

Alibaba Cloud DataWorks

Data Aggregation

Issue: 20200628









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 due to product version upgrades, adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall 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 the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults" and "as available" basis. 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 bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequential, exemplary, incidental, 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 contact Alibaba Cloud directly if you discover any errors in this document.

Document conventions

Style	Description	Example
	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.
	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.
	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.
	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 <code>cd /d C:/window</code> command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	<code>bae log list --instanceid Instance_ID</code>
[] or [a b]	This format is used for an optional value, where only one item can be selected.	<code>ipconfig [-all -t]</code>

Style	Description	Example
{ } or {a b}	This format is used for a required value, where only one item can be selected.	switch {active stand}

Contents

Legal disclaimer.....	I
Document conventions.....	I
1 Overview.....	1
2 Homepage.....	4
3 Common configurations.....	5
3.1 Test data store connectivity.....	5
3.2 Add a route.....	16
3.3 Configure a whitelist.....	20
3.4 Configure a security group.....	24
4 Batch data synchronization.....	26
4.1 Connections.....	26
4.1.1 Supported data stores and plug-ins.....	26
4.1.2 Connection isolation.....	29
4.2 Node configuration.....	32
4.2.1 Create a sync node by using the codeless UI.....	32
4.2.2 Create a sync node by using the code editor.....	37
4.3 Synchronize incremental data.....	42
4.4 Migrate databases to MaxCompute.....	47
4.4.1 Rules and restrictions.....	47
4.4.2 Migrate a MySQL database.....	50
4.4.3 Migrate an Oracle database.....	52
4.5 Migrate multiple tables to the cloud at a time.....	54
4.5.1 Migrate multiple tables to the cloud at a time.....	54
4.5.2 Add multiple connections at a time.....	57
5 Real-time synchronization.....	59
5.1 Create, commit, and manage real-time sync nodes.....	59
5.2 Supported data stores.....	61
5.3 Fields used for real-time synchronization.....	63
5.4 Reader.....	65
5.4.1 MySQL binlog reader.....	65
5.4.2 Oracle CDC reader.....	66
5.4.3 Datahub reader.....	67
5.4.4 LogHub reader.....	68
5.4.5 Kafka reader.....	69
5.4.6 ApsaraDB for POLARDB reader.....	70
5.5 Writer.....	72
5.5.1 MaxCompute writer.....	72
5.5.2 Hologres writer.....	74
5.5.3 Datahub writer.....	76

5.5.4 Kafka writer.....	77
5.6 Transformation.....	79
5.6.1 Data filtering.....	79
5.6.2 Groovy.....	79
5.6.3 String replacement.....	81
6 Node optimization.....	83
6.1 Optimize synchronization performance.....	83
6.2 Optimize a sync node.....	87
7 Resource groups.....	90
7.1 Overview.....	90
7.2 Use the default resource group.....	93
7.3 Use exclusive resource groups for data integration.....	94
7.4 Add a custom resource group.....	99
8 Appendixes.....	105
8.1 Connection configuration.....	105
8.1.1 Configure an AnalyticDB for MySQL 2.0 connection.....	105
8.1.2 Configure an SQL Server connection.....	107
8.1.3 Configure a MongoDB connection.....	110
8.1.4 Configure a Datahub connection.....	113
8.1.5 Configure a DM connection.....	115
8.1.6 Configure a DRDS connection.....	116
8.1.7 Configure an FTP connection.....	118
8.1.8 Configure an HDFS connection.....	120
8.1.9 Configure a LogHub connection.....	121
8.1.10 Configure a Memcache connection.....	123
8.1.11 Configure a MySQL connection.....	125
8.1.12 Configure an Oracle connection.....	128
8.1.13 Configure an OSS connection.....	129
8.1.14 Configure a Table Store connection.....	132
8.1.15 Configure a PostgreSQL connection.....	133
8.1.16 Configure a Redis connection.....	136
8.1.17 Configure a HybridDB for MySQL connection.....	138
8.1.18 Configure an AnalyticDB for PostgreSQL Connection.....	142
8.1.19 Configure a POLARDB connection.....	145
8.1.20 Configure an AnalyticDB for MySQL 3.0 connection.....	147
8.1.21 Configure a DLA connection.....	149
8.1.22 Configure a MaxCompute connection.....	151
8.1.23 Configure a GDB connection.....	153
8.1.24 Configure an ApsaraDB for OceanBase connection.....	155
8.1.25 Configure a Vertica connection.....	160
8.2 Reader configuration.....	162
8.2.1 Configure DRDS Reader.....	162
8.2.2 Configure HBase Reader.....	169
8.2.3 Configure HDFS Reader.....	177

8.2.4 Configure MongoDB Reader.....	187
8.2.5 Configure DB2 Reader.....	192
8.2.6 Configure MySQL Reader.....	198
8.2.7 Configure Oracle Reader.....	207
8.2.8 Configure OSS Reader.....	216
8.2.9 Configure FTP Reader.....	225
8.2.10 Configure Table Store Reader.....	233
8.2.11 Configure AnalyticDB for MySQL 3.0 Reader.....	239
8.2.12 Configure SQL Server Reader.....	247
8.2.13 Configure LogHub Reader.....	255
8.2.14 Configure Table Store Reader-Internal.....	263
8.2.15 Configure RDBMS Reader.....	270
8.2.16 Stream Reader.....	275
8.2.17 Configure HybridDB for MySQL Reader.....	277
8.2.18 Configure AnalyticDB for PostgreSQL Reader.....	285
8.2.19 Configure POLARDB Reader.....	293
8.2.20 Configure Elasticsearch Reader.....	301
8.2.21 Configure Kafka Reader.....	305
8.2.22 Configure MaxCompute Reader.....	311
8.2.23 Configure InfluxDB Reader.....	319
8.2.24 Configure OpenTSDB Reader.....	322
8.2.25 Configure Prometheus Reader.....	325
8.2.26 Configure PostgreSQL Reader.....	328
8.2.27 Configure OTSStream Reader.....	337
8.2.28 Configure MetaQ Reader.....	347
8.2.29 Configure Hive Reader.....	350
8.2.30 Configure Vertica Reader.....	354
8.2.31 Configure SAP HANA Reader.....	359
8.2.32 Gbase8a Reader.....	361
8.2.33 Configure Datahub Reader.....	367
8.2.34 Configure ApsaraDB for OceanBase Reader.....	370
8.2.35 Hologres Reader.....	376
8.2.36 TSDB Reader.....	385
8.3 Writer configuration.....	398
8.3.1 Configure Datahub Writer.....	398
8.3.2 Configure Db2 Writer.....	401
8.3.3 Configure DRDS Writer.....	404
8.3.4 Configure FTP Writer.....	410
8.3.5 Configure HBase Writer.....	416
8.3.6 Configure HBase11xsql Writer.....	423
8.3.7 Configure HDFS Writer.....	426
8.3.8 Configure Memcache Writer.....	434
8.3.9 Configure MongoDB Writer.....	439
8.3.10 Configure MySQL Writer.....	443
8.3.11 Configure Oracle Writer.....	449

8.3.12 Configure OSS Writer.....	455
8.3.13 Configure PostgreSQL Writer.....	464
8.3.14 Configure Redis Writer.....	471
8.3.15 Configure SQL Server Writer.....	476
8.3.16 Configure Elasticsearch Writer.....	482
8.3.17 Configure LogHub Writer.....	489
8.3.18 Configure Open Search Writer.....	492
8.3.19 Configure Table Store Writer.....	496
8.3.20 Configure RDBMS Writer.....	501
8.3.21 Configure Stream Writer.....	505
8.3.22 Configure HybridDB for MySQL Writer.....	506
8.3.23 Configure AnalyticDB for PostgreSQL Writer.....	512
8.3.24 Configure POLARDB Writer.....	518
8.3.25 Configure TSDB Writer.....	524
8.3.26 Configure AnalyticDB for MySQL 3.0 Writer.....	529
8.3.27 Configure GDB Writer.....	534
8.3.28 Configure Hive Writer.....	542
8.3.29 Configure MaxCompute Writer.....	546
8.3.30 Configure Kafka Writer.....	553
8.3.31 Configure Maxgraph Writer.....	555
8.3.32 Configure Vertica Writer.....	561
8.3.33 Configure Gbase8a Writer.....	564
8.3.34 Configure ClickHouse Writer.....	568
8.3.35 Configure ApsaraDB for OceanBase Writer.....	571
8.3.36 Hologres Writer.....	575

1 Overview

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

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

Data Integration supports a wide range of data stores, including:

- Relational databases: MySQL, SQL Server, PostgreSQL, Oracle, DM, Distributed Relational Database Service (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: Object Storage Service (OSS), Hadoop Distributed File System (HDFS), and File Transfer Protocol (FTP)
- NoSQL: MongoDB, Memcache, Redis, and Table Store
- Message queue: LogHub
- Graph computing engine: GraphCompute
- Real-time data: MySQL binlog and Oracle Change Data Capture (CDC)

For more information, see [Supported data stores and plug-ins](#).

