
  - **LRU**: indicates that only the recently used data is cached.

    If this cache policy is selected, you must specify the cacheSize and cacheTTLMs parameters.
  - **ALL**: indicates that all data is cached.

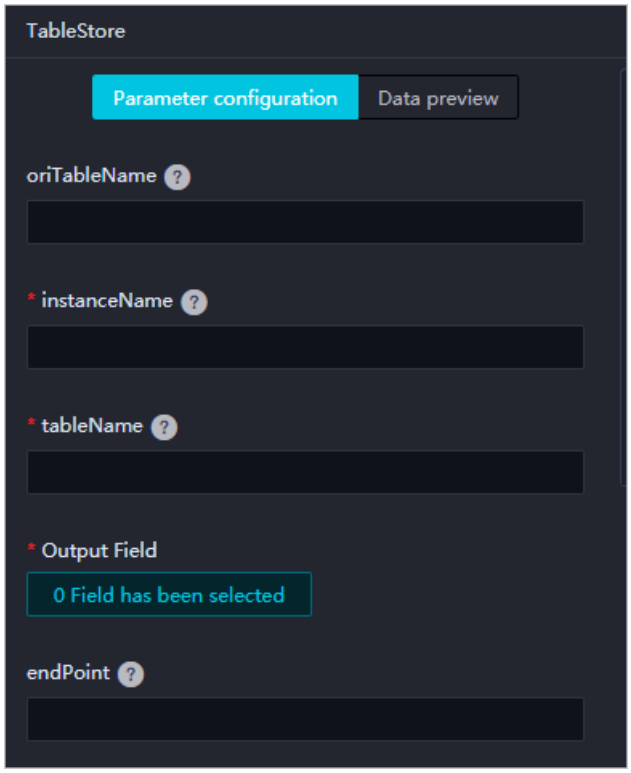
Before the system runs a node, it loads all the data in the remote table to the memory. Then, the system searches the cache for data in all dimension table queries. If a cache miss occurs, all data is cached again after the cache times out. The ALL cache policy applies to scenarios where the remote table is small but a large number of missing keys exist. If this cache policy is selected, you must specify the cacheTTLMs and cacheReloadTimeBlackList parameters.

- If the ALL cache policy is used, the system reloads data asynchronously. Therefore, you must increase the memory of the JOIN operator. The size of the increased memory is twice the data size of the remote table.
- If the ALL cache policy is used, pay special attention to the memory of the JOIN operator to prevent out of memory (OOM) errors.

16.5.2.2. Tablestore

Tablestore is a distributed NoSQL database service that is developed based on the Apsara distributed operating system. It features high availability and data reliability.

Parameter configuration



Parameter	Description
oriTableName	The table name used in the CREATE TABLE statement. It must be a globally unique name.
instanceName	The name of the instance.
tableName	The name of the table.
Output Field	The fields that you want to return to the descendant component.

Parameter	Description
endPoint	The endpoint of the instance. It corresponds to the endPoint parameter in the WITH clause of the CREATE TABLE statement.
accessId	The AccessKey ID of the account that you use to read data from the table.
accessKey	The AccessKey secret of the account that you use to read data from the table.
Cache Policy	The policy used to cache data. Valid values: **None** and **LRU**.
primaryKey	The primary key field in the output fields.

16.5.2.3. MaxCompute

This topic describes the parameter configuration, field type mapping, and metrics of a MaxCompute dimension table.

Parameter configuration

Parameter	Description	Remarks
oriTableName	The table name used in the CREATE TABLE statement. It must be a globally unique name.	None
endPoint	The endpoint used to access MaxCompute. It corresponds to the endPoint parameter in the WITH clause of the CREATE TABLE statement.	None
tunnelEndpoint	The endpoint of the Tunnel service. It corresponds to the tunnelEndPoint parameter in the WITH clause of the CREATE TABLE statement.	This parameter is required for a MaxCompute dimension table deployed in a Virtual Private Cloud (VPC).
project	The name of the MaxCompute project to which the dimension table belongs.	None
accessId	The AccessKey ID used to read data from MaxCompute.	None
accessKey	The AccessKey secret used to read data from MaxCompute.	None

Parameter	Description	Remarks
Output Field	The fields to be returned to the descendant component.	None
partition	The partition name of the MaxCompute dimension table.	None
maxRowCount	The maximum number of data records that can be read from the MaxCompute dimension table	None
Cache Policy	The policy for caching data.	Default value: **ALL**.
cacheSize	The maximum number of data records that can be cached.	This parameter is required if you set the Cache Policy parameter to `LRU`. Default value: 100,000.
cacheTTLMs	The time interval at which the cache is refreshed. It corresponds to the cacheTTLMs parameter in the WITH clause of the CREATE TABLE statement. Units: milliseconds.	This parameter specifies the cache refresh interval when the Cache Policy parameter is set to `ALL`. The cache is not refreshed by default.
cacheReloadTimeBlackList	Optional. The time period during which the cache is not refreshed. This parameter is valid when the Cache Policy parameter is set to `ALL`. During the time period specified by this parameter, for example, the Double 11 Shopping Festival, the cache is not refreshed.	This parameter is left empty by default. If you want to set this parameter, specify the time period in the format shown in the following example: `2017-10-24 14:00 -> 2017-10-24 15:00, 2017-11-10 23:30 -> 2017-11-11 08:00` Separate multiple time periods with commas (,). Separate the start and end time for a time period with the string "->".
primaryKey	The primary key field of the output fields.	None

Field type mapping

MaxCompute data type	Realtime Compute data type
TINYINT	TINYINT
SMALLINT	SMALLINT
INT	INT

MaxCompute data type	Realtime Compute data type
BIGINT	BIGINT
FLOAT	FLOAT
DOUBLE	DOUBLE
BOOLEAN	BOOLEAN
DATETIME	TIMESTAMP
TIMESTAMP	TIMESTAMP
VARCHAR	VARCHAR
STRING	STRING
DECIMAL	DECIMAL
BINARY	VARBINARY

Metrics

When you join the dimension table to another table, you can view metrics such as the correlation degree and cache hit ratio. You can use K-Monitor to view the metrics.

Query statement	Description
fetch qps	Queries the total number of queries per second (QPS) against the dimension table, including hits and misses. The metric name is `blink.projectName.jobName.dimJoin.fetchQPS`.
fetchHitQPS	Queries the number of hits (in QPS) against the dimension table, including cache hits and hits against the physical dimension table. The metric name is `blink.projectName.jobName.dimJoin.fetchHitQPS`.
cacheHitQPS	Queries the number of cache hits (in QPS) against the dimension table. The metric name is `blink.projectName.jobName.dimJoin.cacheHitQPS`.
dimJoin.fetchHit	Queries the correlation degree of the dimension table and the table to which the dimension table is joined. The metric name is `blink.projectName.jobName.dimJoin.fetchHit`.
dimJoin.cacheHit	Queries the cache hit ratio of the dimension table. The metric name is `blink.projectName.jobName.dimJoin.cacheHit`.

Note

- We recommend that you use Realtime Compute V2.1.1 and later.
- To use a MaxCompute dimension table, you must grant the read permission to the account for

accessing MaxCompute.

- When you declare a dimension table, you must specify the primary key. When you join a dimension table with another table, the ON condition must contain an equivalent condition that includes the primary key of either table.
- The primary key value for each row of a MaxCompute dimension table must be unique. Otherwise, the duplicate records are removed.
- If the dimension table is a partitioned table, Realtime Compute does not currently support writing the partition key column to the schema.
- When the cache policy is set to ALL, Realtime Compute reloads data asynchronously. Therefore, you must increase the memory of the JOIN operator. The size of the increased memory is twice the data size of the remote table.
- The following failover message may appear when you run a node:

```
RejectedExecutionException: Task
java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTas,
```

Generally, this message appears because dimension table joining in Realtime Compute V1.x has certain issues. We recommend that you upgrade Realtime Compute to V2.1.1 or later. If you want to continue using the existing version, we recommend that you pause the node and resume it after troubleshooting. To troubleshoot the failover, check the specific error information that was generated for the first failover record in the failover history.

16.5.3. Data operators

16.5.3.1. Filter

The Filter component allows you to configure filter conditions. It corresponds to the WHERE clause in SQL statements.

Parameter configuration

Enter the filter expression to configure this component. The filter expression supports functions and operators (=, <>, >, >=, <, and <=), for example, `city = 'Beijing'`.

16.5.3.2. GroupBy

The GroupBy component corresponds to the GROUP BY clause in SQL statements.

Parameter configuration

Parameter	Description
Select grouping field	The fields based on which data is grouped. You can specify multiple fields.
Output Field	The fields to be returned, that is, the fields to be selected. You can specify the fields in the same way that you configure the Select component.

16.5.3.3. Join

The Join component corresponds to the JOIN clause in SQL statements.

Parameter configuration

Parameter	Description
JoinMode	The JOIN mode to be used. Valid values: INNER JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, and FULL OUTER JOIN.
expression	The JOIN expression. An equijoin is supported, for example, leftId = rightId AND limit = 0, whereas a non-equijoin is not supported.
Select Field	The fields to be returned, that is, the fields to be selected.

16.5.3.4. Select

The Select component allows you to configure the fields to be returned and supports field expressions. It corresponds to SELECT statements.

Parameter configuration

Select or configure the output fields in the Select field dialog box.

You can select fields to be returned in the **Field list** section and set an alias for a field in the **Field alias** column. To set a field expression, click the Edit icon next to the target field name. In the Edit dialog box that appears, enter the required SQL statement.

16.5.3.5. UDTF

The UDTF component allows you to configure custom functions. It corresponds to the UDTF clause in SQL statements.

Parameter configuration

Parameter	Description
JoinMode	The JOIN mode for the custom function. Only *INNER JOIN* and *LEFT OUTER JOIN* are supported. • *INNER JOIN*: returns an empty result set when the UDTF clause returns no result. • *LEFT OUTER JOIN*: returns the NULL string when the UDTF clause returns no result.
Select function	The name of the function that the current node references. To reference a function for the current node, upload the related resources on the Resource Reference page and select the target resource.

Parameter	Description
parameter expression	The input parameters and output parameters of the referenced function.
Output Field	The fields to be returned. You can configure the name, alias, and expression of each field.

16.5.3.6. UnionAll

The UnionAll component corresponds to the UNION ALL clause in SQL statements.

Parameter configuration

No parameter configuration is required.

16.5.3.7. Dynamic column splitting

Dynamic column splitting allows you to split data records with a dynamic number of columns.

Example

Input data:

```
k1=v1,k2=v2,k3=v3,k4=v4
```

In the preceding example, the data is stored in key-value pairs in the format of key=value. Different data records may have different numbers of key-value pairs, that is, they may have different numbers of columns. In this case, you can use the first-level delimiter, which is comma (,) in the preceding example, to split the data to different key-value pairs. Then, you can use the secondary-level delimiter, which is equal sign (=) in the preceding example, to split each key-value pair to the key and value.

Parameter configuration

Parameter	Description
Select Field	The name of the field to split.
first level column delimiter	The delimiter used to split the field at the first level. Default value: \u0001.
secondary level column delimiter	The delimiter used to split the field at the secondary level. Default value: \u0002.
Add column	The fields that store the split data. Specify a key for each field. An alias is allowed.

16.5.3.8. Static column splitting

Static column splitting allows you to split data records with fixed columns that are separated by a fixed delimiter.

Example

You can use commas (,) as the delimiter to split the following data to four new columns, that is, 1111, 2222, 3333, and 4444.

```
1111,2222,3333,4444
```

The static column splitting method is applicable to data records with fixed columns that are separated by a fixed delimiter.

Parameter configuration

Parameter	Description
Select Field	The name of the field to split.
column separator	The delimiter used to split the field. You can use full-width characters or half-width characters as needed.
Add column	The fields that store the split data. Specify a key and a sequence number for each field. An alias is allowed.

16.5.3.9. Row splitting

Row splitting allows you split a row to multiple rows based on a field by using the specified delimiter.

Example

The following table lists the input data.

id	num
1	1,2

Split the row to multiple rows based on the num field by using the comma (,) as the delimiter, and place the split data in the new field new_num. The following table lists the output data.

id	num	new_num
1	1,2	1
1	1,2	2

Parameter configuration

Parameter	Description	Remarks
Select Field	The name of the field to split.	This parameter is set to num in the preceding example.

Parameter	Description	Remarks
Field separator	The delimiter used to split the field. Default value: (\n).	This parameter is set to comma (,) in the preceding example.
Define new column name	The name of the new field that stores the split data.	This parameter is set to new_num in the preceding example.

16.5.4. Result tables

16.5.4.1. Datahub

Datahub is a real-time data distribution platform that is designed to process streaming data. It provides a channel for the Apsara Stack DTplus platform to process big data. Datahub works with multiple Apsara Stack services to provide an end-to-end data processing solution. Realtime Compute typically uses Datahub to store source and result tables for streaming data processing.

Parameter configuration

Parameter	Description
oriTableName	The table name used in the CREATE TABLE statement. It must be a globally unique name.
Output Field	The fields to be returned.
endPoint	The endpoint used to access DataHub. It corresponds to the endPoint parameter in the WITH clause of the CREATE TABLE statement.
project	The name of the Datahub project to which data is to be written. It corresponds to the project parameter in the WITH clause of the CREATE TABLE statement.
topic	The name of the Datahub topic to which data is to be written. It corresponds to the topic parameter in the WITH clause of the CREATE TABLE statement.
accessId	The AccessKey ID used to access Datahub. It corresponds to the accessId parameter in the WITH clause of the CREATE TABLE statement.
accessKey	The AccessKey secret used to access Datahub. It corresponds to the accessKey parameter in the WITH clause of the CREATE TABLE statement.
maxRetryTimes	The maximum number of retries for writing data to DataHub. It corresponds to the maxRetryTimes parameter in the WITH clause of the CREATE TABLE statement.

Parameter	Description
batchSize	The number of data records that are written at a time. It corresponds to the batchSize parameter in the WITH clause of the CREATE TABLE statement.
batchWriteTimeoutMs	The interval at which the cache is cleared. It corresponds to the batchWriteTimeoutMs parameter in the WITH clause of the CREATE TABLE statement.
maxBlockMessages	The maximum number of data blocks that are written at a time. It corresponds to the maxBlockMessages parameter in the WITH clause of the CREATE TABLE statement.

Field type mapping

The following table lists the mapping between Datahub and Realtime Compute data types. We recommend that you declare the type mapping in the DDL statement.

Datahub data type	Realtime Compute data type
BIGINT	BIGINT
DOUBLE	DOUBLE
TIMESTAMP	BIGINT
BOOLEAN	BOOLEAN
DECIMAL	DECIMAL

16.5.4.2. Log Service

As an all-in-one real-time data logging service, Log Service allows you to quickly finish tasks such as data ingestion, consumption, delivery, query, and analysis without any extra development work. This can help you improve O&M and operational efficiency, and build up the capability to process large amounts of logs in the data technology era.

Parameter configuration

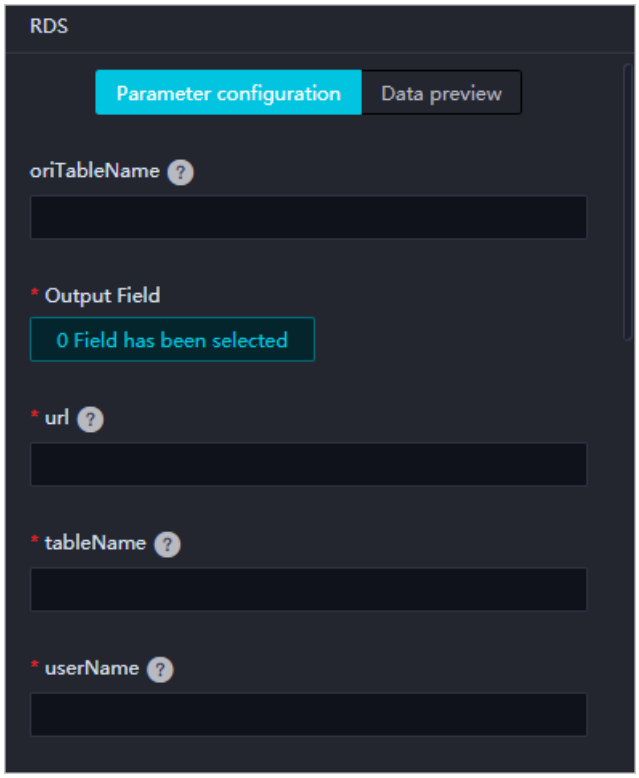
Parameter	Description
oriTableName	The table name used in the CREATE TABLE statement. It must be a globally unique name.
Output Field	The fields to be returned.
endPoint	The endpoint used to access Log Service. It corresponds to the endPoint parameter in the WITH clause of the CREATE TABLE statement.

Parameter	Description
project	The name of the Log Service project to which the data is to be written. It corresponds to the project parameter in the WITH clause of the CREATE TABLE statement.
primaryKey	The primary key field of the output fields.
source	The name of the log source. It corresponds to the source parameter in the WITH clause of the CREATE TABLE statement.
accessId	The AccessKey ID used to access Log Service.
accessKey	The AccessKey secret used to access Log Service.
mode	The mode of data writing. It corresponds to the mode parameter in the WITH clause of the CREATE TABLE statement. Default value: `random`. If you set this parameter to `partition`, data is written by partition.
logStore	The name of the Logstore in the Log Service project to which the data is to be written.

16.5.4.3. ApsaraDB RDS

ApsaraDB RDS is a stable, reliable, and scalable cloud database service.

Parameter configuration



Parameter	Description
oriTableName	The table name used in the CREATE TABLE statement. It must be a globally unique name.
Output Field	The fields that you want to write to the table in your ApsaraDB RDS database.
url	The URL used to access your ApsaraDB RDS instance. This parameter corresponds to the url parameter in the WITH clause of the CREATE TABLE statement.
tableName	The name of the table to which you want to write data. This parameter corresponds to the tableName parameter in the WITH clause of the CREATE TABLE statement.
userName	The username that is used to access your ApsaraDB RDS database. This parameter corresponds to the userName parameter in the WITH clause of the CREATE TABLE statement.
password	The password that is used to access your ApsaraDB RDS database. This parameter corresponds to the password parameter in the WITH clause of the CREATE TABLE statement.
maxRetryTimes	The maximum number of retries for writing data to the table. This parameter corresponds to the maxRetryTimes parameter in the WITH clause of the CREATE TABLE statement.

Parameter	Description
batchSize	The number of data records that can be written at a time. This parameter corresponds to the batchSize parameter in the WITH clause of the CREATE TABLE statement.
bufferSize	The maximum number of data records that can be stored in the buffer before deduplication is triggered. This parameter corresponds to the bufferSize parameter in the WITH clause of the CREATE TABLE statement. You can use this parameter only after the primaryKey parameter is specified.
flushIntervalMs	The time interval at which the buffer is cleared. Unit: milliseconds. This parameter corresponds to the flushIntervalMs parameter in the WITH clause of the CREATE TABLE statement.
excludeUpdateColumns	The fields that are not updated when Realtime Compute updates data records with the same primary key value. This parameter corresponds to the excludeUpdateColumns parameter in the WITH clause of the CREATE TABLE statement.
ignoreDelete	Specifies whether to skip delete operations. This parameter corresponds to the ignoreDelete parameter in the WITH clause of the CREATE TABLE statement.
partitionBy	Specifies the partitioning rule for the result table. Before Realtime Compute writes data to the sink node, Realtime Compute performs hash partitioning based on the value of this parameter. The data then flows to the relevant hash node. This parameter corresponds to the partitionBy parameter in the WITH clause of the CREATE TABLE statement.
primaryKey	The primary key field in the output fields.