**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 either of the following modes:

- **Codeless UI:** Data Integration provides step-by-step instructions to help you quickly 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 UI can be converted to a script. This conversion is irreversible. After the conversion is completed, you cannot switch back to the codeless UI mode.
- Before writing code, you must configure a connection and create the destination table.

Network types

A data store can reside on a 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. Networks of this type are easy to use.
- **VPC:** a network created on Alibaba Cloud, which is available to only one Alibaba Cloud 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. With 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, ApsaraDB for POLARDB, DRDS, HybridDB for MySQL, AnalyticDB for PostgreSQL, and AnalyticDB for MySQL 3.0. With 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, Table Store, 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.

For more information about classic networks and VPCs, see [FAQ](#).

**Note:**

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

Limits

- Data Integration can synchronize structured, semi-structured, and unstructured data. Structured data stores include RDS and DRDS. Unstructured data, such as OSS objects and text files, must be capable of being converted to structured data. That is, 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 the same region or across regions.

Data can be transmitted between some regions over a classic network, but the network connectivity is not guaranteed. If the transmission fails over a classic network, we recommend that you use a public network connection.

- Data Integration supports only data synchronization but not data consumption.

Reference

- For more information about how to configure a sync node, see [Create a sync node](#).
- For more information about how to process unstructured data, such as objects stored in OSS, see [Access OSS unstructured data](#).
- DataWorks provides the default resource group for you to migrate large amounts of data to the cloud for free. However, the default resource group does not work if a high transmission speed is required or your data stores are deployed in complex environments. You can use exclusive or custom resource groups to run your sync nodes. This guarantees connections to your data stores and enables a higher transmission speed. For more information about exclusive resource groups for data integration, see [Use exclusive resource groups for data integration](#) and [Add a custom resource group](#).

2 Homepage

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

Log on to the [DataWorks console](#), find the target workspace, and click **Data Integration** in the Actions column. The homepage appears by default.

- **New Task:** Click here to go to the **Data Analytics** page, where you can create sync nodes. For more information, see [#unique_11](#).
- **Connection:** Click here to go to the **Data Source** page, where you can 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.
- **Documentation:** You can click **Supported DataSource**, **Job Configuration**, or **Frequently Asked Questions** to view help documents as required.

3 Common configurations

3.1 Test data store connectivity

This topic describes the data stores that support connectivity testing and how to troubleshoot common connectivity testing failures.

Support for connectivity testing

The following table describes the support for connectivity testing by various data stores. If the connectivity test fails for a data store connection added in **connection string mode**, the possible causes are as follows:

- The data store is not started. Check whether the data store is started.
- DataWorks cannot access the network where the data store resides. Make sure that the network where the data store resides is connected to Alibaba Cloud.
- DataWorks is prohibited to access the network where the data store resides by a network firewall. Add the IP addresses or Classless Inter-Domain Routing (CIDR) blocks used by DataWorks to a whitelist. For more information, see [Configure a whitelist](#).
- The domain name of the data store cannot be resolved. Make sure that the domain name of the data store can be resolved properly.
- The default resource group is used but the data store is deployed in a Virtual Private Cloud (VPC) or an Internet data center (IDC). Use a [custom resource group](#) or an [exclusive resource group](#) to guarantee network connectivity. In this case, connectivity testing is not supported. Whether a sync node can be run depends on the selected resource group.

Data store	Data store type	Network type	Connectivity testing
MySQL	ApsaraDB	Classic network	Supported
		VPC	Supported
	Connection string mode		Not supported if an internal endpoint is used
	User-created data store hosted on Elastic Compute Service (ECS)	Classic network	Supported
		VPC	Not supported

Data store	Data store type	Network type	Connectivity testing
SQL Server	ApsaraDB	Classic network	Supported
		VPC	Supported
	Connection string mode		Not supported if an internal endpoint is used
	User-created data store hosted on ECS	Classic network	Supported
		VPC	Not supported
PostgreSQL	ApsaraDB	Classic network	Supported
		VPC	Supported
	Connection string mode		Not supported if an internal endpoint is used
	User-created data store hosted on ECS	Classic network	Supported
		VPC	Not supported
Oracle	Connection string mode		Not supported if an internal endpoint is used
	User-created data store hosted on ECS	Classic network	Supported
		VPC	Not supported
DRDS	ApsaraDB	Classic network	Supported
		VPC	Coming soon
HybridDB for MySQL	ApsaraDB	Classic network	Supported
		VPC	Supported
AnalyticDB for PostgreSQL	ApsaraDB	Classic network	Supported
		VPC	Supported
MaxCompute	ApsaraDB	Classic network	Supported
AnalyticDB for MySQL 2.0	ApsaraDB	Classic network	Supported
		VPC	Not supported
OSS	ApsaraDB	Classic network	Supported
		VPC	Supported

Data store	Data store type	Network type	Connectivity testing
Hadoop Distributed File System (HDFS)	Connection string mode		Not supported if an internal endpoint is used
	User-created data store hosted on ECS	Classic network	Supported
		VPC	Not supported
FTP	Connection string mode		Not supported if an internal endpoint is used
	User-created data store hosted on ECS	Classic network	Supported
		VPC	Not supported
MongoDB	ApsaraDB	Classic network	Supported
		VPC	Not supported
	Connection string mode		Not supported if an internal endpoint is used
	User-created data store hosted on ECS	Classic network	Supported
		VPC	Not supported
Memcache	ApsaraDB	Classic network	Supported
		VPC	Not supported
Redis	ApsaraDB	Classic network	Supported
		VPC	Not supported
	Connection string mode		Not supported if an internal endpoint is used
	User-created data store hosted on ECS	Classic network	Supported
		VPC	Not supported
Table Store	ApsaraDB	Classic network	Supported
		VPC	Not supported
Datahub	ApsaraDB	Classic network	Supported
		VPC	Not supported

If you specify an internal endpoint in a VPC when adding a connection for a data store, connectivity testing is not supported. You can save the settings without clicking

Test Connection. You must select a custom resource group or an exclusive resource group when creating a sync node that uses the connection. In addition, you can create the sync node only in the code editor. For more information, see [#unique_15/unique_15_Connect_42_section_hcx_l05_vcg](#), [Add a custom resource group](#).

Connectivity testing scenarios

This section describes the scenarios of connectivity testing by using relational databases as an example.

- Data store deployed in a local IDC
 - Public network access available: Connectivity testing is supported for a connection to such a data store. You must add the connection based on the JDBC URL. When you add the connection based on the JDBC URL, check the network reachability and whitelist settings to make sure that the resource group for running sync nodes can access the data store over the public network. If you use a public endpoint, pay attention to the data transfer cost for the public network. For more information, see [#unique_16/unique_16_Connect_42_section_qvr_1d5_hpo](#).
 - Public network access unavailable: Connectivity testing is not supported for a connection to such a data store. You can create sync nodes that use the connection only in the code editor. You can add the connection based on the JDBC URL. If you have connected the local IDC to a VPC, purchase an exclusive resource group for data integration and submit a ticket. For more information, see [#unique_15/unique_15_Connect_42_section_hcx_l05_vcg](#). You can also upgrade your DataWorks to Professional Edition and run sync nodes on a custom resource group. For more information, see [Add a custom resource group](#).
- User-created data store hosted on ECS
 - Public network access available: Connectivity testing is supported for a connection to such a data store. You must add the connection based on the JDBC URL. When you add the connection based on the JDBC URL, check the network reachability and whitelist settings to make sure that the resource group for running sync nodes can access the data store over the public network. If you use a public endpoint, pay

attention to the data transfer cost for the public network. For more information, see [#unique_16/unique_16_Connect_42_section_qvr_1d5_hpo](#).

- Data store that resides on a classic network:
 - If the DataWorks workspace and the data store are in the same region, connectivity testing is supported, and you can add the connection based on the JDBC URL. You can run sync nodes on the default resource group, which is not recommended.
 - If the DataWorks workspace and the data store are in different regions, connectivity testing is not supported. You can add the connection based on the JDBC URL. In this case, you must run the sync nodes that uses the connection on a custom resource group and can create the sync nodes only in the code editor.
 - If the user-created data store is hosted on ECS instances that reside on a classic network, network connectivity is not guaranteed when sync nodes are run on the default resource group. We recommend that you run the sync nodes on a custom resource group. If you use a custom resource group or the connectivity test fails, you must create the sync nodes in the code editor.
 - We recommend that you migrate the data store to a VPC.
- Data store that resides in a VPC and uses an internal endpoint: Connectivity testing is not supported. You can add the connection based on the JDBC URL. In this case, you must run the sync nodes that use the connection on a custom resource group or an exclusive resource group for data integration and can create the sync nodes only in the code editor.
- Alibaba Cloud services
 - Connection added in instance mode:
 - DataWorks automatically delivers the endpoints for connections added in instance mode for Apsara DB for POLARDB, Distributed Relational Database Service (DRDS), HybridDB for MySQL, AnalyticDB for PostgreSQL, AnalyticDB for MySQL 3.0, ApsaraDB RDS for MySQL, ApsaraDB RDS for PostgreSQL, and ApsaraDB RDS for SQL Server, according to the running status and environment of sync nodes. Connectivity testing is supported for such connections, and you can run the sync nodes on the default resource group.
 - You can also add connections in instance mode for ApsaraDB for Redis, ApsaraDB for MongoDB, and AnalyticDB for MySQL 2.0. However, such connections do not support reverse VPC access or connectivity testing. You must run the sync nodes

that use such connections on a custom resource group and can create the sync nodes only in the code editor.

- Public network access available: Connectivity testing is supported for a connection to such a data store. You must add the connection based on the JDBC URL. We recommend that you add a connection in instance mode preferentially. When you add the connection based on the JDBC URL, check the network reachability and whitelist settings to make sure that the resource group for running sync nodes can access the data store over the public network. If you use a public endpoint, pay attention to the data transfer cost for the public network.
- Data store that resides on a classic network:
 - If the DataWorks workspace and the data store are in the same region, connectivity testing is supported. You must add the connection based on the JDBC URL.
 - If the DataWorks workspace and the data store are in different regions, connectivity testing is not supported. You can add the connection based on the JDBC URL. In this case, you must run the sync nodes that uses the connection on a custom resource group and can create the sync nodes only in the code editor.
 - We recommend that you add a connection in instance mode preferentially.
- Data store that resides in a VPC and uses an internal endpoint: Connectivity testing is not supported. You can add the connection based on the JDBC URL. In this case, you must run the sync nodes that use the connection on a custom resource group or an exclusive resource group for data integration and can create the sync nodes only in the code editor. We recommend that you add a connection in instance mode preferentially.

Three types of endpoints are available for centralized services, such as MaxCompute, Object Storage Service (OSS), and LogHub. You can select one according to your needs.

**Note:**

- The constraints on endpoints for other data stores such as HDFS, Redis, and MongoDB are the same as those for relational databases.
- When you select an endpoint for a connection, you must check the node configuration mode (codeless UI or code editor) and selected resource group (default, custom, or exclusive) to make sure that the resource group can access the data store.

- Considering the characteristics of HBase and HDFS, we recommend that you use a custom resource group or an exclusive resource group for data integration to run sync nodes for these data stores.
- Connectivity testing is supported for connections to data stores in Finance Cloud, and you can add such connections in instance mode. If the connectivity test fails, run sync nodes on a custom resource group.

Application scenarios of exclusive resources

- Scenario 1: The data store in a VPC and the DataWorks workspace are in different regions.

An exclusive resource group for data integration cannot access data stores across VPCs. If your data store and the DataWorks workspace are in different regions, follow these steps:

1. Create a VPC in the region where the DataWorks workspace resides.
 2. Connect the VPC created in the previous step to the VPC where the data store resides through Cloud Enterprise Network.
 3. Purchase an exclusive resource group for data integration in the same zone as that of the data store and bind the resource group to the created VPC.
 4. Submit a ticket to enable network access.
- Scenario 2: The data store in a VPC and the DataWorks workspace are in the same region.

To synchronize data from or to data stores in a VPC, you must purchase an exclusive resource group for data integration in the same zone as that of the data stores and bind the resource group to the VPC where the data stores reside. If the synchronization fails after binding, add the CIDR block of the VPC to the whitelist or security group of the data stores.

Services for enabling network access

- For more information about how to enable network access through Enterprise Cloud Network, see [Enterprise Cloud Network](#).
- For more information about how to enable network access through Express Connect, see [Express Connect](#).
- For more information about how to enable network access through VPN Gateway, see [VPN Gateway](#).

Note on the scheduling cluster

- Currently, Alibaba Cloud has deployed scheduling clusters in the China (Hangzhou), China (Shenzhen), China (Hong Kong), and Singapore regions. DataWorks assumes that the scheduling cluster deployed in the China (Hangzhou) region is used when checking the network connectivity to your data store. For example, if your MongoDB data store is deployed on a classic network in the China (Beijing) region, DataWorks determines that the scheduling cluster cannot access the data store due to the region difference.
- The OXS cluster and the ECS cluster cannot communicate with each other through the internal network.

The scheduling cluster for RDS databases is an OXS cluster. The OXS cluster can communicate with RDS databases in all regions in mainland China through the internal network. An ECS cluster on a classic network serves as the scheduling cluster for other data stores.

For example, when you synchronize data from an RDS database to a user-created database, the connectivity test can be passed for the connections to both databases. However, during node scheduling, the RDS database uses the OXS cluster to schedule the sync node, whereas the user-created database uses the ECS cluster to schedule the sync node. The ECS cluster cannot access the RDS database, and the synchronization fails. We recommend that you add the connection for the RDS database as a MySQL connection in JDBC URL mode. This guarantees that both databases can be accessed by the ECS cluster, and the synchronization is successful.

View the resource group on which a sync node is run

- A sync node for an RDS database is scheduled in the OXS cluster.

```
Copyright 2014 Alibaba Group, All rights reserved.
2017-08-28 16:04:39 : Start Job[47529327], traceId
[14849#30623#32374#373#2311230979#23149290, 561660734#1354746#group_14849_cdp_ecs], running in
Pipeline[basecommon_group_14849_cdp_oxs]
2017-08-28 16:04:39 : ---
Reader: sqlserver
    shared=false
    instanceName=[rm-bp19d6og9h641w21a]
    password=[xxxxxxx]
column=[["id","investid","memberid","recommended","level","type","createtime","issue","sharehomelevel","percent"]
    description=[xtrds
    gmtCreate=[2017-08-25 14:26:18
    type=[rds]
    datasourceType=[rds
    database=[xintou365db
    datasourceBackup=[xtrds
    name=[xtrds
    tenantId=[14849
    subType=[sqlserver
    where=[
    id=[69444
    splitPk=[
    rdsOwnerId=[1208951357358306
    projectId=[56023]
```

To determine the resource group running the sync node, check the log details.

- If the logs contain information similar to the following, the sync node is run on the default resource group:

```
running in Pipeline[basecommon_group_xxxxxxxxx]
```

- If the logs contain information similar to the following, the sync node is run on a custom resource group:

```
running in Pipeline[basecommon_xxxxxxxxx]
```

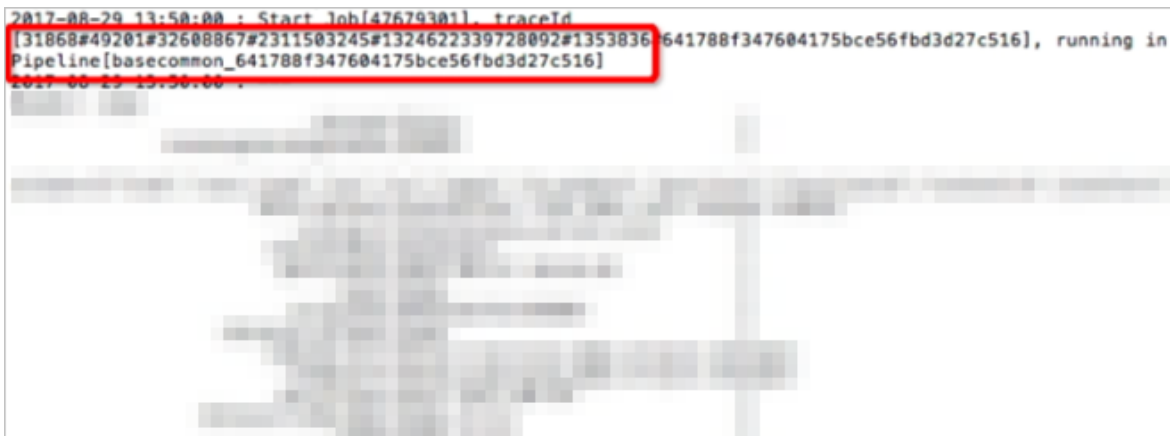
- If the logs contain information similar to the following, the sync node is run on an exclusive resource group for data integration:

```
running in Pipeline[basecommon_S_res_group_xxx]
```

- When a sync node for other types of data stores is scheduled in the ECS cluster, the log information is as shown in the following figure.

```
Copyright 2014 Alibaba Group, All rights reserved.
2017-08-28 17:17:21 : Start Job[120407321], traceId [142#20013#100005750464#100035526779#1860140385521331#581760#group_142_cdp_ecs],
running in Pipeline[basecommon_group_142_cdp_ecs]
2017-08-28 17:17:21 : ---
Reader: rds
    shared=false
    1
```

- When a custom scheduling resource is used as the scheduling cluster, the log information is as shown in the following figure. You can determine whether a custom resource group is used based on the following log information.