Data type mapping

ApsaraDB RDS data type	Realtime Compute data type
TEXT	VARCHAR
BYTE	VARCHAR

ApsaraDB RDS data type	Realtime Compute data type
INTEGER	INT
LONG	BIGINT
DOUBLE	DOUBLE
DATE	VARCHAR
DATETIME	VARCHAR
TIMESTAMP	VARCHAR
TIME	VARCHAR
YEAR	VARCHAR
FLOAT	FLOAT
DECIMAL	DECIMAL
CHAR	VARCHAR

JDBC parameters

Parameter	Description	Default value	Required JDBC version
useUnicode	Specifies whether to use the Unicode character set. This parameter must be set to true if you set the characterEncoding parameter to gb2312 or gbk.	*false*	1.1g
characterEncoding	The character encoding format. This parameter must be set if the useUnicode parameter is set to true. You can set this parameter to gb2312 or gbk.	*false*	1.1g
autoReconnect	Specifies whether to automatically re-establish a connection if the connection to the database is unexpectedly interrupted.	*false*	1.1

Parameter	Description	Default value	Required JDBC version
autoReconnectForPools	Specifies whether to apply the reconnection policy to a database connection pool.	*false*	3.1.3
failOverReadOnly	Specifies whether to set the connection to read-only after the database is automatically reconnected.	*true*	3.0.12
maxReconnects	The maximum number of reconnection attempts allowed. This parameter must be set if the autoReconnect parameter is set to true.	*3*	1.1
initialTimeout	The interval between two reconnection attempts. Unit: seconds. This parameter must be set if the autoReconnect parameter is set to true.	*2*	1.1
connectTimeout	The timeout period when you use a socket connection to access the database server. Unit: milliseconds. Default value: 0. This value indicates that the connection never times out. This parameter applies to JDK 1.4 and later.	*0*	3.0.1
socketTimeout	The timeout period for read and write operations on a socket connection. Unit: milliseconds. Default value: 0. This value indicates that read or write operations never time out.	*0*	3.0.1

FAQ

- Q: When the output data of Realtime Compute is written to an ApsaraDB RDS table, is the result

table updated based on the primary key or is a new data record generated in the table?

A: The processing method depends on whether the primary key is defined in the DDL statement.

- If a primary key is defined in the DDL statement, the result table is updated by using `insert into on duplicate key update`. For a data record, if the primary key does not exist, the record is inserted into the table as a new row. If the value of the primary key field exists, the original row in the table is updated.
- If no primary key is defined in the DDL statement, new data records are inserted into the table by using `insert into`.

- Q: What do I need to pay attention to when I perform GROUP BY operations based on the unique index of an ApsaraDB RDS table?

  A: An ApsaraDB RDS table has only one auto-increment primary key. Therefore, this auto-increment primary key cannot be declared as the primary key in a Realtime Compute job. If you want to perform GROUP BY operations based on the unique index of the table, declare the unique index as the primary key in the job.

16.5.4.4. Table Store

Table Store is a distributed NoSQL database service built on the Apsara distributed operating system of Alibaba Cloud. Based on data sharding and load balancing technologies, Table Store has high performance in scaling out and handling concurrent transactions. You can use Table Store to store and query large amounts of structured data.

Parameter configuration

Parameter	Description
oriTableName	The table name used in the CREATE TABLE statement. It must be a globally unique name.
Output Field	The fields to be written to the Table Store table.
instanceName	The name of the Table Store instance. It corresponds to the instanceName parameter in the WITH clause of the CREATE TABLE statement.
tableName	The name of the Table Store table to which data is to be written. It corresponds to the tableName parameter in the WITH clause of the CREATE TABLE statement.
endPoint	The endpoint used to access Table Store. It corresponds to the endPoint parameter in the WITH clause of the CREATE TABLE statement.
accessId	The AccessKey ID used to access Table Store. It corresponds to the accessId parameter in the WITH clause of the CREATE TABLE statement.

Parameter	Description
accessKey	The AccessKey secret used to access Table Store. It corresponds to the accessKey parameter in the WITH clause of the CREATE TABLE statement.
valueColumns	The names of fields to be inserted to the result table. Separate multiple names with commas (,).
bufferSize	The buffer size after data deduplication. It corresponds to the bufferSize parameter in the WITH clause of the CREATE TABLE statement.
batchWriteTimeoutMs	The timeout period for writing data to Table Store. Unit: milliseconds. It corresponds to the batchWriteTimeoutMs parameter in the WITH clause of the CREATE TABLE statement.
batchSize	The maximum number of retries for writing data to Table Store. It corresponds to the batchSize parameter of the WITH clause in the CREATE TABLE statement.
retryIntervalMs	The retry interval at which data is written. It corresponds to the retryIntervalMs parameter in the WITH clause of the CREATE TABLE statement.
ignoreDelete	Specifies whether to skip DELETE operations. It corresponds to the ignoreDelete parameter in the WITH clause of the CREATE TABLE statement.
primaryKey	The primary key field of the output fields.

Field type mapping

Table Store data type	Realtime Compute data type
INTEGER	BIGINT
STRING	VARCHAR
BOOLEAN	BOOLEAN
DOUBLE	DOUBLE

16.5.4.5. MaxCompute

Realtime Compute supports creating a MaxCompute table as the result table.

Parameter configuration

Parameter	Description
oriTableName	The table name used in the CREATE TABLE statement. It must be a globally unique name.
tableName	The name of the MaxCompute table to which data is to be written.
Output Field	The fields to be written to the MaxCompute table.
endPoint	The endpoint used to access MaxCompute. It corresponds to the endPoint parameter in the WITH clause of the CREATE TABLE statement.
tunnelEndPoint	The endpoint of the Tunnel service, which is required for a MaxCompute project deployed in a Virtual Private Cloud (VPC). It corresponds to the tunnelEndPoint parameter in the WITH clause of the CREATE TABLE statement.
project	The name of the MaxCompute project to which data is to be written. It corresponds to the project parameter in the WITH clause of the CREATE TABLE statement.
accessId	The AccessKey ID used to access MaxCompute. It corresponds to the accessId parameter in the WITH clause of the CREATE TABLE statement.
accessKey	The AccessKey secret used to access MaxCompute. It corresponds to the accessKey parameter in the WITH clause of the CREATE TABLE statement.
partition	The partitions to which the data is to be written. It corresponds to the partition parameter in the WITH clause of the CREATE TABLE statement. This parameter must be specified for a partitioned table. For example, if the partition name of a table is `ds=20180905`, you can specify the parameter as `` `partition` = 'ds=20180905' ``. Separate multiple levels of partitions with commas (,), for example, `` `partition` = 'ds=20180912,dt=xxxyyy' ``.

Note Realtime Compute writes cached data to a MaxCompute table every time when a checkpoint is reached.

Field type mapping

MaxCompute data type	Realtime Compute data type
TINYINT	TINYINT
SMALLINT	SMALLINT
INT	INT
BIGINT	BIGINT
FLOAT	FLOAT
DOUBLE	DOUBLE
BOOLEAN	BOOLEAN
DATETIME	TIMESTAMP
TIMESTAMP	TIMESTAMP
VARCHAR	VARCHAR
STRING	STRING
DECIMAL	DECIMAL
BINARY	VARBINARY

FAQ

Q: Does a real-time computing node clear the result table before it writes data to the MaxCompute sink that is in Stream mode when `isOverwrite` is set to `true`?

A: The `isOverwrite` parameter is set to `true` by default. That is, a real-time computing node clears the result table and result data before it writes data to the sink. Every time a real-time computing node starts or resumes after being paused, it clears data of the existing result table or the result partition before it writes data. Certain data may be lost when data is cleared after a paused real-time computing node is resumed.

16.5.5. FAQ

This topic describes the frequently asked questions (FAQs) about Stream Studio.

Q: What computing engine do I need to activate before using Stream Studio?

A: You must first activate Realtime Compute because Stream Studio is a development platform based on Realtime Compute.

Q: Where can I create a Realtime Compute project? How do I bind the project to Stream Studio?

A: You can create a Realtime Compute project in the Realtime Compute console. After a project is created, you can bind it to an existing DataWorks workspace in the DataWorks console or directly create a workspace and bind the project to it. After the Realtime Compute project is bound to your workspace, you can develop real-time computing nodes in Stream Studio.

Q: What are the advantages of the directed acyclic graph (DAG) mode in Stream Studio? What are the similarities and differences between the DAG mode and SQL mode?

A: Stream Studio supports both the DAG mode and the SQL mode to develop real-time computing nodes. In DAG mode, you can perform drag-and-drop operations on components to configure real-time computing nodes without writing code. In this mode, what you see is what you get. You can also switch to the SQL mode to configure nodes by writing SQL statements.

Q: What types of SQL does Stream Studio support?

A: Realtime Compute is based on Apache Flink. Therefore, Stream Studio supports Flink SQL.

17.Data Protection

17.1. Overview

Data Protection is a data security management platform. It can be used to detect data assets, detect sensitive data, classify data, de-identify data, monitor data access behavior, report alerts, and audit risks.

Data Protection provides security management services for MaxCompute.

Access Data Protection

1. Log on to the DataWorks console.
2. On the DataWorks page that appears, click the icon in the upper-left corner and choose **All Products > Data Protection**.
3. Click **Try now** to go to the Data Protection page.

Features

Data Protection provides the following features:

- Intelligent sensitive data detection

  Data Protection automatically detects an enterprise's sensitive data based on self-training models and algorithms, and clearly displays statistics on data types, volume, and visitors. It also recognizes custom data types.

- Accurate data classification: Data Protection allows you to classify data and create custom levels for better data management.
- Flexible data de-identification

  Data Protection provides diverse and configurable methods for dynamic data de-identification.

- Risky behavior monitoring and auditing

  Data Protection uses various correlation analysis algorithms to detect risky behavior. It also provides alerts and supports visualized auditing for detected risks.

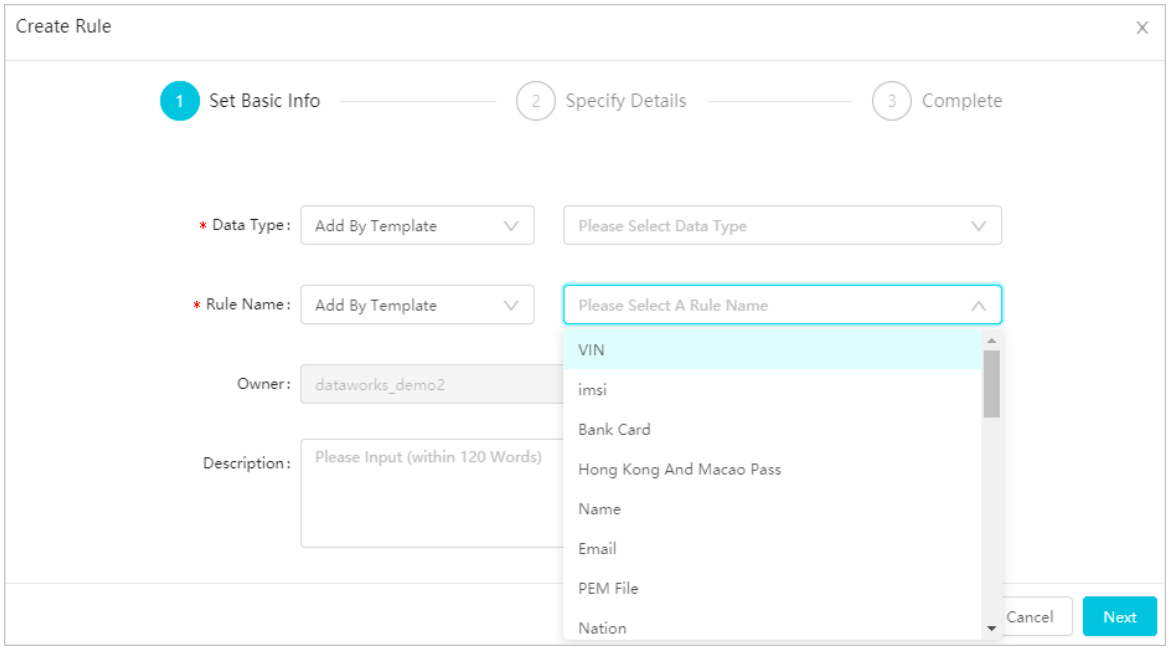
17.2. Configure rules for defining sensitive data

This topic describes how to configure rules for defining sensitive data.

Procedure

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Data Protection**.
3. Click **Try now** to go to the **Data Security Guard** page.
4. In the left-side navigation pane, choose **Management > Data Definition**. On the Data Recognition Rules page that appears, click **Create Rule**.
5. In the dialog box that appears, set parameters in the **Set Basic Info** step.

You can create a template-based identification rule or a custom identification rule.



Parameter	Description
Data Type	The category of the rule. You can select Template or Custom from the Data Type drop-down list. ○ If you select **Template**, you can select **Personal Information**, **Merchant Information**, or **Company Information** from the right drop-down list. ○ If you select **Custom**, you can enter a data type.
Data Name	○ If you select **Template** from the drop-down list, you can select a built-in identification rule template from the right drop-down list. You can select **Email**, **SeatNumber**, **MobilePhoneNumber**, **IP**, **MacAddress**, **CarNo**, **PostCode**, **IdCard**, or **BankCard**. ○ If you select **Custom**, you can enter a data name.
Owner	The owner of the rule.
Description	The description of the rule.

6. Click **Next**. Set the **Level** and **Data Definition** parameters.

Parameter	Description
Level	The security level of the sensitive data to which the rule is applied. If the existing security levels cannot meet your needs, click **Levels** in the left-side navigation pane to change the level settings.

Parameter	Description
Content Scanning	Specifies whether to enable content scanning. This option is selected by default for all the built-in data identification templates. ○ If you select a template, you cannot modify the identification rule, but you can verify the accuracy of the identification rule. ○ If you select regular expression matching, you can customize the identification rule.
Field Scanning	Specifies whether to enable field scanning. This approach provides two matching methods: exact matching and fuzzy matching of field names. Multiple-field matching is supported, and the relationship between the fields is OR.

7. Click **Next**. After you confirm the configuration, click **Save and Apply**.

> **Note** When you create a rule to define sensitive data, note the following:
> - The rule name must be unique.
> - The content scanning and field scanning configuration must be unique.
> - You can only view the sensitive data that is detected based on the data identification rule one day after the rule takes effect.

17.3. View the distribution of sensitive data

On the next day after you configure and activate sensitive data identification rules as a data security administrator, you can access Data Recognition to view the distribution of sensitive data.

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Data Protection**.
3. In the left-side navigation pane, click **Data Recognition**. On the Data Recognition page that appears, you can view the overall data distribution and field details.

17.4. View the information about data activities

On the next day after you configure and activate sensitive data identification rules as a data security administrator, you can access Data Activities to view related activity statistics, trend, and details.

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Data Protection**.
3. In the left-side navigation pane, click **Data Activities** to go to the **Data Activities** page.

The Data Activities page allows you to view the information of each activity that involves sensitive data. On the Manipulations and Queries tab, you can view the statistics, trend, user, and details of data access activities. On the Export tab, you can view the statistics and details of data export.

17.5. View the data audited as risky

Data activities are audited manually or based on the risk identification rules and AI-based identification rules. The Data Risks page displays data activities that are audited as risky. You can comment audit results as required.

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Data Protection**.
3. Click **Try now** to go to the **Data Security Guard** page.
4. In the left-side navigation pane, click **Data Risks** to filter and view the data audited as risky as needed.

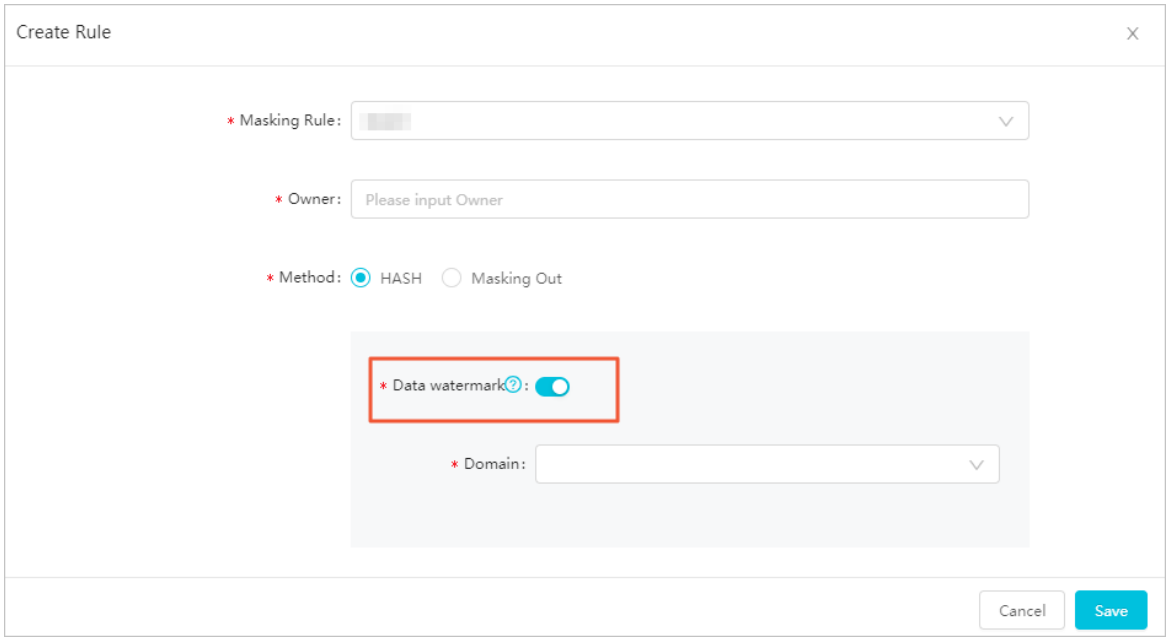
17.6. Track data

You can create a data tracking task to track the source of leaked data. You can also download a data tracking file and delete a data tracking task.

Configure a data watermark

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > Data Protection**.
3. On the page that appears, click **Try now**.
4. On the Data Protection page, choose **Rule Change > Data Masking** in the left-side navigation pane.
5. On the Data Masking page, click Create Rule to create a data masking rule. For more information, see Customize de-identification rules.

   When you configure a data masking rule, you can turn on **Data watermark**. This way, a data watermark can be embedded into data when data masking is performed on the data, which facilitates the tracking of the leaked data.



> **Note** A data watermark cannot be added to Chinese data.

Create a data tracking task

1. In the left-side navigation pane, click **Data traceability**.
2. On the **Data traceability** page, click **New data tracing task**.
3. Drag the file in which data is leaked to the middle part of the **Traceability tasks** dialog box or click **Upload** to upload the file.

   > **Note** You can upload a CSV file whose size is no more than 200 MB.

4. Click **Start tracing** to track the source of the leaked data.

   > **Note** After the tracking computing starts, you can close the dialog box, and the computing is not affected.

5. After the computing is complete, click the icon that corresponds to your data tracking task and view the tracking result.

   You can also enter a file name in the search box of the Data traceability page to view the historical data tracking task.

   > **Note** Tracking results are for reference only. A data tracking task may not generate results or generate one or more results.

Download a data tracking file

On the **Data traceability** page, find a data tracking file that you want to download in the Traceability file column and click the icon to download the file.

Delete a data tracking task

1. On the **Data traceability** page, find a data tracking task that you want to delete and click the icon.
2. In the message that appears, click **Confirm**.

17.7. Manage a self-generated data recognition model

This topic describes how to train a model that can be used to summarize the data characteristics of specific columns in a table based on these columns. It also describes how to use the trained model to identify sensitive data.

Go to the Self Generated Data Recognition Model tab

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > Data Protection**.
3. On the page that appears, click **Try now**.
4. On the Data Protection page, choose **Rule Change > Data Recognition Rules** in the left-side navigation pane.
5. Then, click **Self Generated Data Recognition Model**.

Create a self-generated data recognition model

1. On the **Self Generated Data Recognition Model** tab, click **Add Model**.
2. In the **Add Model** dialog box, configure the parameters.

Parameter	Description
Model Name	The name of the model that you want to create. The name can contain letters, digits, and underscores (_).
Owner	The name of the owner for the model. The name of the model can contain letters, digits, spaces, and underscores (_).
Selected Samples	You can select one or more columns from an existing MaxCompute table.

3. Click **Next** to go to the **Model Training** step.
4. Select **I accept data umbrella sampling for model training** and click **Start Training**.

> **Note** The column that you select must contain more than 10 rows. Otherwise, the system cannot start to train the model.

After the training starts, you can close the dialog box and view the training progress in the recognition model list.

5. View the status of the model in the recognition model list. After the training is complete, the status of the model becomes **Training Completed**. Click the icon in the Actions column. Then, the Assess step of the Edit Model dialog box appears.
6. View the recognition results in the Assess step. In this step, a maximum of 10 results are displayed. You can also adjust the results based on your business requirements. If the accuracy of the recognition can meet your requirements, click **Create**. Then, the model is created. If excessive mismatches exist, you can click **Retrain** to retrain the model after you adjust the recognition results.

> **Note** In most cases, you must perform two to three training to optimize the model.