- You can go to the testing page of Data Integration and click **Run** for a sync node to schedule it in the ECS cluster. Sync nodes for an RDS database must be scheduled in the OXS cluster. Therefore, an RDS-related sync node may be run manually, but fails to be run as scheduled. In this case, you must click **Test Run** on the **Scheduling Maintenance** page.

Common connectivity test failures

When a connectivity test fails, verify that the region, network type, RDS whitelist, database name, and username are properly configured for the connection. If your connectivity test fails, you can first troubleshoot the failure based on common Data Integration failures. The common connectivity test failures are as follows:

- The database password is incorrect.
- The network connection fails, as shown in the following figure.

`"com.mysql.jdbc.exceptions.jdbc4.CommunicationsException:
Communications link failure`

- A network error occurs during synchronization.

Check the logs and determine which resource group incurs the issue. Check whether the problematic resource group is a custom one.

If so, check whether the CIDR block of the custom resource group is added to the whitelist of the corresponding data store, such as the ApsaraDB for RDS instance.



Note:

The CIDR block of the custom resource group must be added to the whitelist of the MongoDB data store.

Check whether the connectivity test is passed for both connections and whether whitelists of RDS and MongoDB are complete.

**Note:**

If required information is unavailable in the whitelists, sync nodes may fail to be run. If a sync node is delivered to a scheduling server whose IP address has been added to the whitelists, the sync node can be run. Otherwise, the sync node fails to be run.

- The result shows that a sync node is run but an error is reported, indicating that port 8000 is disconnected.

This issue occurs because a custom resource group is used and no inbound rule is configured for the IP address 10.116.134.123 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 sync node again.

Examples of connectivity test failures

Example 1

- Symptom

A data store failed the connectivity test. Database URL: `jdbc:mysql://xx.xx.xx.x:xxxx/t_uoer_brade`. Username: `xxxx_test`. Error message: Access denied for user '`xxxx_test`' '@'%' to database '`yyyy_demo`'.

- Troubleshooting method

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

Example 2

- Symptom

A data store failed the connectivity test.

error message: Timed out after 5000 ms while waiting for a server that matches ReadPreferenceServerSelector{readPreference=primary}. Client view of cluster state is {type=UNKNOWN, servers=[(xxxxxxxxxx), type=UNKNOWN, state=CONNECTING,

```
exception={com.mongodb.MongoSocketReadException: Prematurely reached end of stream}}]
```

- Troubleshooting method

Before testing the connectivity to a MongoDB data store that is not deployed in a VPC, you must add related CIDR block to the whitelist of the data store. For more information, see [Configure a whitelist](#).

3.2 Add a route

This topic describes how to add a route for your exclusive resource group to access a data store in a Virtual Private Cloud (VPC) or an on-premises data center.

Prerequisites

An exclusive resource group is bound to your VPC. For more information, see [#unique_21](#).

Context

You can add a route to connect your exclusive resource group to a specific data store. If the exclusive resource group still fails to connect to the data store after you add the route, fix the issue as follows:

- Check whether the data store resides in an on-premises data center or a different VPC from the exclusive resource group. If so, connect the VPC to which the exclusive resource group is bound to the network where the data store resides through Express Connect or Cloud Enterprise Network (CEN). For more information, see [Network connection overview](#).
- Check whether the data store has a whitelist or access to the data store is controlled by security group. If the data store has a whitelist, add the information, such as the Elastic Network Interface (ENI) IP address, about the exclusive resource group to the whitelist. If access to the data store is controlled by security group, configure a security group rule to allow the exclusive resource group to access the data store.
- Make sure that Object Storage Service (OSS) and the corresponding service port are normal.

Add a route to a data store in a VPC

1. Log on to the [DataWorks console](#).
2. In the left-side navigation pane, click **Resource Groups**.
3. On the Exclusive Resource Groups tab that appears, find the target exclusive resource group and click **Add VPC Binding** in the Actions column.

4. On the page that appears, click **Custom routing** in the Actions column of the exclusive resource group.
5. In the **Custom routing** pane that appears, click **New route**. The New route dialog box appears.



If the exclusive resource group is bound to a VSwitch, the system adds a route for the exclusive resource group by default. The destination of this route is the Classless Inter-Domain Routing (CIDR) block of the VPC where the VSwitch resides. This ensures that the exclusive resource group can access all data stores in this VPC. You can delete this route in the **Custom routing** pane.

6. In the **New route** dialog box, set the parameters as required.

The screenshot shows the 'New route' dialog box with the following configuration:

- Purpose type:** VPC (selected, highlighted with a red box)
- Destination:** China (Shan...)
- Connection mode:** Switch (selected)
- Destination:** (empty dropdown)
- Switch instance:** New Switch
- Routing Diagram:** (visual representation of the network topology)
- Buttons:** Generate route, Close

Parameter	Description
Purpose type	The type of the route destination. If the data store that the exclusive resource group needs to access resides in a VPC, select VPC .

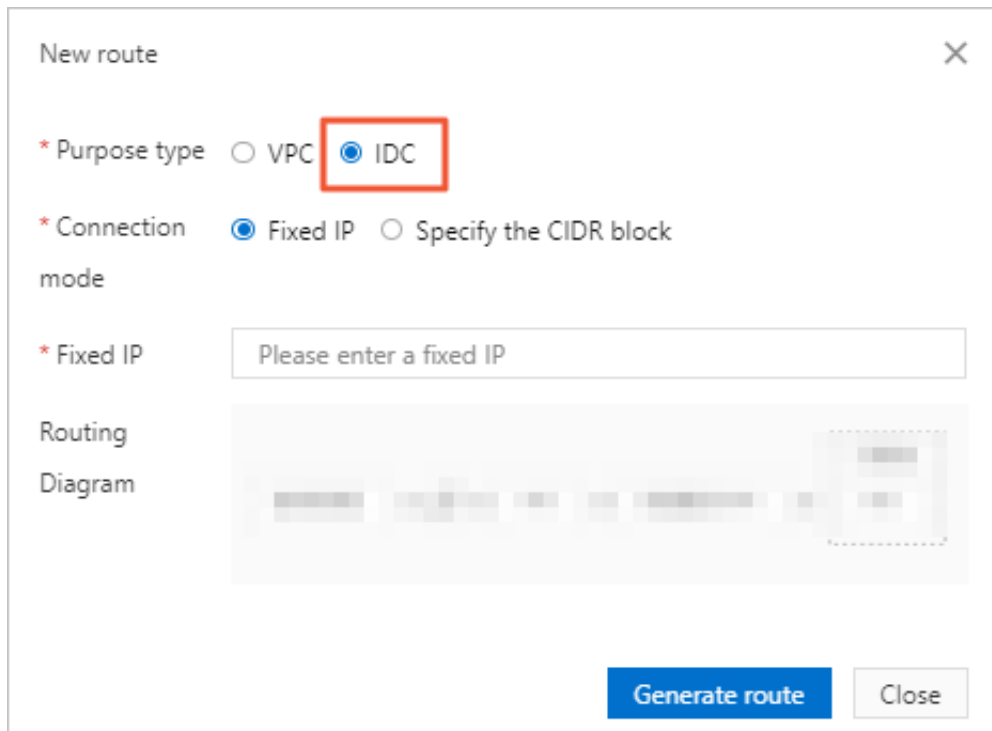
Parameter	Description
Destination VPC	<p>The region and name of the destination VPC where the data store resides.</p> <div>  Note: This parameter appears only when Purpose type is set to VPC. </div>
Connection mode	<p>The connection mode between the exclusive resource group and the destination VPC. Valid values:</p> <ul style="list-style-type: none"> Switch: If you select this option, you must select a VSwitch from the Destination Switch instance drop-down list. <div>  Note: You need to set Destination Switch instance only when you set Connection mode to Switch. </div> <p>If you set Connection mode to Switch and specify a VSwitch, make sure that the target data store is bound to the specified VSwitch. If the target data store resides on multiple ECS instances that are bound to different VSwitches, add a route to each VSwitch.</p> Fixed IP: If you select this option, you must enter an IP address in the Fixed IP field. <p>You must enter the IP address of the ECS instance where the target data store resides in the destination VPC.</p> Specify the CIDR block: If you select this option, you must enter a CIDR block in the Destination CIDR block field. <p>You must enter a subset of the CIDR block of the destination VPC, which contains the IP address of the ECS instance where the target data store resides.</p>
Routing Diagram	The routing diagram, which cannot be changed.

7. Click **Generate route**.

Add a route to a data store in an on-premises data center

- Log on to the DataWorks console. In the left-side navigation pane, click **Resource Groups**.
- On the Exclusive Resource Groups tab that appears, find the target exclusive resource group and click **Add VPC Binding** in the Actions column.

3. On the page that appears, click **Custom routing** in the Actions column of the exclusive resource group.
4. In the **Custom routing** pane that appears, click **New route**.
5. In the **New route** dialog box that appears, set the parameters as required.



New route

* Purpose type ☐ VPC ☒ IDC

* Connection mode ☒ Fixed IP ☐ Specify the CIDR block

* Fixed IP

Routing Diagram

Parameter	Description
Purpose type	The type of the route destination. If the data store that the exclusive resource group needs to access resides in an on-premises data center, select IDC .
Connection mode	<p>The connection mode between the exclusive resource group and the destination on-premises data center. Valid values:</p> <ul style="list-style-type: none">• Fixed IP: If you select this option, you must enter an IP address in the Fixed IP field. <p>You must enter the IP address of the ECS instance where the target data store resides in the destination on-premises data center.</p> <ul style="list-style-type: none">• Specify the CIDR block: If you select this option, you must enter a CIDR block in the Destination CIDR block field. <p>You must enter a subset of the CIDR block of the destination on-premises data center, which contains the IP address of the ECS instance where the target data store resides.</p>

Parameter	Description
Routing Diagram	The routing diagram, which cannot be changed.

6. Click **Generate route**.

3.3 Configure a whitelist

This topic describes how to configure whitelists for DataWorks workspaces in different regions.

If you use ApsaraDB for RDS as a data store, you must configure a whitelist for the ApsaraDB for RDS instance to ensure that it is accessible.

Before using a data store, you must add the IP addresses or Classless Inter-Domain Routing (CIDR) blocks that you use to access the data store to a whitelist of the instance where the data store resides. This improves security and stability of the database.



Note:

The configured whitelist is valid for data integration nodes only.

Determine the IP addresses or CIDR blocks to be added to a whitelist

1. Log on to the [DataWorks console](#). In the left-side navigation pane, click **Workspaces**.
2. Move the pointer over the region in the top navigation bar and click the target region.

Currently, DataWorks supports multiple regions. Select the region of the MaxCompute project that you have purchased.

3. Determine the IP addresses or CIDR blocks to be added to a whitelist based on the region of the workspace.

Currently, access to some data stores is restricted by whitelists. You must add the IP addresses or CIDR blocks used by Data Integration to these whitelists. Otherwise, Data Integration cannot access these data stores. For example, you must add IP addresses or CIDR blocks to a whitelist of an ApsaraDB for RDS, ApsaraDB for MongoDB, or ApsaraDB

for Redis instance that serves as a data store. Add IP addresses or CIDR blocks to a whitelist based on the resource group type as follows:

- If sync nodes run on a custom resource group, add internal and public IP addresses of Elastic Compute Service (ECS) instances on the custom resource group to a whitelist of the data store.
- If sync nodes run on the default resource group, add the IP addresses or CIDR blocks of the region where the workspace resides to a whitelist of the data store. The following table lists the IP addresses or CIDR blocks used by each region.

Region	Whitelist
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
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
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)	10.152.162.0/24,11.192.196.0/24,11.193.11.0/24,100.64.0.0/10,11.192.196.0/24,47.89.61.0/24,47.91.171.0/24,11.193.118.0/24,47.75.228.0/24

Region	Whitelist
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
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)	11.193.203.0/24,11.194.68.0/24,11.194.69.0/24,100.64.0.0/10,47.252.55.0/24,47.252.88.0/24
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
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
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
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

Region	Whitelist
China (Beijing) Alibaba GovCloud	<p>11.194.116.0/24,100.64.0.0/10</p> <p>If you fail to add the preceding CIDR blocks, add the following IP addresses:</p> <p>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</p>

Configure a whitelist for an ApsaraDB for RDS instance

An RDS connection can be added in either of the following modes:

- ApsaraDB for RDS instance mode

You can add an RDS connection by specifying the corresponding ApsaraDB for RDS instance. Currently, connectivity testing is supported for RDS connections added in this mode, including RDS connections for ApsaraDB for RDS instances deployed in a VPC. If an RDS connection added in this mode fails the connectivity test, add the RDS connection in JDBC URL mode.

- JDBC URL mode

When adding an RDS connection in JDBC URL mode, specify the internal endpoint of the corresponding ApsaraDB for RDS instance as the JDBC URL. If no internal endpoint is available, enter the public endpoint as the JDBC URL. If an internal endpoint is used, data is transferred inside an IDC of Alibaba Cloud. In this case, the data synchronization is fast. If a public endpoint is used, the data synchronization speed depends on your public network bandwidth.

Configure a whitelist for an ApsaraDB for RDS instance

To allow Data Integration to synchronize data from or to an ApsaraDB for RDS instance, you must connect Data Integration to the ApsaraDB for RDS instance through a standard database protocol. An ApsaraDB for RDS instance allows connections from all IP addresses by default. However, if you have specified a whitelist for the ApsaraDB for RDS instance, you must add the IP addresses or CIDR blocks used by Data Integration to the whitelist. This operation is unnecessary if you have not specified a whitelist for the ApsaraDB for RDS instance.

If you have specified an endpoint whitelist for the ApsaraDB for RDS instance, go to the **Data Security** page for the instance in the ApsaraDB for RDS console, and modify the whitelist to add the corresponding IP addresses or CIDR blocks.

**Note:**

If you use a custom resource group to run sync nodes that synchronize data from or to the ApsaraDB for RDS instance, you must add the IP addresses of ECS instances on the custom resource group to the whitelist.

3.4 Configure a security group

This topic describes how to configure security groups for DataWorks workspaces in different regions.

If you use a user-created data store deployed on Elastic Compute Service (ECS) instances, you must configure a security group to guarantee successful connection to the data store.

Before using a data store, you must add the IP addresses or Classless Inter-Domain Routing (CIDR) blocks that you use to access the data store to a whitelist of the instance where the data store resides. This improves security and stability of the database. For more information, see [Configure a whitelist](#).

Determine the security group rule to be configured

- If sync nodes for a user-created data store deployed on your ECS instances run on a custom resource group, add internal or public IP addresses and ports of ECS instances on the custom resource group to your security group.
- If sync nodes for a user-created data store deployed on your ECS instances run on the default resource group, you must authorize the default resource group to access your ECS instances. For example, your ECS instances reside in the China (Beijing) region. You must add the authorization object sg-2ze3236e8pcbxw61o9y0 and account ID 1156529087455811 to your security group, as described in the following table. In addition, you can add connections for the data store only in the China (Beijing) region.

Region	Security group	Account ID
China (Hangzhou)	sg-bp13y8iuj33uqpqvqgw2	1156529087455811
China (Shanghai)	sg-uf6ir5g3rlu7thymywza	1156529087455811
China (Shenzhen)	sg-wz9ar9o9jgok5tajj7ll	1156529087455811

Region	Security group	Account ID
Singapore	sg-t4n222njci99ik5y6dag	1156529087455811
China (Hong Kong)	sg-j6c28uqpqb27yc3tjmb6	1156529087455811
US (Silicon Valley)	sg-rj9bowpmdvhy153lza2j	1156529087455811
US (Virginia)	sg-0xienf2ak8gs0puz68i9	1156529087455811
China (Beijing)	sg-2ze3236e8pcbxw61o9y0	1156529087455811

**Note:**

The default resource group uses IP addresses on classic networks. If your ECS instances reside in a Virtual Private Cloud (VPC), you cannot add the preceding information to your security group due to the network type difference.

Configure a security group for ECS instances

1. Log on to the ECS console.
2. In the left-side navigation pane, choose **Network & Security > Security Groups**. Select the target region.
3. Find the security group for which you want to add an authorization rule, and click **Add Rules** in the Actions column.
4. On the **Security Group Rules** page, click the **Inbound** tab and then click **Add Security Group Rule** in the upper-right corner.
5. In the **Add Security Group Rule** dialog box that appears, set the parameters.
6. Click **OK**.

4 Batch data synchronization


4.1 Connections




4.1.1 Supported data stores and plug-ins

Data integration is a stable, efficient, and scalable data synchronization platform. It provides transmission channels for batch data stored in Alibaba cloud services such as MaxCompute, AnalyticDB for PostgreSQL, and Hologres.

**Notice:**

- DataWorks does not support configuring data sources such as DB2, Elasticsearch, HBase, InfluxDB, Kafka, MaxGraph, MetaQ, OpenSearch, OpenTSDB, OTSStream, Prometheus, RDBMS, SAP HANA, Stream, TSDB, and Vertica. Logtail can only be used in script mode to read the corresponding data source information.
- Currently, data in the Excel format cannot be imported. You can modify data in the Excel format to be CSV before importing data.
- Data sources must be connected to networks for data synchronization. For more information, see [Test data store connectivity](#).