Retrain the self-generated data recognition model

1. Click **Retrain** to retrain the model if the recognition effect presented by the recognition results in the Assess step is not satisfactory.
2. Navigate to the Select Samples step. The system automatically places the recognition results in the Assess step into the Sample Field and Exclude Field sections based on the recognition results. Click **Next** to continue the training.
3. View the recognition results after the training process is complete.
4. Click **Create** if the recognition results are satisfactory to finish creating the model. If the recognition results are still not satisfactory, click **Retrain** to continue retraining the model.

Use the self-generated data recognition model

1. After the model is created, click **Go To Data Recognition Rules Online Page**.
2. In the **Create Rule** dialog box, Configure the parameters.

Parameter	Description
Data Type	You can set this parameter to **Custom** or **Add By Template**.
Rule Name	The value cannot be changed.
Owner	The value cannot be changed.
Description	The description of the data recognition rule. The description can be a maximum of 120 characters in length and cannot contain special characters.

3. In the **Specify Details** step, specify **Level** and select **Field Scanning**. Select a trained model from the **Data Recognition Rules** drop-down list and click **Next**.
4. In the Complete step, click **Save**.

Stop training a self-generated data recognition model

1. Find a model that is being trained and click the icon in the Actions column to stop training the

model.

2. Find the model whose training is stopped and click the icon to continue the training.

Delete a self-generated data recognition model

Find a model that is not being trained and click the **Delete** icon in the Actions column to delete the model.

> Note

- A model that is being trained cannot be deleted. If you want to delete such a model, you can stop training the model and delete it.
- A model that is in use cannot be deleted. If you want to delete such a model, you can delete the data recognition rules configured for the model and delete the model.

17.8. Manage the data security levels

When creating a rule, you can specify a security level for the data to which the rule applies. On the Levels page, you can create and delete security levels. You can also modify the priority of each security level and manage rules by security level.

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Data Protection**.
3. Click **Try now** to go to the **Data Security Guard** page.
4. In the left-side navigation pane, choose **Management > Levels**.

   On the **Levels** page, you can create and delete security levels. You can also modify the priority of each security level and manage rules by security level.

Operation	Description
Create a security level	Click **Create Level**. Specify the security level name and operator.
Manage rules by security level	Find the target security level and click the icon in the Actions column. In the Manage Rules by Level dialog box that appears, you can select a rule and adjust its security level.
Delete a security level	Find the target security level and click the icon in the Actions column. In the dialog box that appears, click **Delete**.
Modify the priority of a security level	Find the target security level. Drag and drop the icon in the Actions column.

17.9. Manage data that is incorrectly detected

On the Manual Check page, you can manually correct the sensitive data that is incorrectly detected by rules. For example, you can delete incorrectly detected data, change the type of the detected data, and delete or recover data in batches.

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Data Protection**.
3. Click **Try now** to go to the **Data Security Guard** page.
4. In the left-side navigation pane, choose **Management > Manual Check**.

   On the Manual Check page, you can delete incorrectly detected data, change the type of the detected data, and delete or recover data in batches.

   - To delete a data record that is incorrectly detected, turn off the switch in the Status column of the data record.

     > **Note** You can recover data records that you have deleted.

   - To change the type of a data record, click the edit icon next to the name of the target rule and select a rule.

     > **Note** You can only select a rule that has been configured in DataWorks.

   - To delete or recover multiple data records at the same time, you can select the data records and click **Remove** or **Recover**.

17.10. Customize de-identification rules

This topic describes how to customize de-identification rules in Data Security Guard so that DataWorks can dynamically de-identify the results of ad hoc queries.

Prerequisites

Sensitive data detection rules are created and data security levels are specified. For more information, see Configure rules for defining sensitive data and Manage the data security levels.

Go to the Data Masking page

1. Log on to the DataWorks console.
2. On the DataStudio page, click the DataWorks icon in the upper-left corner and choose **All Products > Data Protection**.
3. Click **Try now**.
4. In the left-side navigation pane, choose **Management > Data Masking**.

The **Data Masking** page has two tabs: **Data Masking** and **Whitelist**.

Customize de-identification rules in Data Security Guard

1. Set the **Masking Scene** parameter to **Global Config (_default_scene_code)** and click **Create Rule** in the upper-right corner.
2. In the **Create Rule** dialog box, set the **Rule**, **Owner**, and **Method** parameters.

   > **Note** Data Security Guard provides three methods for de-identifying ID card numbers and email addresses, including **Pseudonymisation**, **Hashing**, and **Masking Out**. For other types of data, Data Security Guard only provides the **Hashing** and **Masking Out** methods.

   - **Pseudonymisation**

     This method replaces the text of a data record with an artificial pseudonym of the same data type. If you select this method, specify a security domain. Rules with different security domains generate different pseudonyms for the same data record.

   - **Hashing**

     If you select this method, specify a security domain. Rules with different security domains generate different hash values for the same data record.

   - **Masking Out**

     This method uses asterisks (*) to mask specified parts of a data record. It is commonly used.

Parameter	Description
Recommended	You can select recommended policies to mask data of common types such as ID card numbers and bank card numbers.
Custom	You can flexibly specify whether to mask the specified number of characters at the first, middle, or last part of a data record.

3. Click **OK**.
4. On the **Data Masking** tab of the **Data Masking** page, set the status of the de-identification rule to **Active** or **Inactive**.

   You can click the **Test** icon in the Actions column of the rule to test whether it works.

5. Click the **Whitelist** tab. On the Whitelist tab, click **Add Account**.
6. In the **Add Account** dialog box, set the **Rule**, **Account**, **Effective From**, and To parameters. For more information about user groups, see Manage user groups.

   > **Note** If you query data beyond the time range specified for the whitelist, the query results will be de-identified.

7. Click **Save**.

Verify the de-identification effect in DataWorks

After you create and configure de-identification rules, DataWorks dynamically de-identifies the results of queries in your workspace based on the rules.

> **Note** You must first turn on Mask Data in Page Query Results for your workspace in the DataWorks console.

17.11. Manage user groups

You can create a user group on the GroupManagement page and reference it in a de-identification whitelist. You can also copy, edit, and delete user groups on the GroupManagement page.

Go to the GroupManagement page

1. Log on to the DataWorks console.
2. On the DataStudio page, click the DataWorks icon in the upper-left corner and choose **All Products > Data Protection**.
3. Click **Try now**.
4. In the left-side navigation pane, choose **Management > GroupManagement**. The GroupManagement page appears.

Create a user group

1. On the **GroupManagement** page, click **Create Group** in the upper-right corner.
2. In the **Create Group** dialog box that appears, set the parameters described in the following table.

Parameter	Description
Name	Enter the name of the user group. **Note** The name of the user group must be unique.
Owner	Enter the owner of the user group.
Source Type	Specify the source of accounts in the user group. Valid values: ○ **Text**: If you select this option, click **Upload File** next to **Source File**, select a local file to upload, and then click **Open**. ○ **Select Existing Accounts**: If you select this option, select the accounts to add next to **Add Members** and click **>**.

3. Click **Save**.

Copy a user group

On the **GroupManagement** page, find the target user group and click in the Actions column. An identical user group is generated.

Note

- The name of the generated user group contains the -copy suffix. You can click to change the name.
- You can only copy the content but not the dependencies of a user group.

Edit a user group

To edit an existing user group, follow these steps:

1. On the **GroupManagement** page, find the target user group and click in the Actions column.
2. In the **Edit Group** dialog box that appears, modify parameters such as **Name**, **Owner**, and **Source Type**.
3. Verify the settings and click **Save**.

Delete a user group

To delete a user group, find the user group on the GroupManagement page and click **Delete** in the Actions column. In the dialog box that appears, click **Delete**.

Note You cannot delete a user group that is referenced in a de-identification whitelist.

If you still want to delete the user group, delete the user group from the corresponding de-identification whitelist first.

17.12. Automatically mark security levels for sensitive data

After you turn on Open Marking in DataWorks, DataWorks can identify sensitive data, automatically mark the security level of the sensitive data, and then display the security level as a label for the sensitive data that belongs to a MaxCompute project. The security level of the sensitive data is displayed in Data Map of DataWorks.

Precautions

If you turn on Open Marking in DataWorks, Data Protection automatically configures security levels for the columns that contain sensitive data. This configuration affects your access permissions on sensitive data. Fully evaluate the impact before you turn on Open Marking in DataWorks.

Turn on Open Marking in DataWorks

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > Data Protection**.
3. Click **Try now** to go to the **Data Security Guard** page.
4. In the left-side navigation pane, click **System Config**.
5. On the System Config page, turn on **Open Marking**. **MarkingOpen** is displayed on the right of the

switch.

17.13. Mask the underlying data of a MaxCompute project

This topic describes how to mask the underlying data of a MaxCompute project on the Data Masking page. After the data is masked, the data queried from each of the MaxCompute query entries is masked.

Prerequisites

- The data masking function base_meta.masking_v2 is available. If the function is unavailable, contact O&M personnel to publish the function.
- A network whitelist is enabled and SQL properties are configured for the MaxCompute projects whose underlying data needs to be masked. If no network whitelist is enabled, contact the O&M personnel.
- The rules for identifying sensitive data are created.

Go to the Data Masking page

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data governance > Data Protection**.
3. Click **Try now** to go to the **Data Security Guard** page.
4. In the left-side navigation pane, choose **Rule Change > Data Masking**.

Configure the rules for masking the underlying data of a MaxCompute project

1. On the **Data Masking** page, set **Masking Scene** to **MaxCompute Config(maxcompute_desense_code)**.
2. Select a MaxCompute project whose underlying data needs to be masked.
   i. Click **Select Desensitization Project**.
   ii. In the **Authorize The Desensitization Of Account** dialog box, select the name of the MaxCompute project whose underlying data needs to be masked in the **Not Desensitized Project** section, and click the icon to add it to the **Desensitized project** section.
   iii. Select **I agree to authorize data protection umbrella to desensitize the maxcompute underlying layer of the above projects.**
   iv. Click **OK**.
3. In the left-side navigation pane, choose Rule Change > Custom Identification Rules. On the **Rule Settings** tab of the page that appears, you can create, edit, or delete a data masking rule.

> **Note** The data masking rules that are configured for MaxCompute projects take effect only in underlying data masking scenarios, and can be edited or deleted only in such scenarios.

What's next

You can go to the MaxCompute project for which underlying data masking is enabled to check whether the data is masked.

18.App Studio

18.1. Overview

App Studio is a tool designed to help you develop data products. It comes with a rich set of front-end components that you can drag and drop to simply and quickly build front-end apps.

With App Studio, you do not need to download and install a local integrated development environment (IDE) or configure and maintain environment variables. Instead, you can use a browser to write, run, and debug apps and enjoy the same programming experience as that in a local IDE. App Studio also allows you to publish apps online.

Advantages

App Studio has the following core advantages:

- Data development anytime, anywhere

  You do not need to download and install a local IDE or configure and maintain environment variables. Instead, you can use a browser to develop data in your office, at home, or anywhere that you can connect to the network.

- Editor with complete features

  App Studio provides a browser-based editor that allows you to easily write, run, and debug projects. When you enter the code, App Studio provides code hinting, code completion, and repair suggestions. You can also find all references and the definition of a method to automatically generate code.

- Online debugging

  App Studio comes with all breakpoint types and operations of a local IDE. It supports thread switching and filtering, variable checking and watching, remote debugging, and hot code replacement.

- Multi-feature terminal

  You can directly access the runtime environment, which is currently built based on CentOS as the base image. The multi-feature terminal supports all bash commands, including vim and other interactive commands.

- Collaborative coding

  You and your team members can use App Studio to share the development environment for collaborative coding. Currently, App Studio allows a maximum of eight users to edit the same file of a project online concurrently, improving work efficiency. In the future, the collaborative coding component will support chatting, bullet screen messages, code annotations, videos, and other features to make teamwork efficient and pleasant.

- Plug-in system

  App Studio supports business plug-ins, tool plug-ins, and language plug-ins.

  - App Studio allows you to customize any required menu or add any service portal based on your business needs.
  - You can customize project management processes, project types, and templates dedicated to your business.

  - You can develop common tools, such as enhanced Git features, code rule scanning, keyboard shortcuts, enhanced editing features, and code snippets, and integrate them into App Studio.
  - You can use language plug-ins to enrich the languages supported by App Studio, enabling App Studio to serve users with more languages while addressing your own business needs.

- Visual building

  App Studio provides a WYSIWYG designer that has rich components and deeply integrates DataService Studio and DataStudio. Among all components of DataWorks, you can call DataWorks API operations only in App Studio. In addition to calling the API operations, you can quickly build front-end apps by dragging and dropping components and configuring them in the WYSIWYG designer based on the santa file system, developing web apps without code.

- Rich templates and flexible project management

  App Studio provides rich project templates, allowing you to develop your project accordingly with fewer steps and higher efficiency. You can also save your project as a template for future development and use, or share it with other users.

18.2. Get started with App Studio

To build a data portal, engineers need to develop data, build backend services, and develop front-end pages. This topic describes the basic features of App Studio and how to use App Studio.

Originally, DataWorks is mainly used by data engineers to implement offline or streaming data development. As DataWorks becomes increasingly easy to use, many roles such as algorithm engineers, BI analysts, operators, and product managers who are familiar with SQL can use DataWorks to develop data.

App Studio helps different types of users quickly build webpages for data viewing and apps for data query.

Go to the App Studio page

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > App Studio**. The **Projects** page appears.

Create a front-end project

App Studio provides complete front-end development capabilities that allow you to develop front-end projects in the same way as in a local integrated development environment (IDE). Without the need to master or understand any new concepts, you can create front-end projects in App Studio and develop HTML, CSS, JavaScript, and React files in a way that you are familiar with.

1. Create a project based on the sample project.
   i. Go to the **App Studio** page and click **Projects** in the left-side navigation pane. On the Projects page, click **Create Project from Code**.

ii. On the **Create Project** page, set the **Name** and **Description** parameters, and set the runtime environment to **react-demo**.

> **Note**
>
> - The name of the project must start with a letter and can contain digits, letters, underscores (_), and hyphens (-).
> - The description of the project can be 2 to 500 characters in length.

iii. After the configuration is completed, click **Submit**.

2. Set running parameters.

   In the upper-right corner, choose **Edit Config** > Edit Configurations. In the **Run/Debug Configurations** dialog box that appears, set the required parameter. Select the instance type and specify the port number as required. You can use the default configuration unless otherwise required. Then, click **OK**.

Parameter	Description
Install Cmd	The command used to install the dependency, for example, **npm install**.
Start Cmd	The command used to start the app, for example, **npm start**.
Environment Variables	The environment variables.
Initialize Script	The path of the script used to initialize a container in the code library.
PORT	The port of the Elastic Compute Service (ECS) instance. Default value: 3000.
ECS Instance	The instance type. Valid values: **1vCPU 2GMemory**, **2vCPU 3GMemory**, **4vCPU 8GMemory**, and **8vCPU 16GMemory**.

3. Run the project.

   Click the Run icon in the upper-right corner to run the project. Currently, you can run the `tnpm start` command to start front-end projects. You can seamlessly run projects with *webpack-dev-server* configured.

   During project running, you can view the dependency installation and app startup logs. After the project running is completed, the Preview tab appears in the right-side navigation pane. You can edit and save the code in real time. The edited code takes effect immediately.

4. Access the project.

   Click the **Preview** tab in the right-side navigation pane, and click the arrow next to the access link to open the project.

In App Studio, you can edit and develop front-end projects in the same way as in a local IDE. App Studio supports code completion, method signature, refactoring, and redirection for HTML, CSS, LESS, SCSS, JavaScript, TypeScript, JSX, and TSX files. In addition, you can develop front-end projects based on templates without the need to build any environment or download any dependency.

Create a backend project

1. Create a project based on the sample project.
   i. Go to the **App Studio** page and click **Projects** in the left-side navigation pane. On the Projects page, click **Create Project from Code**.
   ii. On the **Create Project** page, set the **Name** and **Description** parameters, and set the runtime environment to **springboot**.
      - The name of the project must start with a letter and can contain digits, letters, underscores (_), and hyphens (-).
      - The description of the project can be 2 to 500 characters in length.
   iii. After the configuration is completed, click **Submit**.
2. Set running parameters.

   In the upper-right corner, choose **Edit Config** > Edit Configurations. In the **Run/Debug Configurations** dialog box that appears, set the required parameter and then click **OK**.

Parameter	Description
Main class	Select the main method. If no main method is available, check whether your project has a main method.
VM options	The virtual machine (VM) options.
Program arguments	The app parameters.
Environment Variables	The environment variables.
JRE	The Java runtime environment (JRE). By default, this parameter cannot be modified.
PORT	The port of the ECS instance. Default value: *7001*.
ECS Instance	The instance type. Valid values: **1vCPU 2GMemory**, **2vCPU 3GMemory**, **4vCPU 8GMemory**, and **8vCPU 16GMemory**.
Pre-Launch Option	The commands to be run before the project is run. You can specify up to three commands.
Enable Hot Code	Specifies whether to enable hot code replacement.

You can click **Add** on the left of the Run/Debug Configurations dialog box to add multiple configurations for running.

3. Run the project.

   Click the Run icon in the upper-right corner to run the project.

   The first time that the project is run takes a longer time because App Studio needs to allocate the ECS instance and initialize the language service. After the running is completed, the Runtime tab appears, showing the access link.

4. Access the project.

   Click **Open Link** to access the project.

   Append /testapi to the link and refresh the page.

Understand App Studio

The following operations are supported for created projects:

- Top navigation bar
  - **Project**

    From the Project menu, you can configure the project or view detailed information by selecting **Character Set** or **Project Information**. Provided information about the current project includes the ID specified by **Project ID**, name specified by **Project Name**, type specified by **Project Type**, creation time specified by **Created At**, and **UUID**.

  - **File**

    From the File menu, you can create a file or open a recently created file by selecting **Create File** or **Re-Open Most Recent Files**.

  - **Edit**

    From the Edit menu, you can perform common editing operations. To search all the code in the project and open the related file, select Find in Path.

- **Version**

  From the Version menu, you can select **Switch Branch**, **View Changes**, **Submit**, **View Log**, **Connect to Remote Repo**, and **Merge Abort**.

  - **Switch Branch**

    In the Check Out Branch dialog box, you can click **+Create Branch** to create a local branch and push it to the remote repo. You can click a local branch and select **checkout** from the shortcut menu on the right to switch to the branch. You can also select **merge** to merge the selected branch to the current branch.

    You can click a remote branch and select **check out as a new local branch** from the shortcut menu on the right to check out the remote branch locally. Then rename the branch. You can also select **merge** to merge the selected branch to the current branch.

  - **View Changes**

    Click **View Changes** to view the list of edited files on a local branch in the right-side navigation pane.

  - **Submit**

    Click Submit to commit edits on a local branch for staging. You must enter the commit information.

  - **View Log**

    On the Log page, you can view all commit records of branches and filter them.

  - **Connect to Remote Repo**

    You can associate a new project with a remote repo for version control.

- **View**

  You can click **Toggle Full Screen** or press **Esc** on the keyboard to enter or exit the full screen mode of the page. You can also click **Hide Sidebar** or **Hide Status Bar** to hide the right-side navigation pane or the status bar. If they are hidden, you can click Show Sidebar or Show Status Bar to show them respectively.

- **Debug**

  - If you create a front-end project, you can set running parameters and add custom images.
  - App Studio supports Java-based debugging. In addition to setting running parameters and adding custom images, you can perform many other operations for debugging backend projects. You can also perform full or incremental builds and compile the Main.java file.

- **Settings**

  From the Settings menu, you can set the Git configuration to import the Git code to create a project. You can also configure your preference and shortcut keys.

- **Deploy**

  You can choose **Deploy > Download Source Code** to download the source code.

- **Template**

  You can choose **Template > Manage Templates** to go to the **My Templates** page to manage templates.

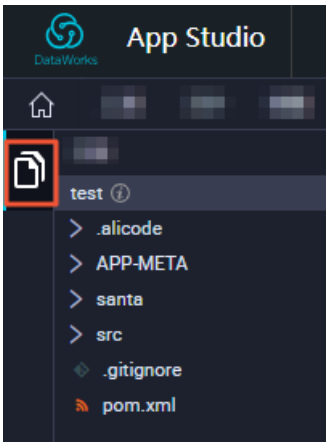
- Left-side navigation pane

- Entry

  Click the icon framed in red. The project section appears.