Data store	Reader	Writer
AnalyticDB for MySQL 2.0	supported	supported
AnalyticDB for MySQL 3.0	AnalyticDB for MySQL 3.0 Reader	AnalyticDB for MySQL 3.0 Writer
AnalyticDB for PostgreSQL	AnalyticDB for PostgreSQL Reader	AnalyticDB for PostgreSQL Writer
ApsaraDB For Oceanbase <div> Note: Currently, this data source only supports Use exclusive resource groups for data integration.</div>	ApsaraDB For Oceanbase Reader	ApsaraDB For Oceanbase Writer

Data store	Reader	Writer
ClickHouse  Note: Currently, this data source only supports Use exclusive resource groups for data integration .	No	ClickHouse Writer
DataHub	DataHub Reader	DataHub Writer
Db2	DB2 Reader	DB2 Writer
DM	RDBMS Reader	RDBMS Writer
Distribute Relational Database Service (DRDS)	DRDS Reader	DRDS Writer
Elasticsearch	Elasticsearch Reader	Elasticsearch Writer
FTP	FTP Reader	FTP Writer
GBase8a  Note: Currently, this data source only supports Use exclusive resource groups for data integration .	supported	supported
Graph Database(GDB)	No	GDB Writer
HBase  Note: Currently, this data source only supports Use exclusive resource groups for data integration .	HBase Reader	<ul style="list-style-type: none"> HBase Writer HBase 11xsql Writer
HDFS	HDFS Reader	HDFS Writer

Data store	Reader	Writer
Hive  Note: Currently, this data source only supports Use exclusive resource groups for data integration .	Hive Reader	Hive Writer
Hologres  Note: Currently, this data source only supports Use exclusive resource groups for data integration .	supported	supported
HybridDB for MySQL	HybridDB for MySQL Reader	HybridDB for MySQL Writer
InfluxDB	InfluxDB Reader	No
Kafka	Kafka Reader	Kafka Writer
LogHub	LogHub Reader	LogHub Writer
MaxCompute	MaxCompute Reader	MaxCompute Writer
MaxGraph	No	Maxgraph Writer
Memcache	No	Memcache Writer
MetaQ	MetaQ Reader	No
MongoDB	MongoDB Reader	MongoDB Writer
MySQL	MySQL Reader	MySQL Writer
Open Search	No	OpenSearch Writer
OpenTSDB	OpenTSDB Reader	No
Oracle	Oracle Reader	Oracle Writer
OSS	OSS Reader	OSS Writer
OTSSStream	OTSSStream Reader	No
PolarDB	PolarDB Reader	PolarDB Writer
PostgreSQL	PostgreSQL Reader	PostgreSQL Writer
Prometheus	Prometheus Reader	No
RDBMS	RDBMS Reader	RDBMS Writer

Data store	Reader	Writer
Redis	No	Redis Writer
SAP HANA	SAP HANA Reader	No
Stream	Stream Reader	Stream Writer
SQL Server	SQL Server Reader	SQL Server Writer
Table Store	Table Store(OTS)Reader	Table Store(OTS)Writer
TSDB	Supported	TSDB Writer
Vertica	Vertica Reader	Vertica Writer

4.1.2 Connection isolation

The connection isolation feature can be used to isolate data of the development environment from data of the production environment for workspaces in standard mode.

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


**Note:**

Currently, only workspaces in standard mode support the connection isolation feature.

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

The connection isolation feature has the following impacts on workspaces:

- Workspaces in basic mode: The features and configuration dialog boxes of connections are the same as those before the connection isolation feature is added.
- Workspaces in standard mode: The Applicable Environment parameter is added to the configuration dialog boxes of connections.
- Workspaces upgraded from the basic mode to the standard mode: During the upgrade, you will be prompted to upgrade connections. After the upgrade, the connections in the development environment are isolated from those in the production environment.

GUI element	Description
Add Connections button	Currently, you can add only multiple MySQL, SQL Server, or Oracle connections at a time. A template is available for you to add multiple connections at a time. The template contains the connection type, connection name, description, applicable environment (0 for development and 1 for production), and URL. You can download the template, configure multiple connections in the template, and upload the template to add the connections at a time. Details about the connections will appear in the Add Connections dialog box.
Add Connection button	<ul style="list-style-type: none">Connections in the development environment: You can select such a connection when creating a sync node, and then run the sync node in the development environment. However, you cannot commit the sync node to the production environment for running.Connections in the production environment: You can use such a connection only in the production environment. You cannot select such a connection when creating a sync node. <div> Note: The same connection must have the same name in the development and production environments.</div>
Environment column	This column is unavailable for a workspace in basic mode.

GUI element	Description
Actions column	<ul style="list-style-type: none"> • Add: This button appears if no connection is configured for the corresponding environment. Click the button to add a connection for the environment. • Edit and Delete: These buttons appear if a connection is configured for the corresponding environment. Click the buttons to modify or delete the connection. <ul style="list-style-type: none"> - Before deleting a connection for both the development and production environments, check whether the connection is used by any sync node in the production environment. The delete operation cannot be rolled back. After the connection is deleted, you cannot select it when configuring a sync node in the development environment. <p>If a sync node in the production environment uses this connection, the sync node cannot be run after the connection is deleted. Delete the sync node before deleting the connection.</p> - Before deleting a connection for the development environment, check whether the connection is used by any sync node in the production environment. The delete operation cannot be rolled back. After the connection is deleted, you cannot select it when configuring a sync node in the development environment. <p>If a sync node in the production environment uses this connection, you cannot obtain metadata when editing the sync node after the connection is deleted. However, the sync node can be run in the production environment.</p> - Before deleting a connection for the production environment, check whether the connection is used by any sync node in the production environment. If you select this connection when configuring a sync node in the development environment, you cannot commit the sync node to the production environment after the connection is deleted. <p>If a sync node in the production environment uses this connection, the sync node cannot be run after the connection is deleted.</p>

GUI element	Description
Select	Select multiple connections in this column to test the connectivity of the connections or delete them at a time.

4.2 Node configuration

4.2.1 Create a sync node by using the codeless UI

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

Development process

1. Create connections.
2. Create a batch synchronization node.
3. Select a source connection.
4. Select a destination connection.
5. Map the fields in the source and destination tables.
6. Configure channel control policies, such as the maximum transmission rate and the maximum number of dirty data records allowed.
7. Configure the node properties.

Create connections

A sync node can synchronize data between various homogeneous and heterogeneous data stores. In the DataWorks console, click the **Workspace Manage** icon in the upper-right corner. On the page that appears, click **Data Source** and add a connection. For more information, see [Add connections](#).

After a connection is added, you can directly select it when configuring a sync node on the DataStudio page. For more information about connection types supported by Data Integration, see [Supported data stores and plug-ins](#).




Note:

- Data Integration does not support connectivity testing for some connection types. For more information, see [Test data store connectivity](#).
- If a data store is deployed on the premises and does not have a public IP address or cannot be directly connected over a network, the connectivity test fails when you configure the connection. Data Integration allows you to resolve this issue by using

a custom resource group. For more information, see [Add a custom resource group](#).

When a data store cannot be directly connected over a network, Data Integration cannot obtain the table schema. In this case, you can only create a sync node for this data store by using the code editor.

Create a workflow

1. Log on to the [DataWorks console](#). In the left-side navigation pane, click Workspaces. On the Workspaces page, find the target workspace and click **Data Analytics** in the Actions column.
2. On the **DataStudio** page that appears, move the pointer over the  icon and click **Workflow**.
3. In the **Create Workflow** dialog box that appears, set **Workflow Name** and **Description**.
4. Click **Create**.

Create a batch synchronization node

1. Click the workflow to show its content and right-click **Data Integration**.
2. Choose **Create > Batch Synchronization**.
3. In the **Create Node** dialog box that appears, set **Node Name** and **Location**.
4. Click **Commit**.

Select a source connection

After the sync node is created, you can configure the source connection and source table as needed.

01 Connections

Source

The connections can be default

* Connection OSS

* Object Name Prefix user_log.txt Add

* File Type text

* Field Delimiter |

Encoding UTF-8

Null String Enter a sting that represents null.

* Compression None

Format

* Include Header No

Preview

**Note:**

- For more information about how to configure the source connection, see [Reader configuration](#).
- Incremental data synchronization is required when you configure the source connection for some sync nodes. In this case, you can use the [scheduling parameters](#) of DataWorks to obtain the date and time required by incremental data synchronization.

Select a destination connection

After the source settings are completed, you can configure the destination connection and destination table as needed.