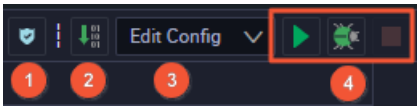
- Edit section

  Double-click a file that you want to edit. In the Edit section that appears, right-click the code section to perform the following operations.



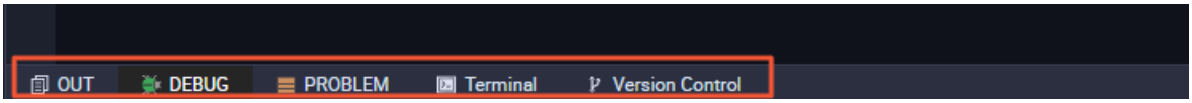
Action	Description
Go to Definition	Navigates to the definition page.
Peek Definition	Previews the definition.
Find All References	Searches for all references.
Workspace Symbol	Searches for a symbol in the project.
Go to Symbol...	Navigates to the symbol in the project.
Generate...	Generates the code.
Rename Symbol	Renames the symbol.
Change All Occurrences	Changes the name of all occurrences of a symbol throughout the file.
Format Document	Formats the file.
Cut	Cuts the file.
Copy	Copies the file.
Command Palette	Goes to the command palette.

- Icons in the upper-right corner



No.	Feature
1	Alibaba Coding Guidelines
2	Build Program. You can perform this operation only when the project is running or being debugged.
3	Run/Debug Configurations. You can set parameters for running or debugging the project.
4	Operations on the project, including running, debugging, or stopping the project.

- Bottom bar



  - OUT tab

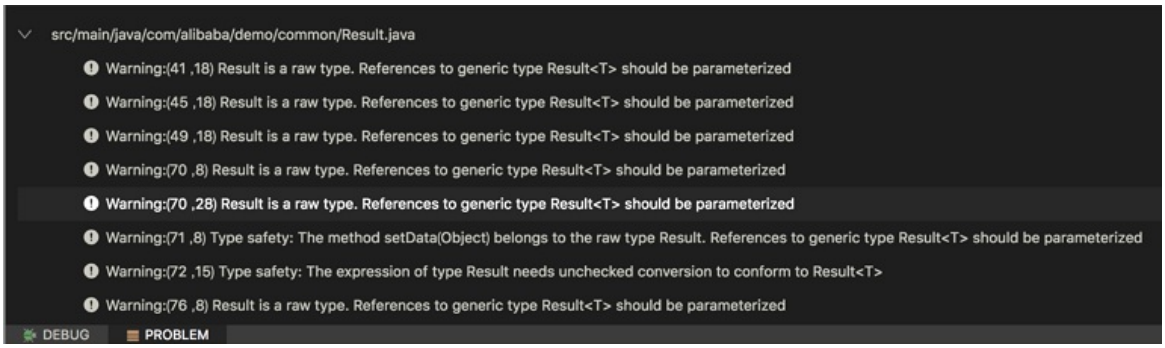
    You can click the OUT tab to view the output.

  - RUN or DEBUG tab

    If you click the Run or Debug icon for a project, this tab appears, showing the progress and information of the project.

  - PROBLEM tab

    If you click the Run or Debug icon for a project that has a problem, this tab appears.



- Terminal tab

  When running or debugging a project, you can click the Terminal tab and run bash or vim commands on the ECS instance.

- Version Control tab

  You can click the Version Control tab to view the logs and history of the project.

18.3. Navigation pane

18.3.1. View and manage projects

You can create and manage projects on the Projects page.

Go to the App Studio page and click Projects in the left-side navigation pane. On the page that appears, you can view projects that you have created. For more information about how to create template-based and code-based projects, see Project management.

Click a project to go to the project editing page. You can also click **Create Template** of a project to create a template based on the project.

Create a template

1. Click **Create Template** of a project.
2. In the **Create Template** dialog box that appears, set each parameter.

Parameter	Description
Name	The name of the template.
Description	The description of the template.
Class	The class of the template.

3. After the configuration is completed, click **OK**.

18.3.2. View and manage templates

You can view all templates created based on projects on the Templates page.

Click a template to go to the template details page. Then, click **Code Editor** to view the project code that this template is based on.

You can also click **Create Project** of a template to create a project based on this template.

18.4. Project management

This topic describes how to create and manage projects.

You can create a template-based or code-based project.

Create a template-based project

1. Go to the App Studio page and click Projects in the left-side navigation pane. On the **Projects** page, click **Create Project from Template**.
2. On the **Create Project** page, specify **Name** and **Description**, and select a template.

   > Note
   > - You can select a custom template or a template provided by the system.
   > - All projects created by using templates support WYSIWYG development.

3. After the configuration is completed, click **Submit**.

Create a code-based project

You can create a project by running code. App Studio provides code templates for three types of runtime environments. Select a code template as required.

1. Go to the App Studio page and click Projects in the left-side navigation pane. On the **Projects** page, click **Create Project from Code**.
2. On the **Create Project** page, specify **Name** and **Description**, and select a template.
3. After the configuration is completed, click **Submit**.

View and manage projects

You can view the created projects on the **Projects** page.

You can click a project name to go to the project editing page. You can also click **Create Template** of a project to create a template based on the project.

> **Note** You can view projects shared by others but cannot create templates based on those projects.

18.5. Code editing

18.5.1. Overview

Code editing supports common IDE features, such as automatic completion, code hinting, syntax diagnosis, and global content search.

The following tables list the basic and advanced features that App Studio supports in different languages.

Basic feature	Java	Python	JavaScript and TypeScript
Completion	Supported	Supported	Supported
Hover	Supported	Supported	Supported
Diagnostics	Supported	Supported	Supported
SignatureHelp	Supported	Supported	Supported
Definition	Supported	Supported	Supported
References	Supported	Supported	Supported
Implementation	Supported (coming soon)	Not supported	Not supported
DocumentHighlight	Supported	Supported	Supported
DocumentSymbol	Supported	Supported	Supported
WorkspaceSymbol	Supported	Supported	Supported
CodeAction	Supported (Alibaba Java Guidelines coming soon)	Supported	Supported
CodeLens	References implementation	Not supported	Not supported
Formatting	Supported	Supported	Not supported
RangeFormatting	Supported	Not supported	Not supported
FindInPath	Supported	Supported	Supported

Advanced feature	Java	Python	JavaScript and TypeScript
Rename	Supported	Supported	Supported
WorkspaceEdit	Supported	Not supported	Not supported
UnitTest (quick start)	Supported	Not supported	Not supported
MainClass	Supported	Not supported	Not supported
MainClassQuickStart	Not supported	Not supported	Not supported
ListModules	Supported	Not supported	Not supported

Advanced feature	Java	Python	JavaScript and TypeScript
Generate	Constructor Override Getter and Setter Implement	Not supported	Not supported

18.5.2. Generate code snippets

Currently, App Studio supports the Java class constructor, getter and setter methods, override methods of the parent class that a child class inherits, and API methods to be implemented.

Entry

Perform either of the following operations to generate the Java code:

- Right-click the code section and select **Generate**.
- Press Command+M on the keyboard. The Java code is automatically generated.

Constructor

On the Generate menu, click **Constructor**.

Select the fields to be included in the constructor and click **OK**.

The constructor that contains the initialization statement of the fields is generated.

Getter and setter methods

Generate the getter and setter methods in a way similar to the constructor.

> **Note** If a Java class does not have any field or the Java class is overwritten by the @data annotation of lombok, the getter or setter method is not required for the Java class. In this case, the Getter, Setter, and Getter And Setter options do not appear on the Generate menu.

Override methods

Click Override Methods on the Generate menu. All methods that can be overridden are listed in the Generate Code dialog box.

Select a method. The corresponding method is generated.

18.5.3. Run UT

App Studio currently supports unit testing (UT), including automatically generating UT code, detecting the entry for UT, running UT code, and displaying the UT result.

Automatically generate UT code

Open the target file, right-click the code editing section, select **Generate** and then click **Create Test**. The UT class file and UT code are automatically generated in the test directory.

Detect the entry for UT

> **Note**
>
> - UT class files must be stored in the *src/test/java* directory. A Java UT class file that is not stored in this directory cannot be identified as the Java UT class.
> - For a method annotated with @Test annotation, Run Test appears, indicating the entry for UT.

After the Java UT class file is created, add the **@Test** annotation of `org.junit.Test` to the corresponding sample UT method.

Run UT code

Click the Run icon in the upper-right corner. The sample UT starts.

18.5.4. Find in Path

App Studio provides the Find in Path feature to support global content search.

Move the pointer over **Edit** in the top navigation bar and select **Find in Path**.

You can select **Match Case**, **Words**, **Regex**, and **File Mask** as required. If you select File Mask, you must also select a file name extension from the right drop-down list to search in files of the specified type.

You can also search for content in the specified project, module, or directory.

After selecting a file, you can locate the searched content in the file and open the file in the editor.

18.6. Debugging

18.6.1. Configuration and startup

You can configure the entry method, start debugging, and set breakpoints to debug an app.

Configure the entry method

Parameter	Description
Main class	The entry method (which is the main method) you want to start. You can select a value from the drop-down list.
VM options	The parameters for starting a Java Virtual Machine (JVM), for example, -D, -Xms, and -Xmx.
Program arguments	The startup parameter, which is obtained by the args parameter in the main method.
Environment Variables	The environment variables.
JRE	The Java runtime environment. Default value: *1.8 - SDK*.

Parameter	Description
PORT	The port you want to expose in the app, for example, classic port 7001 or port 8080 for Spring Boot-based projects.
ECS Instance	The type of the ECS instance used for debugging.
Enable Hot Code	This configuration takes effect only in Run mode. By default, the HotCode2 plug-in that Alibaba Cloud provides is used.

Start debugging

Move the pointer over **Debug** in the top navigation bar and click **Start Debugging**.

The first startup is slower, because the system needs to prepare the runtime environment and download Maven dependencies for you. When you restart debugging, App Studio skips this process and provides user experience similar to that in a local IDE.

18.6.2. Online debugging

App Studio supports the online debugging of Java apps and Spring Boot-based web projects.

Before online debugging, you must configure the entry method and start debugging. For more information, see Configuration and startup.

Exposed services

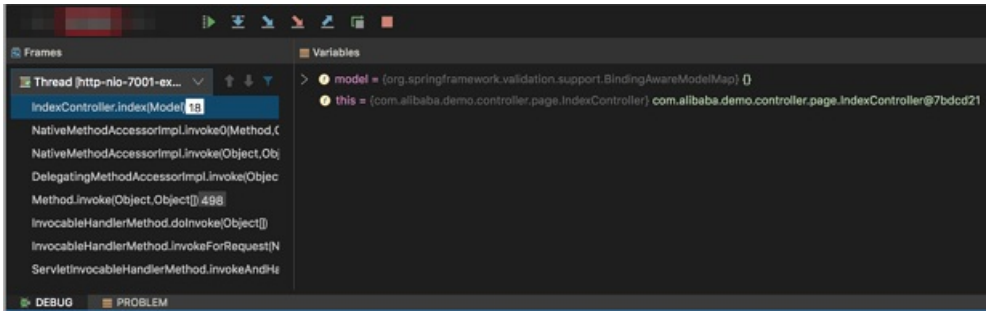
After your app is started, two basic services are provided. You can click the link next to Backend to debug the back-end Java code.

Panel introduction

- Output

  The Output panel displays the standard output, excluding System.in, of all apps. It supports the ANSI color and guarantees consistent experience as a local terminal.

- Call Stack

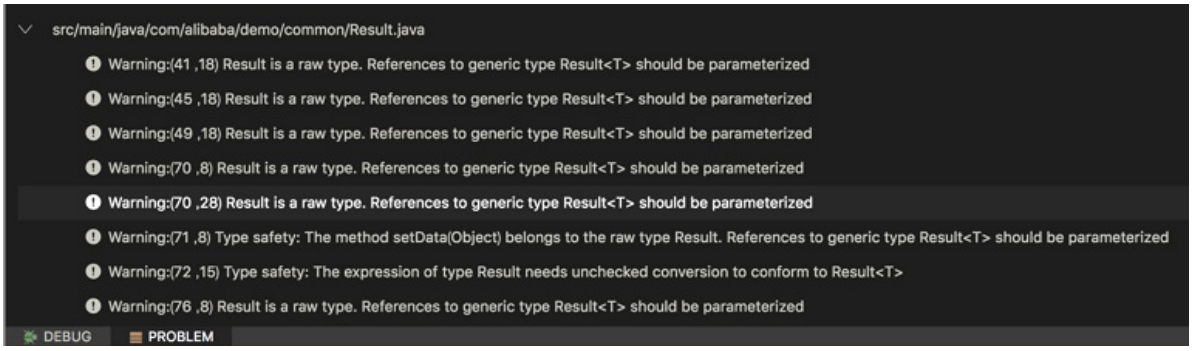


- Breakpoint

  The Breakpoint panel displays the breakpoints that are currently set. For more information about the breakpoint types and usage, see Breakpoint types.

- PROBLEM

The **PROBLEM** panel displays compilation problems of apps. You can click a record to go to the corresponding line in the file.



18.6.3. Breakpoint types

App Studio supports normal line breakpoints, method breakpoints, and exception breakpoints.

Normal line breakpoint

You can click the blank section next to a line in the current file to generate a breakpoint for that line. The breakpoint also appears on the Breakpoint panel.

Method breakpoint

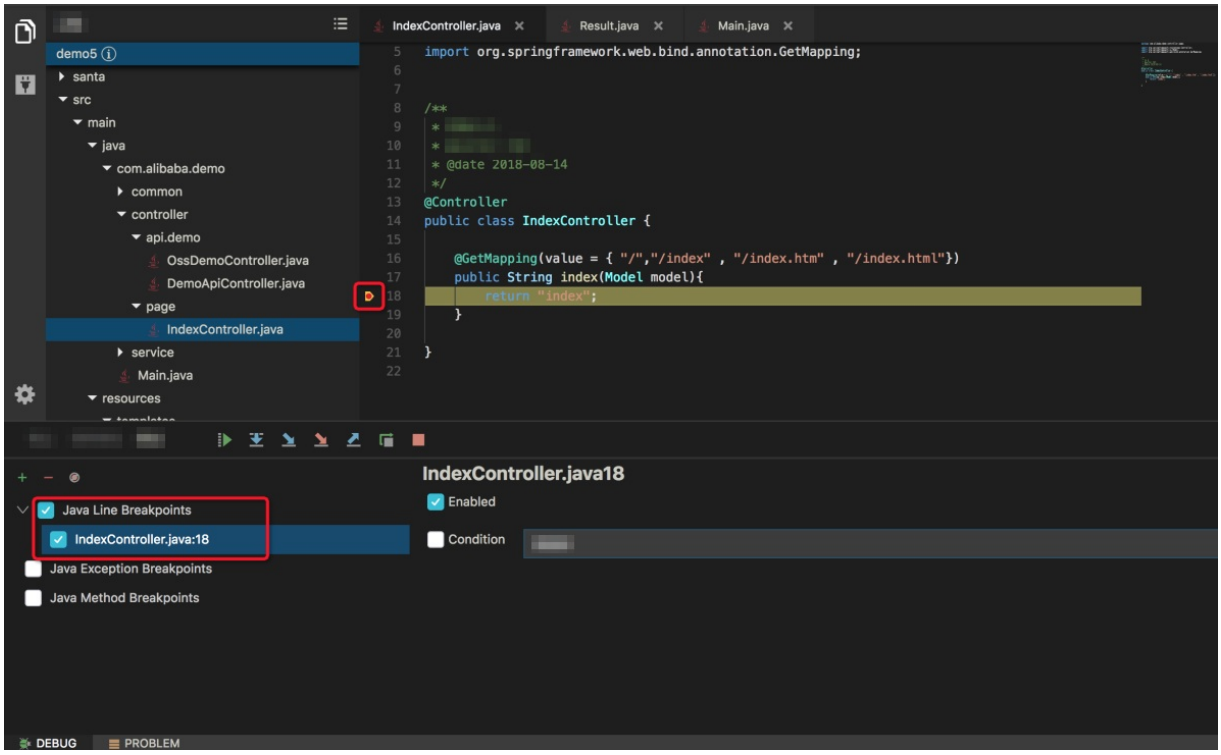
Different from a line breakpoint or an exception breakpoint, a method breakpoint triggers two events, namely, entry and exit. You can manually add a method breakpoint, or set a breakpoint at the place where the method is defined.

If the method breakpoint is triggered, the program stops when stepping into or out of the method.

Exception breakpoint

If an exception breakpoint is set, the program stops when encountering the exception.

As shown in the following figure, after index is triggered, the program stops in line 23 because **NullPointerException** appears.



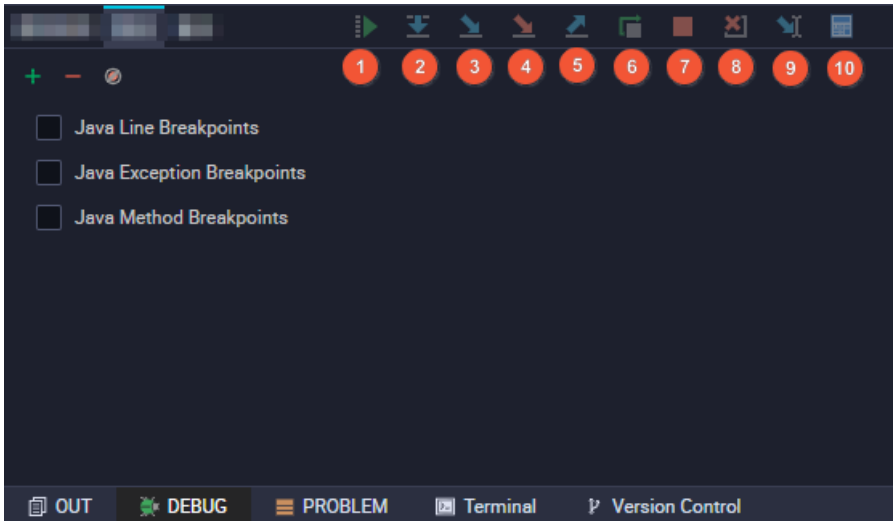
18.6.4. Breakpoint operations

The Breakpoint panel displays the breakpoints that are currently set. This topic describes how to operate breakpoints.

Breakpoints can be classified into normal line breakpoints, method breakpoints, and exception breakpoints. For more information, see Breakpoint types.

Debugging buttons

You can perform the debugging operations by clicking the following buttons listed in the table:



No.	Feature	Description
1	**Continue**	Resumes the current breakpoint to continue the current thread.
2	**Step Over**	Runs to the next line.
3	**Step Into**	Steps into a method.
4	**Force Step Into**	Forcibly steps into a method of a class not to be stepped into. Different from **Step Into**, **Force Step Into** enables you to step into a method from a built-in Java library.
5	**Step Out**	Steps out of the current method.
6	**Restart**	Currently, the **Restart** button is not perfect enough and may not be able to clean up the program. This button is being optimized.
7	**Stop**	Stops debugging.
8	**Drop Frame**	Deletes the current stack and returns to the previous method.
9	**Run to Cursor**	Runs to the current line of code. You can set a temporary breakpoint in a line.
10	**Evaluate Expression**	Calculates an expression.

18.6.5. Terminal

You can start multiple terminals in App Studio.

The **Terminal** tab appears in the lower part of the page.

App Studio supports common shell commands such as **ls** and **cat** and interactive commands such as **vi** and **top**.

18.6.6. Hot code replacement

Using the hot code replacement feature, you can edit the running code of an app and make the edits effective without restarting the app.

For example, after you edit the code while debugging a Spring Boot-based app, you do not need to restart the app. The edited code takes effect once it is saved. App Studio supports this feature by default.

App Studio also supports hot code replacement while an app is running. To trigger hot code replacement, you only need to save the file without installing any plug-in or manually compiling the file.

If you are editing the code in Debug mode, App Studio automatically deletes the current running stack and returns to the method entry.

Configure hot code replacement in Run mode

Enable hot code replacement on the Run/Debug Configurations page.

After you click Run or Debug, the output information of the HotCode2 plug-in appears on the OUT tab.

Save the file after editing it.

Configure hot code replacement in Debug mode

You can use the native Java Debug Interface (JDI) to enable hot code replacement in Debug mode. However, due to Java Virtual Machine (JVM) restrictions, hot code replacement is unavailable when a method is added to or deleted from a class. You can save the file to trigger hot code replacement.

> **Note** The native JVM supports hot code replacement for operations such as adding or deleting a class. However, hot code replacement is unavailable when you change the class structure.

18.7. WYSIWYG designer

18.7.1. Get started with the WYSIWYG designer

This topic describes basic operations in the WYSIWYG designer, including creating a project and building a visual page.

Create a project

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > App Studio**. The **Projects** page appears.
3. Click **Projects** in the left-side navigation pane. On the page that appears, click **Create Project from Code**.
4. On the **Create Project** page, set the **Name** and **Description** parameters, and set **Select the runtime environment** to **appstudio**.
5. After the configuration is completed, click **Submit**.

Build a visual page

Open a project created by using the WYSIWYG designer. Go to the *santa/pages* directory in your project.

Double-click a .santa file to go to the WYSIWYG designer. For example, you can double-click the file named *home.santa*.

You can also right-click **pages** and choose **Create > Template** to develop the page based on a template.

The WYSIWYG designer consists of the component menu and operation panel.

- Component menu

  The component menu lists all components that the WYSIWYG designer presets, including **layout components**, **basic components**, **form components**, **chart components**, and **advanced components**.

Select a component from the component menu and drag and drop it to the visual operation section. Click the component. The **Component Settings** panel appears on the right.

On the **Component Settings** panel, you can configure the component on the **Properties**, **Style**, and **Advance** tabs.

- Operation panel

  You can click the corresponding icon on this panel to **undo an operation**, **redo an operation**, **preview the rendering result**, **enable the code mode**, **use the global style**, **configure the navigation**, **configure a global data flow**, **deploy as a template**, and **save edits**.

  Click the **Configure Navigation** icon in the upper-right corner to go to the navigation configuration page. For more information, see Navigation configuration.

Configure a global data flow

For more information about how to configure a global data flow, see Global data flow.

On the Component Settings panel, you can configure the component on the **Properties**, **Style**, and **Advance** tabs.

- Configure component properties

  On the Properties tab, you can visually configure component properties.

  Based on the rules for configuring component properties, a visual form is generated on the Properties tab. After you configure component properties in this form, the WYSIWYG designer re-renders the component in the visual operation section based on the new properties. You can view the rendering results of the component with different properties in real time.

- Configure component styles

  On the Style tab, you can configure the styles of a component.

  A visual panel for configuring common styles is provided on the Style tab. On this panel, you can customize the basic styles of a component, including the layout, text, background, border, and effect.

  After you add or modify the component styles on this tab, the WYSIWYG designer collects all the style settings and re-renders the component in the visual operation section based on the new component style. You can view the component configuration effect in real time.

- Configure association between components

  On the Advanced Settings tab, you can configure association between components.

  Select a component in the visual operation section and click the **Advance** tab. The properties of the selected component are listed on the left of the tab. Click the Magnifier icon on the right and select the component to be associated to your selected component.

  The properties of the associated component appear on the right of the tab.

  Select a property, for example, searchParams, in the left property list and connect it to a property, for example, requestParams, in the right property list.

  In this way, any change of the searchParams parameter of the left component is transferred to the requestParams parameter of the right component in real time. This achieves property-based association between the two components.

Configure the code mode

By using the code mode, you can implement complex interactions in a more advanced way. For more information, see Code mode.

Save, preview, run, and hot code replacement

For more information, see Save, preview, run, and hot code replacement.

18.7.2. Code mode

By using the code mode, you can implement complex interactions in a more advanced way.

Click the **Code Mode** icon in the upper-right corner of the operation panel to enable the code mode.

The WYSIWYG designer uses domain-specific language (DSL) at the intermediate layer to switch between the visualization mode and code mode. DSL can be considered as a simplified version of React. The DSL syntax is basically the same as the React syntax.

As shown in the code section in the preceding figure, DSL uses a tag to describe a component. The tag properties are the component properties. The property value can be of a simple data type such as a string or a number. The property value can also be an expression. You can enter `state.xxx` to obtain data from the global data flow.

The code mode has the following features:

- If you drag and drop a component or configure the component properties in the visualization section, the edits are updated in the code in real time.
- If you edit the code in the code section, the edits are updated in the visualization section in real time.
- The drag-and-drop operation and component property configuration in the visualization section and code edits in the code section can be converted between each other.

18.7.3. DSL syntax

Domain-specific language (DSL) is a component-based language developed based on the features of React JSX and Vue templates and is more suitable for UI layout design.

JSX

The DSL syntax is similar to the JSX syntax in the React.render method. The following section provides a brief description of JSX:

- You can use `{ }` to switch an HTML scope to a JavaScript scope. In a JavaScript scope, you can write any valid JavaScript expression. The return value appears on the page, for example, `<div>{'Hello' + ' Relim'}</div>`.

  > **Note** You can write any JavaScript expressions such as computing statements or literals in `{ }`.

- An HTML tag is used to switch a JavaScript scope to an HTML scope, for example, `{<div>Hello Relim</div>}`.
- The HTML scope and JavaScript scope can be nested, for example, `{<div>{'Hello' + ' Relim'}</div>}`.

Valid JavaScript expressions

```
// Computing statements
{aaa} // √ Variable aaa must be defined.
{aaa * 111} // √
{1 == 1 ? 1 : 0} // √
{/^123/.test(aa)} // √
{[1,2,3].join('')} // √
{(()=>{return 1})()} // The self-executing function. √
// Literals
{1}
{true}
{[11,22,33]} // √
{{aa:"11",bb:"22"}} // √
{()=>1} // Describe a function, which is valid but meaningless. √
```

Note If certain complex logic must be implemented by multiple computing statements rather than only one statement, you can wrap the logic in a self-executing function, which must be a valid expression. The following statements provide an example:

```
{(function(){
    // Sum the even digits of a number array.
    var input = [1,2,3,4,5,6,7,8,9,10];
    var temp = input.filter(i => i % 2 == 0)
    return temp.reduce((buf, cur) => buf + cur, 0)
})()}
```

Invalid JavaScript expressions

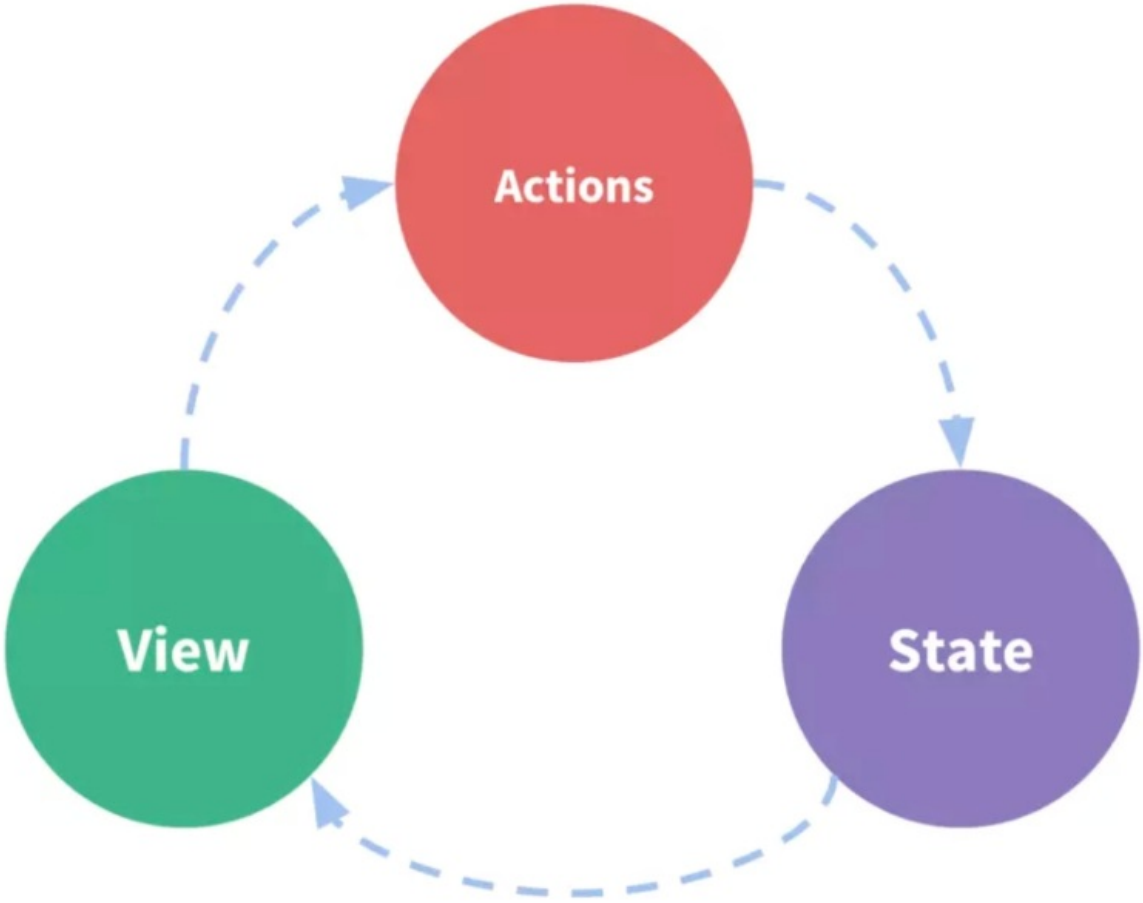
```
{ var a = 1 } // The value assignment statement.
{ aaa * 111; 2} // Multiple statements separated with semicolons (;).
```

18.7.4. Global data flow

A global data flow is used for front-end data management. For multiple components that need to share a state, it is difficult to transfer the state among them. To resolve this issue, you can extract the shared state and use a global data flow to transfer it to all related components.

Principles

In a global data flow, global data is transferred in a globally unique way. Once the data declared in global data changes, the data flow shown in the following figure is executed.



1. A component triggers an action when, for example, a user clicks the component.
2. The action triggers global data changes.
3. Upon the global data changes, components that reference the global state are automatically re-rendered.

Scenarios

A global data flow is applicable to the association of two or more components on a page. You can refine public data into global data for unified management, and then use a global data flow to associate two or more components.

Configure a global data flow

1. Click the **Global Data Flow Settings** icon in the upper-right corner of the operation panel.
2. In the **Global Data Flow Settings** dialog box that appears, set **Variable Name** and **Value**.
   - The variable value can be a number, character string, or JSON string.
   - If the variable value is declared as an API endpoint, data obtained from the API is automatically used as the value of the variable name.
3. Click **Save**.

Use a global data flow

- Obtain global data

  Use `state.name` in the component to obtain global data.

  ```
  <Input value={state.name} />
  ```

- Modify global data

  Use the $setState() method in the component to modify global data.

  ```
  <Input onChange={value => $setState({ name: value })} />
  ```

  Note You must use the $setState() method to modify global data. If you use `state.name = 'new value'`, re-rendering cannot be triggered.

18.7.5. Save, preview, run, and hot code replacement

In the WYSIWYG designer, you can perform operations such as saving edits, previewing the rendering result, running an app, or making edits in hot code replacement mode.

Save edits

The WYSIWYG designer periodically saves your edits. You can also click the **Save** icon in the upper-right corner of the operation panel to save edits.

Preview the rendering results

In the WYSIWYG designer, code in the operation section is in the editable status. However, special processing is added for the editable status of some components. For these components, you can run the rendering logic only when the app is running. To preview the rendering result, click the Preview icon in the upper-right corner of the operation panel.

Run an app

In the WYSIWYG designer, you can open and edit only one santa file at a time. To view the effect of the entire app,

click the Run Program icon on the Debug panel of App Studio to run the app.

Make edits in hot code replacement mode

If you are not satisfied with any page after running the app, you can edit the code in the WYSIWYG designer and save the edits.

The edited code takes effect on the running page in hot code replacement mode.

18.7.6. Navigation configuration

This topic describes how to configure the site navigation in the WYSIWYG designer.

The WYSIWYG designer provides each app with a public page header, a public bottom bar, and public sidebars, where you can configure various menus and themes. You can also specify whether to display the public header, bottom bar, and sidebars as required.

Click the **Navigation Settings** icon in the upper-right corner of the operation panel to go to the page for configuring the navigation of an app.

Configure the public header

You can configure the public header based on your business requirements.

Parameter	Description
Enabled	Specifies whether to display the public header.
Theme	The theme of the public header. You can select a dark or light theme.
Logo Image	The logo image of the site. You can enter an image URL or upload a local image.
Title	The title of the site.
Fix to Page Top	Specifies whether to fix the public header to the top of the page. If you turn on this switch, the public header stays at the top of the page when the page scrolls.
Menu Items	The menu items such as the link name and link URL that are displayed in the public header.

Configure the sidebars

You can configure the sidebars based on your business requirements.

Parameter	Description
Enabled	Specifies whether to display the sidebars.
Theme	The theme of the sidebars. You can select a dark or light theme.
Enable Folding	Specifies whether the sidebar menus can be hidden.

19.Migration Assistant

19.1. Overview

The Migration Assistant service of DataWorks allows you to migrate data objects across different DataWorks versions, Alibaba Cloud accounts, regions, and workspaces.

Migration Assistant allows you to export data objects in your workspace, including auto triggered nodes, manually triggered nodes, resources, functions, data sources, table metadata, ad hoc queries, and components. You can create full export tasks, incremental export tasks, or custom export tasks to export your data objects in DataWorks based on your business requirements.

> **Notice** To create export or import tasks, you must use an Alibaba Cloud account or be the workspace administrator. If you use a Resource Access Management (RAM) user that is not assigned the administrator role, you can only view export and import tasks.

Scenarios

- Back up node code

  You can use Migration Assistant to periodically back up your node code to prevent data from being deleted by mistake. In this case, we recommend that you create a full export task.

- Export a common workflow for replication

  You can use Migration Assistant to export a common workflow that can be replicated in other workspaces. In this case, we recommend that you create a custom export task.

- Build a test environment

  You can use Migration Assistant to copy all the node code and replace the production data with test data to build a test environment. In this case, we recommend that you create a full export task or a custom export task.

- Develop data in a hybrid cloud environment

  You can use Migration Assistant to migrate node code from Alibaba Cloud public cloud to Alibaba Cloud Apsara Stack to develop data in a hybrid cloud environment. We recommend that you create a custom export task. If the difference between Alibaba Cloud public cloud and Alibaba Cloud Apsara Stack is large, a compatibility problem may occur when data objects are migrated.

- Migrate data objects between the development environment and production environment

  If a workspace consists of the production environment and development environment that are completely isolated, you can use Migration Assistant to export nodes from the development environment and import them to the production environment for deployment.

19.2. Cloud tasks

19.2.1. Export tasks from open source engines

DataWorks allows you to migrate tasks from open source scheduling engines such as Oozie and Azkaban to DataWorks. This topic describes the requirements for the files to be exported.

Context

Before you import a task of an open source scheduling engine to DataWorks, you must export the task to your on-premises machine or Object Storage Service (OSS). For more information about the import procedure, see Import tasks of open source engines.

Export a task from Oozie

Requirements and structure of the package to be exported:

- Requirements

  The package must contain XML-formatted definition files and configuration files of a flow task. The package is exported in the ZIP format.

- Structure

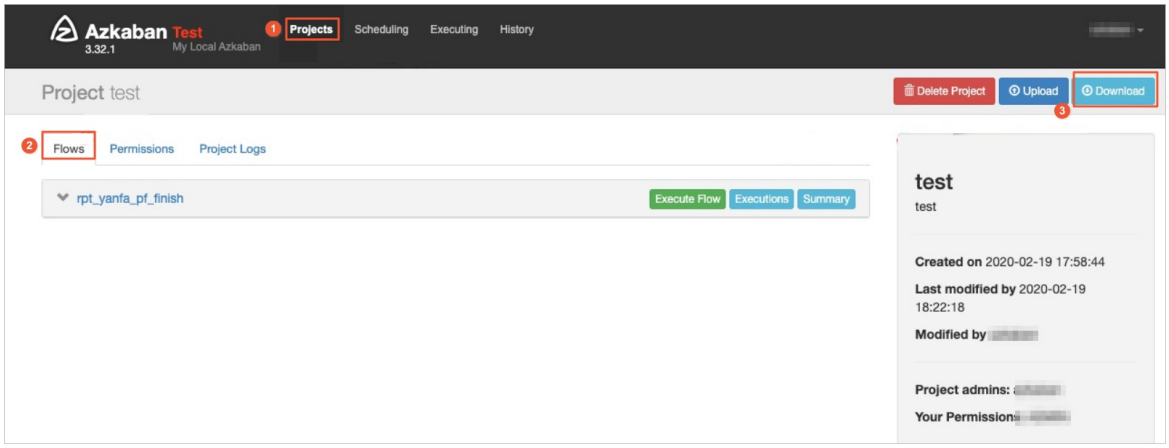
  Oozie task descriptions are saved in a Hadoop Distributed File System (HDFS) directory. For example, each subdirectory under the apps directory in the Examples package at the Apache Oozie official website is a flow task of Oozie. Each subdirectory contains XML-formatted definition files and configuration files of a flow task.

```
$tree
.
├── aggregator
│   ├── coordinator-with-offset.xml
│   ├── coordinator.xml
│   ├── job.properties
│   ├── job-with-offset.properties
│   ├── lib
│   │   └── oozie-examples-4.2.0.jar
│   └── workflow.xml
├── sqoop
│   ├── db.hsqldb.properties
│   ├── db.hsqldb.script
│   ├── job.properties
│   └── workflow.xml
├── cron
│   ├── coordinator.xml
│   ├── job.properties
│   └── workflow.xml
├── cron-schedule
│   ├── coordinator.xml
│   ├── job.properties
│   └── workflow.xml
```

Export a task from Azkaban

You can download a specific flow task in the Azkaban console.

1. Log on to the Azkaban console and go to the **Projects** page.
2. Select a project whose package you want to download. On the page for the project, click **Flows** to show all flow tasks under the project.
3. Click **Download** in the upper-right corner of the page to download the package of the project.



Native Azkaban packages can be exported. No limit is imposed on the packages of Azkaban. The exported package in the ZIP format contains information about all tasks and relationships under a specific project of Azkaban.

Export tasks from other open source engines

DataWorks provides a standard template for you to export the tasks of open source engines except for Oozie and Azkaban. Before you run an export task, you must download the standard template and modify the content to be exported based on the file structure in the template. You can go to the **Open Source engine export** page to download the standard template and view the file structure.

1. Log on to the DataWorks console.
2. On the DataStudio page, click the ☰ icon in the upper-left corner and choose **All Products > Other > Migration Assistant**.
3. In the left-side navigation pane, choose **Cloud tasks > Open Source engine export** to go to the **Open Source engine export scheme selection** page.
4. Click the **Standard Template** tab.
5. On the **Standard Template** tab, click **standard format Template** to download the template.
6. Modify the content to be exported based on the template and generate a package to be exported.

19.2.2. Import tasks of open source engines

This topic describes how to import tasks that are exported from open source engines into DataWorks.

Procedure

1. Go to the **Open Source engine import** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the ☰ icon in the upper-left corner and choose **All Products > Other > Migration Assistant**.

iii. In the left-side navigation pane, choose **Cloud tasks > Open Source engine import**.

2. Create an import task.

i. On the **Import Tasks** page, click **Create Import Task** in the upper-right corner.

ii. In the **Create Import Task** dialog box, configure the parameters as required.

Parameter	Description
Name	The name of the import task. The name can contain only letters, digits, underscores (_), and periods (.).
Engine type	The engine type. Valid values: **Azkaban**, **Oozie**, and **Standard format**.
Upload From	The source of the package that you want to import. Valid values: **Local** and **OSS**. ■ If you select **Local** for this parameter, perform the following steps to upload a package on your machine: a. Click **Upload File**. b. Select the package that you want to upload and click **Open**. c. Click **Check**. d. After the message **The resource package has passed the check** appears, verify that the file format and content are correct. **Note** The size of the package that you want to upload cannot exceed 30 MB. If the size of the package exceeds 30 MB, select OSS for this parameter. ■ If you select **OSS** for this parameter, enter the endpoint of an Object Storage Service (OSS) object in the **OSS Endpoint** field. Then, click **Check** and **Preview** in sequence to check and preview the package that you want to upload.
Remarks	The description of the import task.

iii. Click **OK**. The **Edit import task** page appears.

3. Edit the import task.

i. On the **Edit import task** page, specify **Import objects**.

Periodic tasks is selected for **Import objects** by default. If you want to import data objects of another type, select the required value from the **Import objects** drop-down list.

ii. (Optional)Click **Advanced Settings**. In the **Advanced Settings** dialog box, configure the mappings between the compute engine instances and the node types and click OK.