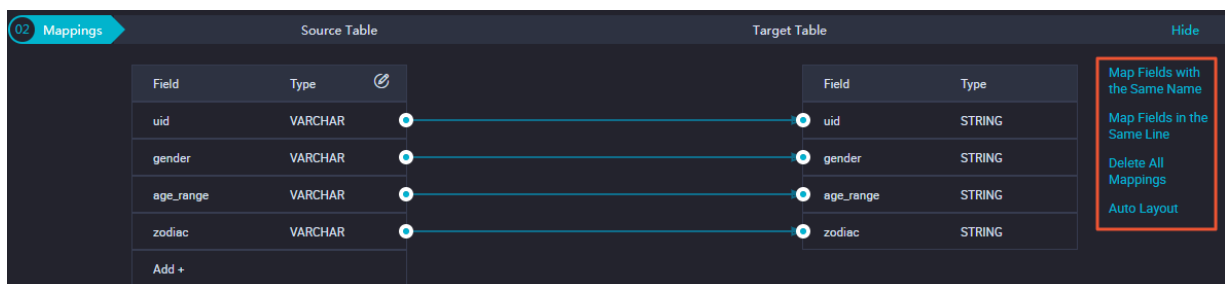
**Note:**

- For more information about how to configure the destination connection, see [Writer configuration](#).

- 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 selecting the source and destination connections, you must specify the mappings between 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**, or **Auto Layout** to perform related operations.



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.
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	<ul style="list-style-type: none"> 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 <code>\${bizdate}</code>. You can enter functions supported by relational databases, such as <code>now()</code> and <code>count(1)</code>. 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, you can continue to configure the channel control policies of the sync node.

03 Channel

You can control the sync process by throttling the bandwidth or limiting the dirty data records allowed. [Learn more.](#)

* Expected Concurrency ?

* Bandwidth Throttling ☐ Disable ☒ Enable MB/s

Dirty Data Records Allowed dirty records, task ends. ?

Resource Group

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read 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. If a large number of nodes including this sync node are deployed on the default resource group, the sync node may need to wait for resources. We recommend that you purchase an exclusive resource group for data integration or add a custom resource group. For more information, see #unique_15 and Add a custom resource group .

Configure the node properties

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

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

You can specify the scheduling parameters by using **\${Variable name}**. After a variable is specified, 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 [#unique_116](#).

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

4.2.2 Create a sync node by using the code editor

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

Development process

To create a sync node by using the code editor, follow these steps:

1. Create connections.
2. Create a batch synchronization node.
3. Apply a template.
4. Configure a reader for the sync node.
5. Configure a writer for the sync node.
6. Map the fields in the source and destination tables.
7. Configure channel control policies, such as the maximum transmission rate and the maximum number of dirty data records allowed.
8. Configure the node properties.

Create connections

A sync node can synchronize data between various homogeneous and heterogeneous data stores. In the DataWorks console, click the **Workspace Manage** icon in the upper-right corner. On the page that appears, click **Data Source** and add a connection. For more information, see [Add connections](#).

After a connection is added, you can directly select it when configuring a sync node on the DataStudio page. For more information about connection types supported by Data Integration, see [Supported data stores and plug-ins](#).


**Note:**

- Data Integration does not support connectivity testing for some connection types. For more information, see [Test data store connectivity](#).
- If a data store is deployed on the premises and does not have a public IP address or cannot be directly connected over a network, the connectivity test fails when you configure the connection. Data Integration allows you to resolve this issue by using a custom resource group. For more information, see [Add a custom resource group](#).

**Note:**

When a data store cannot be directly connected over a network, Data Integration cannot obtain the table schema. In this case, you can only create a sync node for this data store by using the code editor.

Create a workflow

1. Log on to the [DataWorks console](#). In the left-side navigation pane, click Workspaces. On the Workspaces page, find the target workspace and click **Data Analytics** in the Actions column.
2. On the **DataStudio** page that appears, move the pointer over the  icon and click **Workflow**.
3. In the **Create Workflow** dialog box that appears, set **Workflow Name** and **Description**.
4. Click **Create**.

Create a batch synchronization node

1. Click the workflow to show its content and right-click **Data Integration**.
2. Choose **Create > Batch Synchronization**.
3. In the **Create Node** dialog box that appears, set **Node Name** and **Location**.
4. 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 user interface (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 a reader for the sync node

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

```
{
  "type": "job",
  "version": "2.0",
  "steps": [ // Do not modify the preceding lines. They indicate the header code of the
    {
      "stepType": "mysql",
      "parameter": {
        "datasource": "MySQL",
        "column": [
          "id",
          "value",
          "table"
        ],
        "socketTimeout": 3600000,
        "connection": [
          {
            "datasource": "MySQL",
            "table": [
              "`case`"
            ]
          }
        ],
        "where": "",
        "splitPk": "",
        "encoding": "UTF-8"
      },
      "name": "Reader",
      "category": "reader" // Specifies 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, see [Reader Configuration](#).
- Incremental data synchronization is required when you configure the source connection for some sync nodes. In this case, you can use the [scheduling parameters](#) of DataWorks to obtain the date and time required by incremental data synchronization.

Configure a writer for the sync node

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

```
{
  "stepType": "odps",
  "parameter": {
    "partition": "",
    "truncate": true,
    "compress": false,
    "datasource": "odps_first",
    "column": [
      "*"
    ],
    "emptyAsNull": false,
    "table": ""
  },
  "name": "Writer",
  "category": "writer" // Specifies that these settings are related to the writer.
},
],
```

**Note:**

- For more information about how to configure the destination connection, see [Writer Configuration](#).
- 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, you can continue to configure the channel control policies of the sync node. The **setting** parameter specifies node efficiency parameters, including the number of concurrent threads, 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.
  }
},
```

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read 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	<p>You can specify a resource group by clicking Resource Group in the upper-right corner of the configuration tab of the sync node.</p> <p>The resource group used for running the sync node. If a large number of nodes including this sync node are deployed on the default resource group, the sync node may need to wait for resources. We recommend that you purchase an exclusive resource group for data integration or add a custom resource group. For more information, see #unique_15 and Add a custom resource group.</p>

Configure the node properties

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

On the **DataStudio** page, double-click the target batch synchronization node in the workflow. On the node configuration tab that appears, click the **Properties** tab in the right-side navigation pane to configure the node properties.

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

After the sync node is configured, save and commit the node. For more information about how to configure the node properties, see [Schedule](#).

4.3 Synchronize incremental data

This topic describes how to synchronize incremental data from a Relational Database Service (RDS) database to MaxCompute. You can refer to this topic to synchronize incremental data in different scenarios.

Context

Based on whether the data to be synchronized is subject to changes after being written, we can divide the data into unchanged historical data, generally the log data, and dynamically updated data, such as changes of the staff status in the staff table.

If the running results of a node remain the same when you run the node multiple times, you can schedule the node to rerun it. If an error occurs in the node, you can clear dirty data easily. This principle is called idempotence. According to this principle, each time when you write data, the data is written to a separate table or partition or overwrites the historical data in an existing table or partition.

In this topic, the running date of a sync node is set to November 14, 2016 and historical data is synchronized to the ds=20161113 partition on the same day. In the incremental synchronization scenario, automatic scheduling is configured to synchronize incremental data to the ds=20161114 partition in the early morning on November 15. The **optime** field indicates the data modification time and is used to determine whether data is incremental data.

Create a workflow

1. Log on to the [DataWorks console](#). In the left-side navigation pane, click **Workspaces**. On the **Workspaces** page, find the target workspace and click **Data Analytics** in the **Actions** column.
2. In the **Data Analytics** section, right-click **Business Flow** and select **Create Workflow**.
3. In the **Create Workflow** dialog box that appears, set **Workflow Name** and **Description**.
4. Click **Create**.

Synchronize unchanged historical data in incremental mode

Historical data does not change after it is generated. Therefore, you can easily partition a table based on the pattern in which data is generated. Typically, you can partition a table by date, such as generating one partition per day.

1. Execute the following statements in the RDS database to prepare data:

```
drop table if exists oplog;  
create table if not exists oplog(  
  optime DATETIME,  
  uname varchar(50),  
  action varchar(50),  
  status varchar(10)  
);  
Insert into oplog values(str_to_date('2016-11-11','%Y-%m-%d'),'LiLei','SELECT','SUCCESS');  
Insert into oplog values(str_to_date('2016-11-12','%Y-%m-%d'),'HanMM','DESC','SUCCESS');
```

The preceding two data entries are used as historical data. You must synchronize all the historical data to the ds=20161113 partition first.

2. In the **Data Analytics** section, expand the created workflow, right-click **Table** under **MaxCompute**, and then select **Create Table**.
3. In the **Create Table** dialog box that appears, set **Table Name** to ods_oplog and click **Commit**.
4. On the editing tab of the ods_oplog table on the right side, click **DDL Statement**. In the **DDL Statement** dialog box that appears, enter the following statement for creating a MaxCompute table:

```
-- Create a MaxCompute table and partition the table by day.  
create table if not exists ods_oplog(  
  optime datetime,  
  uname string,  
  action string,  
  status string
```

```
) partitioned by (ds string);
```

5. Configure a sync node to synchronize historical data. For more information, see [#unique_11](#).

After you test the sync node, click the **Properties** tab on the right side of the node editing tab. In the **Properties** dialog box, select **Skip Execution** and commit or publish the node again to prevent the node from being automatically scheduled to run.