If multiple compute engine instances are bound to the destination workspace, you must complete the settings in the **Advanced Settings** dialog box. You can configure the mappings between compute engine instances and nodes of the Shell, Hive, and Sqoop types.

iii. On the Edit import task page, click **start import** in the upper-right corner.

4. View the import report.

i. In the **Import progress** dialog box, confirm the import task progress.

ii. After the import task is completed, click **Return to import task list**.

iii. Find the task on the **Import Tasks** page and click **View Import Report** in the Actions column. On the page that appears, view the task information in the **Basic Information**, **Import Settings**, **Import results**, and **Details** sections.

19.3. Migrate data objects in DataWorks

19.3.1. Create and view export tasks

Migration Assistant allows you to export data objects in your workspace, including auto triggered nodes, manually triggered nodes, resources, functions, table metadata, data sources, components, and ad hoc queries. This topic describes how to create and view export tasks.

Prerequisites

To create export or import tasks, you must use an Alibaba Cloud account or be the workspace administrator. If you use a Resource Access Management (RAM) user that is not assigned the administrator role, you can only view export and import tasks.

Context

Migration Assistant allows you to export data objects in different modes. These modes include full export, incremental export, and custom export. You can choose an export mode that best suits your business scenario.

- Full export tasks are used to export all the data objects in a workspace. For example, you can run a full export task to back up node code or clone the workspace to a test environment. When you run a full export task, data objects of the latest version are exported.

  Only saved data objects can be exported. If a node is saved in both the development environment and production environment, the node saved in the development environment is exported.

- Incremental export tasks are used to export data objects that were modified after the specified date.

  > **Note** You cannot configure a blacklist for incremental export tasks.

- Custom export tasks are used to export data objects that you specify. For example, you can run a custom export task to extract a common workflow and clone it to other workspaces. If a workspace runs in both a production environment and a development environment that are completely isolated from each other, you can run a custom export task to export nodes from the development environment and import them to the production environment for deployment.

Go to the Migration Assistant page

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Other > Migration Assistant**. The **DataWorks export** page under **DataWorks migration** appears.

Create a full export task

1. On the **Export Tasks** page, click **Create Export Task** in the upper-right corner.
2. In the **Create Export Task** dialog box, configure the parameters as required.

Parameter	Description
Name	The name of the export task. The name can contain only letters, digits, underscores (_), and periods (.).
Type	The type of the export task. Select **Full export** for this parameter. A full export task is used to export all the auto triggered nodes, manually triggered nodes, table metadata, and data sources that have been saved or committed in the current workspace.
Blacklist	Specifies whether to enable the blacklist feature for full export based on your business needs. If you select the **Add to Blacklist** check box, you can add the nodes and resources that do not need to be exported to a blacklist.
Export Version	Valid values: **Standard**, **Private Cloud(>=V3.12)**, and **Private Cloud(V3.6.1-V3.1.1)**. The DataWorks version determines the format in which data objects are exported. Check the DataWorks version of the destination workspace before you create an export task.
Remarks	The description of the export task.

3. (Optional)Click Add to Blacklist and run the export task.

   If you select **Add to Blacklist**, perform the following steps to configure the blacklist:

   i. In the **Create Export Task** dialog box, click **Add to Blacklist**.

   ii. On the **Set Blacklist** page, select the data objects that you do not want to export.

   iii. Click **Add Selected to Blacklist**.

   iv. Click **Export** in the upper-right corner.

   v. In the **Export confirmation** message, click **Confirm**.
4. (Optional)If you do not select **Add to Blacklist**, click **Export** in the Create Export Task dialog box.
5. In the **Export Progress** dialog box, view the progress of the export task. After the task succeeds, click **Back to Export Tasks**.

Create an incremental export task

1. On the **Export Tasks** page, click **Create Export Task** in the upper-right corner.
2. In the **Create Export Task** dialog box, configure the parameters as required.

Parameter	Description
Name	The name of the export task. The name can contain only letters, digits, underscores (_), and periods (.).

Parameter	Description
Type	The type of the export task. Select **Incremental** for this parameter. An incremental export task is used to export data objects that were modified after the specified date. Supported data objects include auto triggered nodes, manually triggered nodes, table metadata, and data sources that have been saved or committed in the current workspace.
Start Date	The date on which data is modified. The data generated after this date is incremental data.
Export Version	Valid values: **Standard**, **Private Cloud(>=V3.12)**, and **Private Cloud(V3.6.1-V3.1.1)**.
Remarks	The description of the export task.

3. Click **Export**.

Create a custom export task

1. On the **Export Tasks** page, click **Create Export Task** in the upper-right corner.
2. In the **Create Export Task** dialog box, configure the parameters as required.

Parameter	Description
Name	The name of the export task. The name can contain only letters, digits, underscores (_), and periods (.).
Type	The type of the export task. Select **Custom** for this parameter. A custom export task is used to export data objects that you specify. Supported data objects include auto triggered nodes, manually triggered nodes, table metadata, and data sources that have been saved or committed in the current workspace.
Export Version	Valid values: **Standard**, **Private Cloud(>=V3.12)**, and **Private Cloud(V3.6.1-V3.1.1)**.
Remarks	The description of the export task.

3. Click **Select Export Objects**.
4. On the **Export Objects** page, select a type of data object that you want to export from the **Export Object** drop-down list.

   Valid values of **Export Object**: **Table**, **Periodic tasks**, **Resources**, **Manual tasks**, **Function**, **DATA_SERVICE**, **Data source**, **Components**, and **Temporary query**.
5. Select the data objects that you want to export and click **Add Selected to Export Package**.

   You can also configure filter conditions such as **Export Object**, **Object Type**, and **Export Environment** to search for data objects. Then, click **Add All to Export Package** to add all the data objects that have been found to the package that you want to export.
6. Click **Export** in the upper-right corner.

View and manage export tasks

On the **Export Tasks** page, you can view the name, type, creator, status, update time, and description of created export tasks. The operations that you can perform on export tasks vary based on their status.

- If an export task is in the **Successful** state, you can perform the following operations on the task:
  - Click **View Export Report** in the Actions column. On the page that appears, view the task information in the **Basic Information**, **Overview**, and **Details** sections.
  - Click **Download** in the upper-right corner to download the package of the export task to a local directory.
  - Clone the export task.
    - Full export task for which the blacklist feature is not enabled: Click **Clone** in the **Actions** column. In the **Clone** dialog box, specify **Name** and click **Export**.
    - Full export task for which the blacklist feature is enabled: Click **Clone** in the **Actions** column. In the **Clone** dialog box, specify **Name** and click **Add to Blacklist**.

      On the **Set Blacklist** page, select the data objects that you do not want to export, click **Add Selected to Blacklist**, and then click **Export** in the upper-right corner.
    - Custom export task: Click **Clone** in the **Actions** column. In the **Clone** dialog box, specify **Name** and click **Select Export Objects**.

      On the **Export Objects** page, select the data objects that you want to export, click **Add Selected to Export Package**, and then click **Export** in the upper-right corner.
- If an export task is in the **Export failed** state, you can click **View Export Package**, **Download Export Package**, or **Re-export** in the Actions column as required. To retry the export task, click **Re-export**.
- If an export task is a custom export task that is in the **Editing** state, you can perform the following operations on the task:
  - Click **Edit** in the Actions column. On the Export Objects page, modify the data objects that you want to export.
  - Click **View Export Package** in the Actions column. On the **Export Package Details** page, view the task information in the **Basic Information**, **Overview**, and **Details** sections.
  - Click **Delete** in the Actions column. In the **Delete** message, click **Ok** to delete the task.
- If an export task is a full export task that is in the **Editing** state, you can click **Edit**, **Delete**, or **View Blacklist** in the Actions column as required. If you click View Blacklist in the Actions column, you can check the blacklist and click **Export** to run the export task or click **Close** in the dialog box that appears.

19.3.2. Create and view import tasks

After you run an export task to export data objects from a workspace, you can create an import task to import these data objects to a specified workspace.

Prerequisites

To create export or import tasks, you must use an Alibaba Cloud account or be the workspace administrator. If you use a Resource Access Management (RAM) user that is not assigned the administrator role, you can only view export and import tasks.

Go to the Migration Assistant page

1. Log on to the DataWorks console.
2. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Other > Migration Assistant**. The **DataWorks export** page under **DataWorks migration** appears.

Create an import task

1. In the left-side navigation pane of Migration Assistant, choose **DataWorks migration > DataWorks import**.
2. On the **Import Tasks** page, click **Create Import Task** in the upper-right corner.
3. In the **Create Import Task** dialog box, configure the parameters as required.

Parameter	Description
Name	The name of the import task. The name can contain only letters, digits, underscores (_), and periods (.).
Upload From	The source of the package that you want to import. Valid values: **Local** and **OSS**. ◦ If you select **Local** for this parameter, perform the following steps to upload a package on your machine: a. Click **Upload File**. b. Select the package that you want to upload and click **Open**. c. Click **Check**. d. After the message **The resource package has passed the check** appears, click **Preview**. On the page that appears, check the package that you want to import. **Note** The size of the package that you want to upload cannot exceed 30 MB. ◦ If you select **OSS** for this parameter, enter the endpoint of an Object Storage Service (OSS) object in the **OSS Endpoint** field. Then, click **Check** and **Preview** in sequence to check and preview the package that you want to upload.
Remarks	The description of the import task.

4. Click **OK**. The **Import Task Settings** page appears.

   Make sure that you have checked the format and content of the package before you click **OK**.
5. Configure the import task.

   When you configure the import task, you must complete the settings in the **Engine Instance Mapping** section. The settings in other sections are optional and can be configured as required.

   (Optional)

i. In the **Engine Instance Mapping** section, select a compute engine of the destination workspace for each compute engine that is bound to the source workspace.

If multiple compute engines are bound to the source workspace and only one compute engine is bound to the destination workspace, the import task fails. This is because some types of nodes cannot be created in the destination workspace due to the absence of the required compute engines.

ii. (Optional)In the **Resource Group Mapping** section, configure resource group mapping between the source and destination workspaces. This ensures that resource groups are available for running imported nodes.

iii. (Optional)In the **Dependency Mapping** section, configure workspace mapping for relevant nodes.

Some nodes use the name of the source workspace in their code. In this case, you must configure workspace mapping for the nodes to run properly after they are imported. Set the **New Workspace** parameter to the name of the destination workspace. The system uses this workspace name to replace the original workspace name in the node code and the names of the ancestor and descendant nodes of the current node. After the import task is complete, the original workspace name is replaced with the new workspace name.

iv. (Optional)In the **Dry-run** section, find the destination node that you want to set as a dry-run node and click **Set up empty run** in the Actions column.

You can also select multiple nodes and click **Batch Configure** to set these nodes as dry-run nodes.

This configuration is used to configure the scheduling mode of auto triggered nodes. The auto triggered node that is set as a dry-run node returns a success response without running and does not generate data.

v. (Optional)In the **Commission Rules** section, configure the commission rules for **Resources**, **Tables**, and **Functions**, and specify whether to enable **Change Owner** as required.

> **Note**
>
> - If a data object with the same name as the data object you want to import exists in the destination workspace, the imported data object cannot be committed.
> - If you select No for the Change Owner parameter and no owner is specified for the node you want to import, you are automatically configured as the owner of the node after it is imported.

6. Click **Import** in the upper-right corner.
7. In the **Confirm** message, click **OK**.

View and manage import tasks

On the **Import Tasks** page, the operations that you can perform on import tasks vary based on their status.

- After an import task is complete, you can view the details of the task. To view the task details, find the task on the **Import Tasks** page and click **View Import Report** in the Actions column. On the page that appears, view the task information in the **Basic Information**, **Import Settings**, **Import results**, and **Details** sections.
- If an import task is in the **Editing** state, you can perform the following operations on the task:

- Click **Edit** in the Actions column. On the **Import Task Settings** page, modify the task configurations.
- Click **Preview** in the Actions column. On the page that appears, view the task information in the **Basic Information**, **Overview**, and **Details** sections.
- Click **Delete** in the Actions column. In the message that appears, click **Ok** to delete the task.

- If an import task is in the **Import failed** state, you can click **Re-import** in the Actions column of the task. In the **Import progress** dialog box, click **Return to import task list** after the import task is complete.

20.Workspace management

20.1. Configure a workspace

On the Workspace Management page of a workspace, you can configure and manage the workspace. DataWorks supports a variety of compute engines, such as MaxCompute, E-MapReduce (EMR), Realtime Compute for Apache Flink, Hologres, Graph Compute, AnalyticDB for PostgreSQL, and AnalyticDB for MySQL.

Go to the Workspace Management page

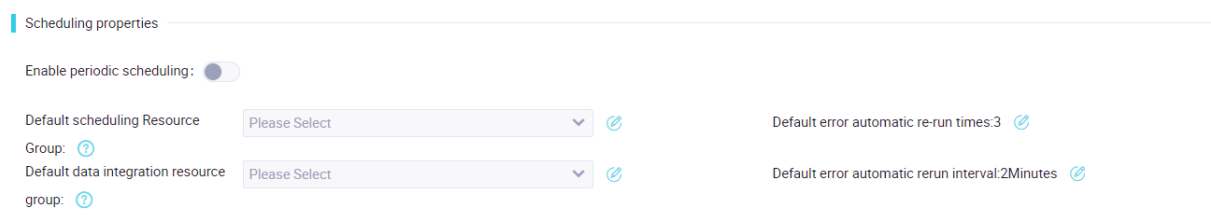
1. Log on to the DataWorks console.
2. On the **DataStudio** page, click the icon in the upper-right corner.
3. On the **Workspace Management** page, set the parameters in the **Basic properties**, **Scheduling Properties**, **Security Settings**, and **Compute Engine Information** sections for the workspace based on your business requirements.

Basic information

Parameter	Description
Workspace ID	The ID of the workspace, which cannot be changed.
Workspace Name	The name of the workspace. The name is not case-sensitive, can contain letters and digits, and must start with a letter. The name uniquely identifies the workspace and cannot be changed after the workspace is created.
Status	The status of the workspace.
Display Name	The display name of the workspace. The display name can contain letters and digits. You can change it based on your business requirements.
Creation Time	The time when the workspace is created. The value cannot be changed.
Mode	The mode of the workspace, which cannot be changed. Valid values: **Simple Mode** and **Standard**.
Owner	The owner of the workspace, which cannot be changed.
Description	The description for the workspace. You can modify the description based on your business requirements. The description can be up to 128 characters in length and can contain letters, special characters, and digits.

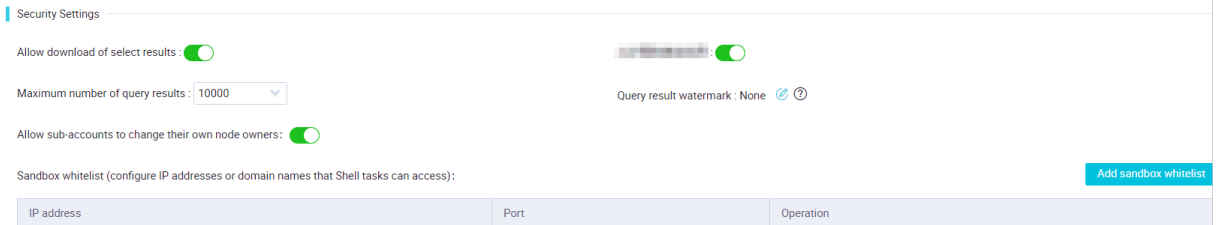
Scheduling properties

In the **Scheduling Properties** section, you can turn on or off the Periodic Scheduling switch for the workspace. You can also configure the **Default Scheduling Resource Group**, **Default Data Integration Resource Group**, **Default Automatic Rerun Times Upon Error**, and **Default Automatic Rerun Interval Upon Error** parameters for the workspace.



Nodes can be periodically run in a workspace only after you turn on **Periodic scheduling** for the workspace.

Security settings



Parameter	Description
Download SELECT Query Result	Specifies whether the query results that are returned by SELECT statements in DataStudio can be downloaded. If you turn off this switch, the query results cannot be downloaded.
Copy Query Result	Specifies whether the query results that are returned in DataStudio can be copied.
Maximum Number of Query Results	The maximum number of data records that can be returned for each query. Valid values: **10**, **100**, **500**, **1000**, **5000**, and **10000**. Default value: **10000**. For example, you set the **Maximum Number of Query Results** parameter to **1000**. Go to the **DataStudio** page 5 minutes later. In the left-side navigation pane, click **Ad-Hoc Query**. Create an SQL node and execute a query statement on the node to query data in a table that has more than 1,000 data records. The number of returned records is 1,000.
Watermark for Query Results	Specifies whether watermarks for the query results are displayed.
Change Node Owner by RAM User	Specifies whether RAM users can be used to change the owners of their nodes.
Sandbox Whitelist (contains IP addresses and domain names that can be accessed by Shell nodes)	The IP addresses or domain names that can be accessed by a Shell node that runs on the shared resource group.

To add an IP address or domain name to the whitelist, perform the following steps:

1. In the **Security Settings** section, click **Add**.
2. In the **Add** dialog box, enter an IP address or a domain name in the **Address** field and a port number in the **Port** field.
3. Click **Confirm**.

Associate a MaxCompute compute engine instance with a workspace

In the **Computing Engine Information** section, click the **MaxCompute** tab. On this tab, you can view the settings of the **MaxCompute Project Name** and **MaxCompute Visitor Identity** parameters for an associated MaxCompute compute engine instance.

Associate an EMR compute engine instance with a workspace

> **Note** If Kerberos authentication is enabled for an EMR cluster., you cannot create tables, resources, and functions in a visualized manner for this cluster.

1. In the **Compute Engine Information** section, click the **E-MapReduce** tab. On this tab, you can view the information about all EMR compute engine instances that are associated with the workspace.
2. Click **Add Instance**.
3. In the **New EMR cluster** dialog box, configure the parameters.

Parameter	Description
Instance Display Name	The display name of the EMR compute engine instance.
Region	The region of the current workspace.
Access Mode	The access mode of the EMR cluster. Valid values: **Shortcut mode** and **Security mode**
Scheduling access identity	The identity that is used to run the code of committed EMR compute engine nodes. Valid values: **Alibaba Cloud primary account** and **Alibaba Cloud sub-account**. .
Cluster ID	The ID of the user who created the EMR cluster.
Project ID	The ID of the project in the EMR cluster.
YARN resource queue	The name of the YARN resource queue in the EMR cluster. Unless otherwise specified, set this parameter to *default*.
Endpoint	The endpoint of the EMR cluster. You can obtain the endpoint in the EMR console.
Resource Group	The resource group that you want to use. Select a resource group based on your business requirements. After you select a resource group, click **Test Connectivity**.

4. After the connectivity test is passed, click **Confirm**.

Associate a Realtime Compute for Apache Flink compute engine instance with a workspace

1. In the **Compute Engine Information** section, click the **Real-time Computing** tab. On this tab, you can view the information about all Realtime Compute for Apache Flink compute engine instances that are associated with the workspace.
2. Click **Add Instance**.
3. In the **Add a real-time computing instance** dialog box, configure the parameters.

Parameter	Description
Instance Display Name	The display name of the Realtime Compute for Apache Flink compute engine instance.
Select Project	The Realtime Compute for Apache Flink project that you want to associate with the workspace as the compute engine instance. Select a project from the drop-down list. If you want to create a project, click **Real-time calculation control platform**.

4. Click **Confirm**.

Associate a Graph Compute compute engine instance with a workspace

1. In the **Compute Engine Information** section, click the **GraphCompute** tab.
2. Click **Bind Graph Compute Instance**.

> **Notice** A Graph Compute instance can be associated with only one DataWorks workspace. After a Graph Compute instance is associated with a DataWorks workspace, the instance cannot be associated with other DataWorks workspaces.

3. In the **Bind Graph Compute Instance** dialog box, configure the parameters.

Parameter	Description
Instance Display Name	The display name of the Graph Compute instance.
Graph Compute Instance Name	The name of the Graph Compute instance that you want to associate with the workspace as the compute engine instance.

4. Click **Bind**.

Associate a Hologres compute engine instance with a workspace

1. In the **Compute Engine Information** section, click the **Hologres** tab. On this tab, you can view the information about all Hologres compute engine instances that are associated with the workspace.
2. Click **Bind Hologres Database**.
3. In the **Bind Hologres Database** dialog box, configure the parameters.

Parameter	Description
Instance Display Name	The display name of the Hologres compute engine instance.
Access identity	The identity that is used to run the code of committed Hologres nodes. Valid values: **Alibaba Cloud primary account** and **Alibaba Cloud sub-account**.
Hologres instance name	The name of the Hologres instance that you want to associate with the workspace as the compute engine instance.
Database name	The name of the database that is created in **SQL Console**, such as testdb.

4. Click **Test Connectivity**.
5. After the connectivity test is passed, click **Confirm**.

Associate an AnalyticDB for PostgreSQL compute engine instance with a workspace

1. In the **Compute Engine Information** section, click the **AnalyticDB for PostgreSQL** tab.
2. Click **Add Instance**.
3. In the **Add AnalyticDB for PostgreSQL Instance** dialog box, configure the parameters.

Parameter	Description
Instance Display Name	The display name of the AnalyticDB for PostgreSQL compute engine instance. The display name must be unique.
InstanceName	The name of the AnalyticDB for PostgreSQL instance that you want to associate with the workspace as the compute engine instance.
DatabaseName	The name of the AnalyticDB for PostgreSQL database that you want to associate with the workspace.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.
Connectivity Test	AnalyticDB for PostgreSQL nodes must be run on exclusive resource groups. Therefore, you must specify an exclusive resource group. Click **Test Connectivity** to test the connectivity between the specified exclusive resource group and AnalyticDB for PostgreSQL instance.

4. After the connectivity test is passed, click **Confirm**.

Associate an AnalyticDB for MySQL compute engine instance with a workspace

1. In the **Compute Engine Information** section, click the **AnalyticDB for MySQL** tab.
2. Click **Add Instance**.

3. In the **Add an AnalyticDB for MySQL instance** dialog box, configure the parameters.

Parameter	Description
Instance Display Name	The display name of the AnalyticDB for MySQL compute engine instance. The display name must be unique.
InstanceName	The name of the AnalyticDB for MySQL cluster that you want to associate with the workspace as the compute engine instance.
DatabaseName	The name of the AnalyticDB for MySQL database that you want to associate with the workspace.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.
Connectivity Test	AnalyticDB for MySQL nodes must be run on exclusive resource groups. Therefore, you must specify an exclusive resource group. Click **Test Connectivity** to test the connectivity between the specified exclusive resource group and the AnalyticDB for MySQL cluster.

4. After the connectivity test is passed, click **Confirm**.

Associate a CDH compute engine instance with a workspace

1. In the **Compute Engine Information** section, click the **CDH** tab.
2. Click **Add Instance**.

   The development environment is isolated from the production environment if the workspace is in standard mode. If the workspace you use is in standard mode, you must add compute engine instances to both the development environment and the production environment.
3. In the **Add CDH Compute Engine** dialog box, configure the parameters.

   You can set the Access Mode parameter to **Shortcut mode** or **Security mode**. If **Security mode** is selected, the permissions on the data of the nodes that are run by using different Alibaba Cloud accounts or RAM users can be isolated.

   Configure the following parameters.

Parameter	Description
Instance Display Name	The display name of the CDH compute engine instance. The display name must be unique.

Parameter	Description
Access Mode	◦ The access mode of the CDH cluster. If **Shortcut mode** is selected, multiple Alibaba Cloud accounts or RAM users map to the same CDH cluster account. These Alibaba Cloud accounts or RAM users can access data in the same CDH cluster account. In this case, data permissions are not isolated. ◦ If **Security mode** is selected, you can configure mappings between Alibaba Cloud accounts or RAM users and CDH cluster accounts to isolate the permissions on the data of the nodes that are run by using the Alibaba Cloud accounts or RAM users.
Select Cluster	◦ If you set the **Access Mode** parameter to **Shortcut mode**, you must select a CDH cluster whose Authentication Type is not set to Kerberos Account Authentication. If no CDH cluster is available, you must create a CDH cluster. ◦ If you set the **Access Mode** parameter to **Security mode**, you must select a CDH cluster whose Authentication Type is set to Kerberos Account Authentication. You can check whether Kerberos Account Authentication is enabled for the CDH cluster in the DataWorks console. On the **Workspace Management** page, click **Hadoop Config** in the left-side navigation pane and find the cluster of which you want to view the configuration. Then, click **Modify** to view the setting of the **Authentication Type** parameter in the **Mapping Configuration** section. If no CDH cluster is available, you must create a CDH cluster.

Parameter	Description
AccessKey ID	○ **Shortcut mode**: The Authentication Type parameter is set to No Authentication by default. You can use only the admin or hadoop account. These accounts are used only to commit nodes. ○ **Security mode**: ■ You can set **Account for Scheduling Nodes** based on your business requirements. This account is used to automatically schedule and run the node after the node is committed to the scheduling system. You need to configure the mappings between the Alibaba Cloud accounts or RAM users and CDH cluster accounts. Valid values: **Task Owner**, **Alibaba Cloud primary account**, and **Alibaba Cloud sub-account**. Note ■ This parameter is available only for the production environment. ■ On the DataStudio page, the identity used to run nodes is the CDH cluster account that is mapped to the Alibaba Cloud account or RAM user used to log on to the console. You must configure the identity mappings for scheduling access identities and for workspace developers to prevent nodes from failing to run. ■ The default value of this parameter for the development environment is **Task owner**.
Exclusive Resource Group for Scheduling	Select an exclusive resource group for scheduling that connects to the DataWorks workspace. If no exclusive resource group for scheduling is available, you must create an exclusive resource group for scheduling. After you select an exclusive resource group for scheduling, click **Test Connectivity** to test the connectivity between the exclusive resource group for scheduling and the CDH cluster.

4. After the connectivity test is passed, click **Confirm**.

20.2. Workspace modes

DataWorks supports workspaces in basic mode and standard mode. You can implement different levels of security control on production data by using workspaces in different modes. This topic describes the differences between the two types of workspaces and the types of accounts or roles that can be used to access each type of workspace.

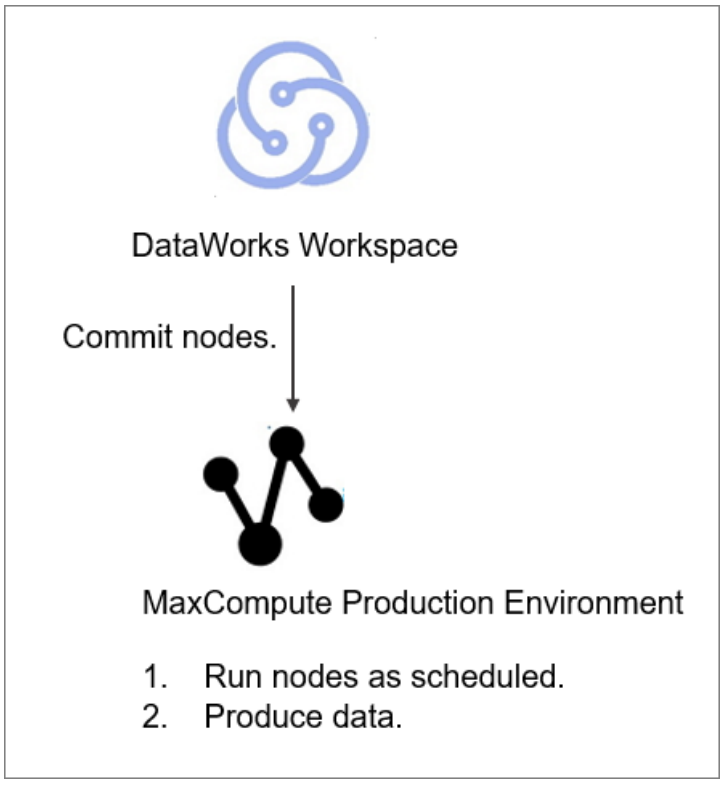
Workspaces in basic mode

A DataWorks workspace in basic mode can be associated with only one project, instance, or database of each compute engine type. A workspace in basic mode does not isolate the development environment from the production environment. In such a workspace, you can perform only basic data development but cannot control the data development process and table permissions.

A workspace in basic mode has the following benefits and risks:

- Benefits: This mode is easy to use. After you commit a node, the scheduling system immediately runs the node on a regular basis to produce data. In this case, you do not need to deploy the node.
- Risks: Developers can modify or commit a node to the scheduling system without the need to obtain approval. This makes the production environment unstable. In addition, if this workspace is associated with a MaxCompute project, developers have the read and write permissions on all the tables of the MaxCompute project by default. Developers can create, delete, or modify tables. This puts data at risk.

The following figure shows the data production process of a DataWorks workspace in basic mode. This workspace is associated with one MaxCompute compute engine instance.



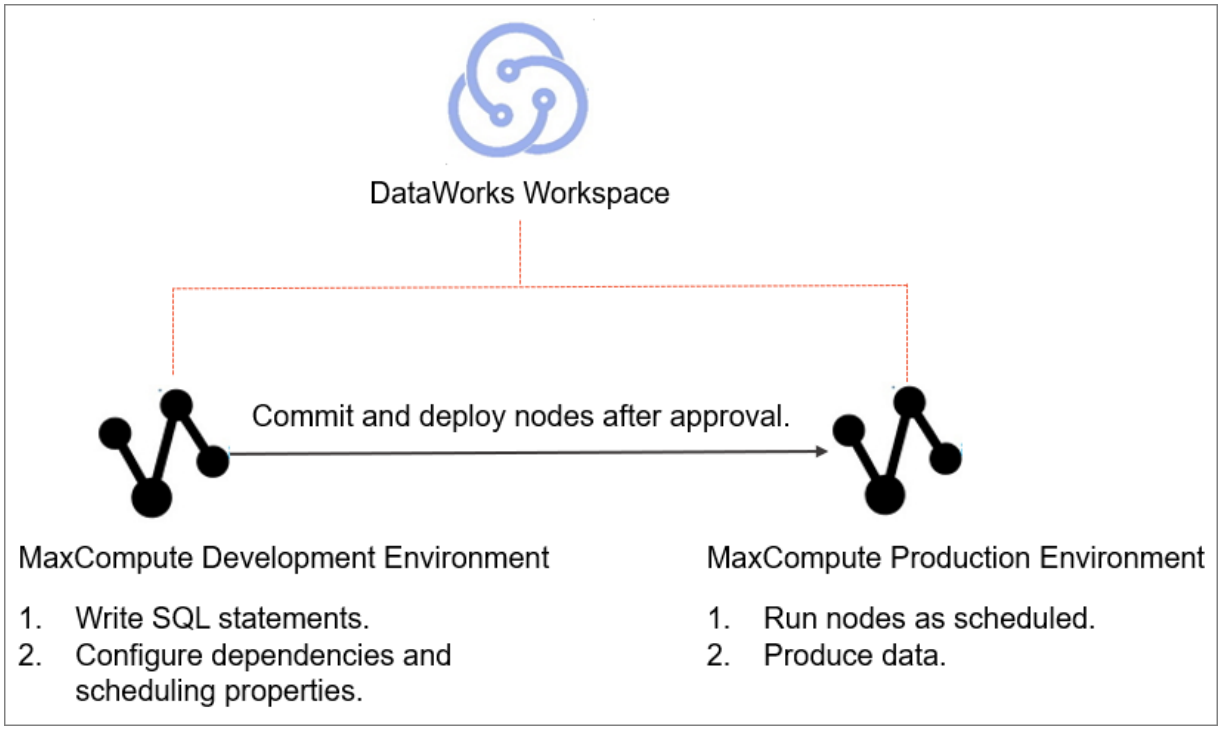
Workspaces in standard mode

A DataWorks workspace in standard mode can be associated with two projects, instances, or databases of each compute engine type. A workspace in standard mode differs from a workspace in basic mode in the following aspects:

- You can modify code only in the development environment.
- After you commit a node, the scheduling system runs the node in the development environment only for smoke testing. The scheduling system does not automatically run this node in the development environment in the future. If you want the scheduling system to automatically run this node in the future, you must deploy it to the production environment.

  You can deploy a node only after you obtain approval from a workspace administrator or O&M expert.

The following figure shows the data production process of a DataWorks workspace in standard mode. This workspace is associated with two MaxCompute compute engine instances.



Types of accounts or roles used to access workspaces in basic mode and standard mode

You can specify the accounts or roles for workspaces in the **Compute Engine Information** section of the **Workspace Management** page.

Workspace mode	Compute engine type	Environment	Access account or role
Standard mode	MaxCompute	Development environment	By default, only the current logon user can perform operations.
		Production environment	The following types of accounts can be specified to perform operations: • Alibaba Cloud account • RAM user
	E-MapReduce (EMR)	Development environment	Only the accounts with the AccessKey IDs and AccessKey secrets specified in the **New EMR cluster** dialog box can be used to perform operations.
		Production environment	
		Development environment	By default, only the current logon user can perform operations.

Workspace mode	Hologres Compute engine type	Environment	Access account or role
		Production environment	The following types of accounts can be specified to perform operations: • Alibaba Cloud account • RAM user
Basic mode	MaxCompute	Development environment, which is also the production environment	By default, only the current logon user can perform operations. The following types of accounts or roles can be specified to perform operations: • Node owner • Alibaba Cloud account
	EMR	Development environment, which is also the production environment	Only the accounts with the AccessKey IDs and AccessKey secrets specified in the **New EMR cluster** dialog box can be used to perform operations.
	Hologres	Development environment, which is also the production environment	By default, only the current logon user can perform operations. The following types of accounts can be specified to perform operations: • Alibaba Cloud account • RAM user

Naming formats of tables in compute engine instances associated with each type of workspace

In a workspace in basic mode, the development environment is not isolated from the production environment. This indicates that the MaxCompute project that is associated with the workspace is used for both the development and production environments. In a workspace in standard mode, the development environment is isolated from the production environment. In this case, the naming formats of tables in the MaxCompute projects that are associated with the workspace differ in the two environments. If you want to access tables for the production environment from the development environment, you must identify tables for the production environment to prevent inappropriate operations. The following table describes the naming formats of tables for the two environments.

Environment	Standard mode	Sample settings
Development environment	Project name_dev.Table name	If you want to create a table named user_info in the projectA project, the table name is shown as projectA_dev.user_info.
Production environment	Project name.Table name	If you want to create a table named user_info in the projectA project, the table name is shown as projectA.user_info.

20.3. Manage members and roles

DataWorks provides roles that have different permissions for you to implement finer-grained permission management. You can add the required users to your workspace and assign the required roles to the users. You can also create custom roles and grant permissions to the roles based on your business requirements.

Background information

Multiple users can be added to the same DataWorks workspace. In this case, if the users have excessive permissions on the workspace, the data security of the workspace may be affected by inappropriate permission use. However, if the users have insufficient permissions on the workspace, they may be unable to use the required features. To resolve this issue, DataWorks provides identities such as members and roles. You can assign different roles to users based on their requirements on the use of workspaces.

If the default roles that are provided by DataWorks cannot meet your requirements, you can create custom roles and grant the required permissions to the roles.

DataWorks provides the following identities:

- **Member**: the Apsara Stack tenant accounts or RAM users that are added to a DataWorks workspace.
- **Cloud account**: Apsara Stack tenant accounts or RAM users.
- **Role**: the carriers that have permissions in a workspace and can be assumed by the members of the workspace. DataWorks provides the following roles:
  - **Project Manager**: the administrators that have all the permissions on the features in a workspace. For example, the workspace administrator role can be used to assign the required role to a RAM user and remove a member that is not the workspace owner from a workspace.
  - **Deploy**: the engineers that have the permissions to deploy nodes.
  - **Development**: the developers that have the permissions to develop and commit nodes.
  - **Model Developer**: the designers that have the permissions to use the data modeling feature.
  - **Visitor**: the visitors that have the read-only permissions on a DataWorks workspace.
  - **Project owner**: the owner that has the highest level of permissions on a workspace.
  - **O&M**: the engineers that have the permissions to allocate resources and deploy nodes.
  - **Security Manager**: the administrators that have the permissions to use Data Security Guard.

  For more information about the permissions of different roles, see Permission list.

Limits

- Only the **Project Manager** and the **Project owner** roles can add users, change the roles of users, and remove **users** and the added **custom roles**.
- You can use only an Apsara Stack tenant account or the RAM user whose role is an administrator or a super administrator of a MaxCompute project to map a custom DataWorks role to a role of the MaxCompute project.

Go to the User Management page

1. Log on to the DataWorks console.
2. On the **DataStudio** page, click the icon in the upper-right corner.

3. In the left-side navigation pane, click **User Management** to go to the **User Management** page.

   You can manage members and roles on the **User Management** page.

Manages members

On the **Manage Members** tab, you can perform the following operations:

- View member information.

  You can view the accounts of members and the roles that are assigned to the members in the current workspace. You can also specify the name of the member, account, or role to search for a specific member. Then, you can view the member information and the number of members to which the role has been assigned. This allows you to realize centralized management of members and roles assigned to the members.

- Add a user.

  i. Click **Add Member** in the upper-right corner of the **Manage Members** tab to add a user to the current workspace.

  ii. In the **Add Member** dialog box, select one or more RAM users from the **Available Accounts** list.

  - **Member**: the Apsara Stack tenant accounts or RAM users that are added to a DataWorks workspace.
  - **Cloud account**: Apsara Stack tenant accounts or RAM users.
  - **Role**: the carriers that have permissions in a workspace and can be assumed by the members of the workspace. DataWorks provides the following roles:
    - **Project Manager**: the administrators that have all the permissions on the features in a workspace. For example, the workspace administrator role can be used to assign the required role to a RAM user and remove a member that is not the workspace owner from a workspace.
    - **Deploy**: the engineers that have the permissions to deploy nodes.
    - **Development**: the developers that have the permissions to develop and commit nodes.
    - **Model Developer**: the designers that have the permissions to use the data modeling feature.
    - **Visitor**: the visitors that have the read-only permissions on a DataWorks workspace.
    - **Project owner**: the owner that has the highest level of permissions on a workspace.
    - **O&M**: the engineers that have the permissions to allocate resources and deploy nodes.
    - **Security Manager**: the administrators that have the permissions to use Data Security Guard.

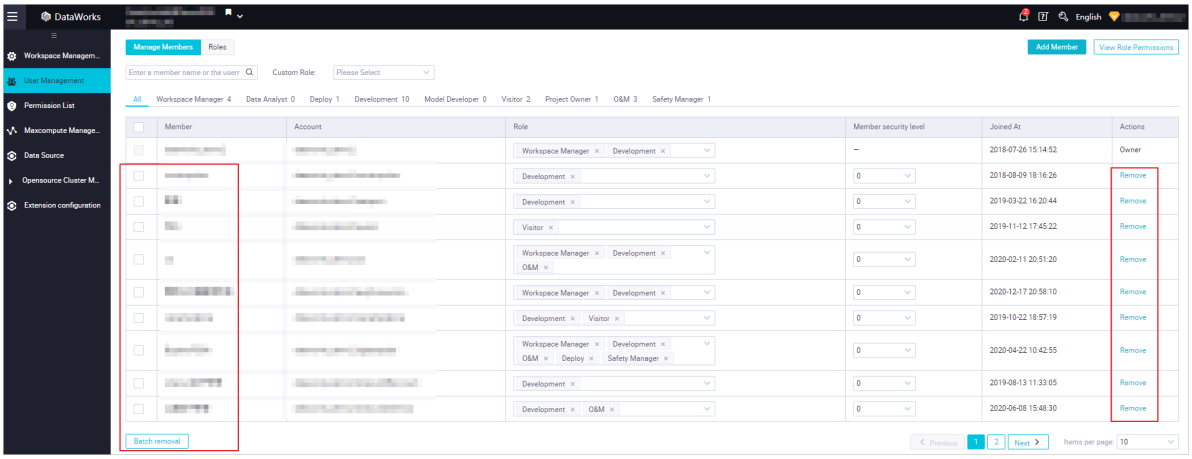
  iii. Click the **>** icon to move the selected RAM users to the **Added Accounts** list.

  iv. Select one or more roles that you want to assign to the selected RAM users.

  v. Click **Confirm**.

- Remove a member.

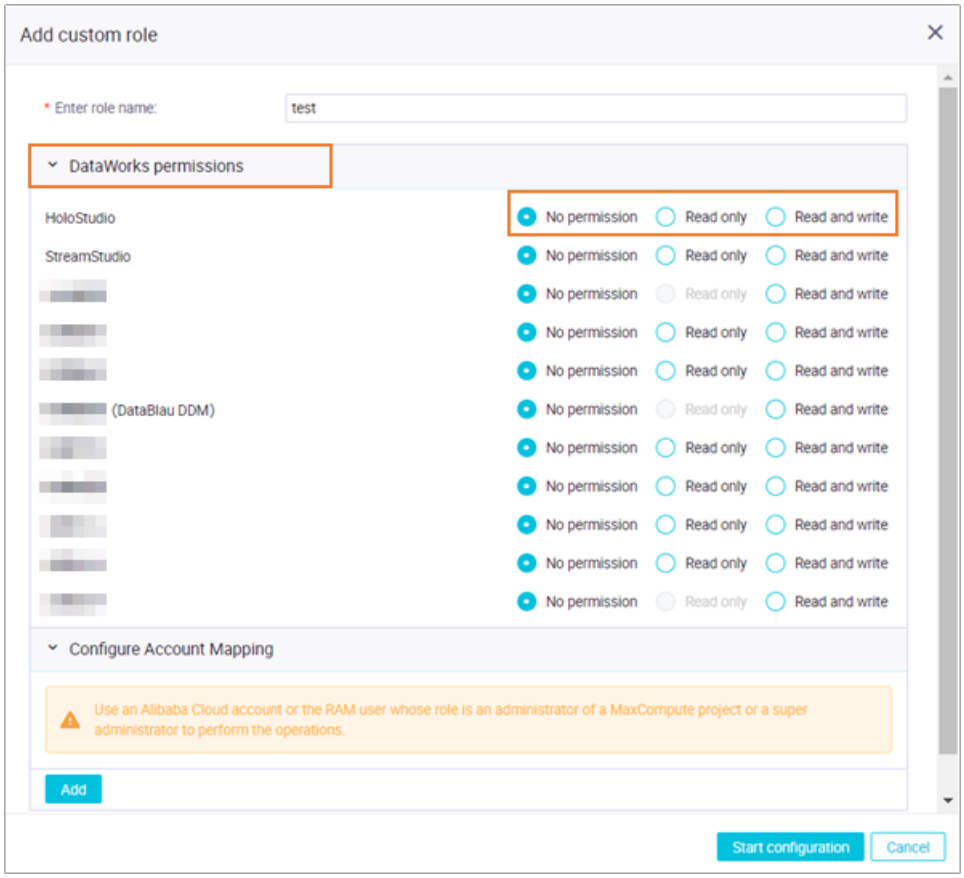
On the **Manage Members** tab, find the member that you want to remove from the workspace and click **Remove** in the **Actions** column to remove the member from the workspace. If you want to remove multiple members from the workspace, you can select them and click Batch removal to remove them at a time.



Manage roles

On the **Roles** tab, you can perform the following operations:

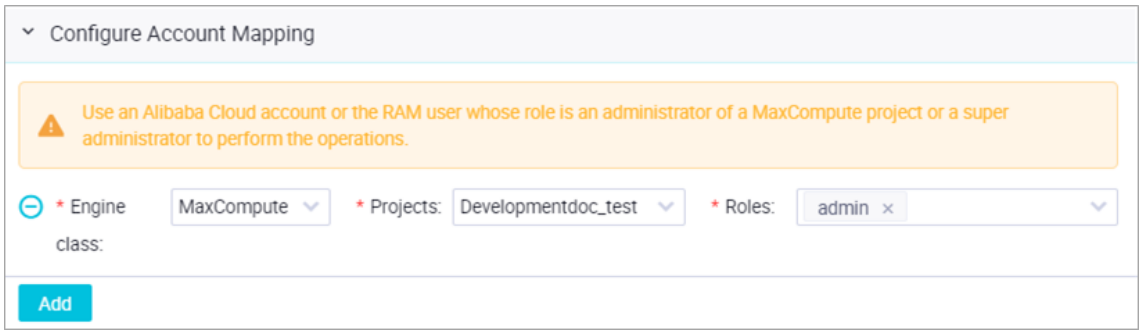
- Create a custom role.
  i. Click **Add Custom Role** in the upper-right corner of the **Roles** tab.
  ii. In the **Add Custom Role** dialog box, enter a name for your custom role, such as test.
  iii. Grant permissions on the required DataWorks modules to the role.
    - **Unauthorized**: indicates that the role has no permissions on the related module.
    - **Read-only**: indicates that the role can only view the data in the related module.
    - **Read and Write**: indicates that the role can modify the data in the related module.



iv. **Map a custom role to a role of a compute engine.**

You can map a custom role to a role of a compute engine. For example, you can map the custom role test to the **Admin** role of a MaxCompute project. In this case, the Admin role is assumed by the custom role when the custom role accesses the MaxCompute project.

> **Note** You can use only an Apsara Stack tenant account or the RAM user whose role is an administrator or a super administrator of a MaxCompute project to map a custom DataWorks role to a role of the MaxCompute project.



v. Click **Configure**.

- View or edit roles.

  You can view the **preset roles** and **custom roles** that have been configured for the workspace on the **Roles** tab. You can also edit or delete **custom roles**. For more information about the permissions of **preset roles**, see Permission list.

20.4. Permission list

DataWorks provides seven roles: workspace owner, workspace administrator, developer, administration expert, deployment expert, visitor, and security expert. You cannot grant the role of the workspace owner to other workspace members. This topic describes the permissions of these roles. In the following tables, Yes indicates that a role has the corresponding permission, and No indicates that a role does not have the corresponding permission.

Data management

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Delete self-created tables	Yes	Yes	Yes	No	No	No	No
Specify categories for self-created tables	Yes	Yes	Yes	No	No	No	No
View favorite tables	Yes	Yes	Yes	No	No	No	No
Create tables	Yes	Yes	Yes	No	No	No	No
Unhide self-created tables	Yes	Yes	Yes	No	No	No	No
Modify the schemas of self-created tables	Yes	Yes	Yes	No	No	No	No
View self-created tables	Yes	Yes	Yes	No	No	No	No
View the content of self-submitted permission requests	Yes	Yes	Yes	No	No	No	No
Hide self-created tables	Yes	Yes	Yes	No	No	No	No
Specify the time-to-live (TTL) for self-created tables	Yes	Yes	Yes	No	No	No	No
Request permissions on tables created by others	Yes	Yes	Yes	No	No	No	No
Delete tables	No	Yes	Yes	No	No	No	No
Update tables	No	Yes	Yes	No	No	No	No
Preview data	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Preview table data of other organizations	Yes	Yes	No	No	No	No	No

Deployment management

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Create deployment tasks	Yes	Yes	Yes	Yes	No	No	No
View the list of deployment tasks	Yes	Yes	Yes	Yes	Yes	Yes	No
Delete deployment tasks	Yes	Yes	Yes	Yes	No	No	No
Run deployment tasks	Yes	Yes	No	Yes	Yes	No	No
View the content of deployment tasks	Yes	Yes	Yes	Yes	Yes	Yes	No

Buttons

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Button: Stop	Yes	Yes	Yes	No	No	No	No
Button: Format	Yes	Yes	Yes	No	No	No	No
Button: Edit	Yes	Yes	Yes	No	No	No	No
Button: Run	Yes	Yes	Yes	No	No	No	No
Button: Zoom In	Yes	Yes	Yes	No	No	No	No
Button: Save	Yes	Yes	Yes	No	No	No	No
Button: Show/Hide	Yes	Yes	Yes	No	No	No	No
Button: Delete	Yes	Yes	Yes	No	No	No	No

Code development

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Save and commit nodes	Yes	Yes	Yes	No	No	No	No

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
View the code of nodes	Yes	Yes	Yes	Yes	Yes	Yes	No
Create nodes	Yes	Yes	Yes	No	No	No	No
Delete nodes	Yes	Yes	Yes	No	No	No	No
View the node list	Yes	Yes	Yes	Yes	Yes	Yes	No
Run nodes	Yes	Yes	Yes	No	No	No	No
Edit the code of nodes	Yes	Yes	Yes	No	No	No	No
Download files	Yes	Yes	Yes	No	No	No	No

Function development

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
View function details	Yes	Yes	Yes	Yes	Yes	Yes	No
Create functions	Yes	Yes	Yes	No	No	No	No
Query functions	Yes	Yes	Yes	Yes	Yes	Yes	No
Delete functions	Yes	Yes	Yes	No	No	No	No

Node types

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Node type: Machine Learning	Yes	Yes	Yes	No	No	No	No
Node type: ODPS MR	Yes	Yes	Yes	Yes	Yes	Yes	No
Node type: Data Sync	Yes	Yes	Yes	Yes	Yes	Yes	No
Node type: ODPS SQL	Yes	Yes	Yes	Yes	Yes	Yes	No

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Node type: XLIB	Yes	Yes	Yes	Yes	Yes	Yes	No
Node type: Shell	Yes	Yes	Yes	Yes	Yes	Yes	No
Node type: Zero-Load Node	Yes	Yes	Yes	Yes	Yes	Yes	No

Resource management

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
View the resource list	Yes	Yes	Yes	Yes	Yes	Yes	No
Delete resources	Yes	Yes	Yes	No	No	No	No
Create resources	Yes	Yes	Yes	No	No	No	No
Upload Python files	Yes	Yes	Yes	No	No	No	No
Upload JAR files	Yes	Yes	Yes	No	No	No	No
Upload TXT files	Yes	Yes	Yes	No	No	No	No
Upload files as Archive resources	Yes	Yes	Yes	No	No	No	No

Workflow development

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Run or stop workflows	Yes	Yes	Yes	No	No	No	No
Save workflows	Yes	Yes	Yes	No	No	No	No
View workflows	Yes	Yes	Yes	Yes	Yes	Yes	No
Commit the code of nodes	Yes	Yes	Yes	No	No	No	No
Modify workflows	Yes	Yes	Yes	No	No	No	No

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
View the workflow list	Yes	Yes	Yes	Yes	Yes	Yes	No
Change the workflow owner	Yes	Yes	No	No	No	No	No
View the code of nodes	Yes	Yes	Yes	No	No	No	No
Delete workflows	Yes	Yes	Yes	No	No	No	No
Create workflows	Yes	Yes	Yes	No	No	No	No
Migrate database tables	Yes	Yes	Yes	No	No	No	No
Create folders	No	Yes	Yes	No	No	No	No
Delete folders	No	Yes	Yes	No	No	No	No
Modify folders	No	Yes	Yes	No	No	No	No
Export workflows	No	Yes	Yes	Yes	No	No	No

Workspace management

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
View the basic information about a workspace	Yes	Yes	Yes	Yes	No	No	No
Create baselines	Yes	Yes	Yes	No	No	No	No
Delete baselines	Yes	Yes	Yes	No	No	No	No
Edit baselines	Yes	Yes	No	No	No	No	No
Search for baselines	Yes	Yes	Yes	No	No	No	No
View baselines	Yes	Yes	No	No	No	No	No
Test connectivity	Yes	Yes	No	No	No	No	No
Create connections	Yes	Yes	No	No	No	No	No
Delete connections	Yes	Yes	No	No	No	No	No

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Edit connections	Yes	Yes	No	No	No	No	No
Search for connections	Yes	Yes	No	No	No	No	No
View the connections configured for a workspace	Yes	Yes	Yes	Yes	Yes	No	No
Enable scheduling	Yes	Yes	No	No	No	No	No
View the settings of scheduling properties of nodes	Yes	Yes	No	No	No	No	No
Add workspace members	Yes	Yes	No	No	No	No	No
Change the roles of workspace members	Yes	Yes	No	No	No	No	No
View the members of a workspace	Yes	Yes	No	No	No	No	No
Remove workspace members	Yes	Yes	No	No	No	No	No
Search for workspace members	Yes	Yes	No	No	No	No	No
Modify the configurations of compute engines	Yes	Yes	No	No	No	No	No
View the configurations of compute engines	Yes	Yes	No	No	No	No	No
Query the members of within the tenant	Yes	Yes	No	No	No	No	No
Modify the basic information about a workspace	Yes	Yes	No	No	No	No	No
View the security policies of compute engines	Yes	Yes	No	No	No	No	No
Modify the security policies of compute engines	Yes	Yes	No	No	No	No	No

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Query the resource groups that are bound to a workspace	Yes	Yes	Yes	No	No	No	No
Query the servers in a resource group	No	Yes	No	No	No	No	No
Delete resource groups	No	Yes	No	No	No	No	No
Remove servers from a resource group	No	Yes	No	No	No	No	No
Configure resource groups for sync nodes	Yes	Yes	Yes	Yes	Yes	Yes	No
Bind multiple resource groups to a workspace	No	Yes	No	No	No	No	No
Add servers to a resource group	No	Yes	No	No	No	No	No
Create resource groups for a workspace	No	Yes	No	No	No	No	No
Query the projects to which a resource group is bound	No	Yes	No	No	No	No	No
Create connections	Yes	Yes	No	No	No	No	No
Edit connections	Yes	Yes	No	No	No	No	No
Share connections	Yes	Yes	No	No	No	No	No
Delete connections	Yes	Yes	No	No	No	No	No
Initialize servers in a resource group	No	Yes	No	No	No	No	No
View the connections configured for a workspace	Yes	Yes	Yes	Yes	Yes	No	No
Test connectivity	Yes	Yes	No	No	No	No	No
Search for connections	Yes	Yes	No	No	No	No	No

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Update the server status and slots of a resource group	No	Yes	No	No	No	No	No
Create real-time sync nodes	Yes	Yes	Yes	No	No	No	No

Workflow O&M

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
View the DAG	Yes	Yes	Yes	Yes	No	No	No
Go to the DataStudio page	Yes	Yes	Yes	No	No	No	No
View the DAG of an instance	Yes	Yes	Yes	Yes	No	No	No
View ancestor and descendant nodes in the DAG	Yes	Yes	Yes	Yes	No	No	No
View the list of workflows	Yes	Yes	Yes	Yes	No	No	No
View the operations logs of workflows	Yes	Yes	Yes	Yes	No	No	No
Perform smoke tests	Yes	Yes	Yes	Yes	No	No	No
Generate retroactive data for nodes	Yes	Yes	Yes	Yes	No	No	No
Change the owner of a node	Yes	Yes	Yes	Yes	No	No	No
Unpublish workflows	Yes	Yes	Yes	Yes	No	No	No
View the details of workflows	Yes	Yes	Yes	Yes	No	No	No
View ancestor and descendant instances of an instance in the DAG	Yes	Yes	Yes	Yes	No	No	No

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Pause instances	Yes	Yes	Yes	Yes	No	No	No
Restore instances	Yes	Yes	Yes	Yes	No	No	No
Terminate an instance	Yes	Yes	Yes	Yes	No	No	No
Terminate multiple instance at a time	Yes	Yes	No	Yes	No	No	No
View the list of instances	Yes	Yes	Yes	Yes	No	No	No
View operational logs	Yes	Yes	Yes	Yes	No	No	No
Rerun an instance	Yes	Yes	Yes	Yes	No	No	No
Rerun multiple instances at a time	Yes	Yes	No	Yes	No	No	No
Rerun multiple instances at a time	Yes	Yes	Yes	Yes	No	No	No
Search for instances	Yes	Yes	Yes	Yes	No	No	No
Set the status of an instance to Successful	Yes	Yes	Yes	Yes	No	No	No
View the lineage of nodes	Yes	Yes	Yes	Yes	Yes	No	No
View node details	Yes	Yes	Yes	Yes	Yes	No	No
View the operations logs of nodes	Yes	Yes	Yes	Yes	Yes	No	No
Freeze and pause nodes	Yes	Yes	Yes	Yes	Yes	No	No
Unfreeze and resume nodes	Yes	Yes	Yes	Yes	Yes	No	No
Change the baseline for nodes	Yes	Yes	Yes	Yes	Yes	No	No
Resume instances	Yes	Yes	Yes	Yes	Yes	No	No
Delete instance dependencies	Yes	Yes	No	Yes	No	No	No
Change the running priority of instances	Yes	Yes	No	Yes	No	No	No

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Forcibly rerun instances	Yes	Yes	No	Yes	No	No	No
View the lineage of instances	Yes	Yes	Yes	Yes	Yes	No	No
View instance details	Yes	Yes	Yes	Yes	Yes	No	No
View runtime logs of instances	Yes	Yes	Yes	Yes	Yes	No	No
View the baselines affected by instances	Yes	Yes	Yes	Yes	Yes	No	No
Unpublish nodes	Yes	Yes	No	Yes	No	No	No

Node maintenance

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Change the baseline for a node	Yes	Yes	Yes	Yes	No	No	No
Change the baseline for multiple nodes at a time	Yes	Yes	No	Yes	No	No	No
View the code of a node	Yes	Yes	Yes	Yes	No	No	No
Change the owner of a node	Yes	Yes	Yes	Yes	No	No	No
Change the owner of multiple nodes at a time	Yes	Yes	No	Yes	No	No	No
Change the resource group for a node	Yes	Yes	Yes	Yes	No	No	No
Change the resource group for multiple nodes at a time	Yes	Yes	Yes	Yes	No	No	No
Perform smoke tests	Yes	Yes	Yes	Yes	No	No	No
Generate retroactive data for nodes	Yes	Yes	Yes	Yes	No	No	No

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Delete instance dependencies	Yes	Yes	No	Yes	No	No	No
Pause instances	Yes	Yes	Yes	Yes	No	No	No
Resume instances	Yes	Yes	Yes	Yes	No	No	No
Refresh the attribute information about instances	Yes	Yes	Yes	Yes	No	No	No
Terminate an instance	Yes	Yes	Yes	Yes	No	No	No
Terminate multiple instance at a time	Yes	Yes	No	Yes	No	No	No
Change the running priority of instances	Yes	Yes	Yes	Yes	No	No	No
Refresh the dependencies of instances	Yes	Yes	Yes	Yes	No	No	No
Rerun an instance	Yes	Yes	Yes	Yes	No	No	No
Rerun multiple instances at a time	Yes	Yes	No	Yes	No	No	No
Set the status of an instance to Successful	Yes	Yes	Yes	Yes	No	No	No
Create data quality rules	Yes	Yes	Yes	Yes	No	No	No
Delete data quality rules	Yes	Yes	Yes	Yes	No	No	No

Dashboard

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
View the number of baselines in the Overtime state	Yes	Yes	Yes	Yes	No	No	No
Remove a record from the dashboard	Yes	Yes	Yes	Yes	No	No	No

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
View the distribution of nodes by status	Yes	Yes	Yes	Yes	No	No	No
View the running information about nodes	Yes	Yes	Yes	Yes	No	No	No
View the distribution of nodes by type	Yes	Yes	Yes	Yes	No	No	No

Baseline checks

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
View the metrics of baselines	Yes	Yes	Yes	Yes	No	No	No
View baselines	Yes	Yes	Yes	Yes	No	No	No

Monitoring and alerts

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
View notification messages	Yes	Yes	Yes	Yes	No	No	No
Disable an alert	Yes	Yes	Yes	Yes	No	No	No
Disable multiple alerts at a time	Yes	Yes	No	No	No	No	No
Enable or disable call notifications	Yes	Yes	Yes	Yes	No	No	No
Create custom notification rules	Yes	Yes	Yes	Yes	No	No	No
Delete custom notification rules	Yes	Yes	Yes	No	No	No	

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Edit custom notification rules	Yes	Yes	Yes	Yes	No	No	No
View custom notification rules	Yes	Yes	Yes	Yes	No	No	No
View all events	Yes	Yes	Yes	Yes	No	No	No
View event details	Yes	Yes	Yes	Yes	No	No	No
View details of personal events	Yes	Yes	Yes	Yes	No	No	No

Data integration

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Resource consumption monitoring menu	Yes	Yes	No	No	No	No	No
View nodes	Yes	Yes	Yes	No	No	No	No
Edit nodes	Yes	Yes	Yes	No	No	No	No
Monitor resource consumption	Yes	Yes	No	No	No	No	No
Delete nodes	Yes	Yes	Yes	No	No	No	No
Migrate database tables	Yes	Yes	No	No	No	No	No

20.5. Manage connections

Connections are used to configure readers and writers during data integration. On the Data Source page of a workspace, you can view and add connections.

Procedure

1. Log on to the DataWorks console.
2. On the **DataStudio** page, click  in the upper-right corner.
3. In the left-side navigation pane, click **Data Source**.

On the **Data Source** page, you can filter connections by conditions such as **Connect To** and **Connection Name**.

Click **Add Connection** in the upper-right corner to add a connection. For more information, see Data sources.