6. Execute the following statements to insert data into the RDS source table as incremental data:

```
insert into oplog values(CURRENT_DATE,'Jim','Update','SUCCESS');
insert into oplog values(CURRENT_DATE,'Kate','Delete','Failed');
insert into oplog values(CURRENT_DATE,'Lily','Drop','Failed');
```

7. Configure a sync node to synchronize incremental data.

In the **Source** section, set **Filter** to `date_format(optime,'%Y%m%d')=${bdp.system.bizdate}`. In the **Target** section, enter `${bdp.system.bizdate}` in the **Partition Key Column** field.



Note:

By setting a data filter, you can query data inserted to the source table on November 14 and synchronize the data to the incremental data partition of the destination table in the early morning on November 15.

8. View the incremental synchronization result.

Click the **Properties** tab on the right side. In the **Properties** dialog box, set **Instance Recurrence** to **Day**. After you commit or publish the incremental sync node, the node is automatically scheduled to run from the next day. After the node is run, you can view data in the destination MaxCompute table.

Synchronize dynamically updated data in incremental mode

Based on the time-variant characteristic of data warehouses, we recommend that you daily synchronize all data in tables that are subject to changes, such as staff and order tables. In other words, full data is stored daily. In this way, you can retrieve historical and current data easily.

In actual scenarios, you only need to synchronize incremental data every day under special circumstances. MaxCompute does not support editing data with the UPDATE statement. Therefore, you can only use other methods to synchronize data. The following section describes how to synchronize data in full mode and in incremental mode.

1. Execute the following statements to prepare data:

```
drop table if exists user ;
create table if not exists user(
  uid int,
  uname varchar(50),
  deptno int,
  gender VARCHAR(1),
  optime DATETIME
);
-- Insert historical data.
insert into user values (1,'LiLei',100,'M',str_to_date('2016-11-13','%Y-%m-%d'));
insert into user values (2,'HanMM',null,'F',str_to_date('2016-11-13','%Y-%m-%d'));
insert into user values (3,'Jim',102,'M',str_to_date('2016-11-12','%Y-%m-%d'));
insert into user values (4,'Kate',103,'F',str_to_date('2016-11-12','%Y-%m-%d'));
insert into user values (5,'Lily',104,'F',str_to_date('2016-11-11','%Y-%m-%d'));
-- Insert incremental data.
update user set deptno=101,optime=CURRENT_TIME where uid = 2; -- Change null to non-null.
update user set deptno=104,optime=CURRENT_TIME where uid = 3; -- Change non-null to non-null.
update user set deptno=null,optime=CURRENT_TIME where uid = 4; -- Change non-null to null.
delete from user where uid = 5;
insert into user(uid,uname,deptno,gender,optime) values (6,'Lucy',105,'F',CURRENT_TIME);
```

2. Synchronize data.

- Daily synchronize all data.
 - a. Execute the following statement to create a MaxCompute table. For more information about how to create a MaxCompute table, see [#unique_120/unique_120_Connect_42_section_lgp_ld4_q2b](#).

```
-- Synchronize all data.
create table ods_user_full(
  uid bigint,
  uname string,
  deptno bigint,
  gender string,
  optime DATETIME
) partitioned by (ds string);ring);
```

- b. Configure a sync node to synchronize all data.



Note:

Set **Instance Recurrence** to **Day** because daily full synchronization is required.

- c. Run the sync node and view data in the destination MaxCompute table after the synchronization is completed.

When full synchronization is performed on a daily basis, no incremental synchronization is performed. You can view the data results in the table after the node is automatically scheduled to run on the next day.

- Daily synchronize incremental data.

We recommend that you do not use this sync mode except in scenarios where the DELETE statement is not supported and you fail to execute relevant SQL statements to view deleted data. Generally, your enterprise code is deleted logically, in which the UPDATE statement is applied instead of the DELETE statement. In scenarios where this method is inapplicable, using this sync mode may cause data inconsistency when a special condition is encountered. Another drawback is that you must merge new and historical data after the synchronization.

Prepare data

Create two tables, one for writing all the latest data and the other for writing incremental data.

```
-- Create a result table.  
create table dw_user_inc(  
    uid bigint,  
    uname string,  
    deptno bigint,  
    gender string,  
    optime DATETIME  
);
```

```
-- Create an incremental data table.  
create table ods_user_inc(  
    uid bigint,  
    uname string,  
    deptno bigint,  
    gender string,  
    optime DATETIME  
)
```

- a. Configure a sync node to write all data to the result table.



Note:

You need to run the node only once. After running the node, click the **Properties** tab on the right side. In the **Properties** dialog box, select **Skip Execution**.

- b. Configure a sync node to write incremental data to the incremental data table. Set the data filter to `date_format(optime,'%Y%m%d')=${bdp.system.bizdate}`.
- c. Execute the following statement to merge data:

```
insert overwrite table dw_user_inc
select
-- The following lists all the SELECT clauses. If the incremental data table
contains data, data in the result table changes. In this case, use data in the
incremental data table.
case when b.uid is not null then b.uid else a.uid end as uid,
case when b.uid is not null then b.uname else a.uname end as uname,
case when b.uid is not null then b.deptno else a.deptno end as deptno,
case when b.uid is not null then b.gender else a.gender end as gender,
case when b.uid is not null then b.optime else a.optime end as optime
from
dw_user_inc a
full outer join ods_user_inc b
on a.uid = b.uid ;
```

View the merge result. It is found that the deleted data entry is not synchronized.

Daily incremental synchronization is advantageous in that it synchronizes only a small amount of incremental data. However, it may cause data inconsistency, requiring an extra computing workload to merge data.

If not necessary, daily synchronize dynamically updated data in full mode. In addition, you can set a lifecycle for the historical data so that it can be automatically deleted after being retained for a certain period.

4.4 Migrate databases to MaxCompute

4.4.1 Rules and restrictions

This topic describes the restrictions on using the database migration feature to migrate databases to MaxCompute, rules for generating sync nodes, and supported database types.

The database migration feature allows you to quickly upload all tables in a database to MaxCompute in an efficient and cost-effective manner. This saves the time spent on creating multiple nodes one by one to migrate your initial data to the cloud.

For example, if your database contains 100 tables, the conventional method may require you to configure 100 sync nodes. With database migration, you can migrate all the tables

at a time. However, some tables may fail to be migrated due to the restrictions of table design specifications of databases.

Restrictions on database migration

In consideration of table design specifications of databases, note the following restrictions on using the database migration feature:

- Currently, you can only migrate the following types of databases to MaxCompute:
[MySQL](#), [PostgreSQL](#), [SQL Server](#), [Distributed Relational Database Service \(DRDS\)](#), [POLARDB](#), [AnalyticDB for PostgreSQL](#), [HybridDB for MySQL](#), [AnalyticDB for MySQL 3.0](#), [Oracle](#), and [Dameng \(DM\)](#).
- You can only upload incremental or full data on a daily basis.

You cannot use the database migration feature to migrate all historical data at a time. To migrate historical data, we recommend that you perform the following operations:

- Configure a scheduled node to migrate data on a daily basis. You can also create retroactive node instances to transmit historical data. This eliminates the need to execute temporary SQL statements to split data into partitions after all the historical data is migrated.
- If you need to migrate all the historical data at a time, configure a sync node in DataStudio and click the **Run** icon to run the node. Daily incremental migration and daily full migration are both one-time operations. After the migration is completed, you must execute SQL statements to convert the data.

If your daily incremental migration node is subject to special business logic where incremental data cannot be simply identified by a date field, you cannot use the database migration feature. In this case, we recommend that you perform the following operations:

- Add a field that indicates the data change timestamp to your database. Generally, incremental data can be identified based on the binary logs provided by Data Transmission Service (DTS) or the date field that indicates the data change timestamp. Currently, Data Integration identifies incremental data based on the date field that indicates the data change timestamp. Therefore, your database must contain such a

field. Data Integration can use this date field to detect whether the data is modified on the same day as the data timestamp. If yes, all modified data is synchronized.

- To facilitate incremental upload, verify that your tables contain the **gmt_create** and **gmt_modify** fields. In addition, we recommend that you add the ID field as a primary key to improve upload efficiency.
- Database migration supports data upload at a time and data upload in batches. Data upload in batches is to upload a part of data at the specified interval each time. Currently, Data Integration does not protect database connection pools.
 - We recommend that you upload tables in batches at the specified interval to protect your database from being overloaded and avoid affecting the business.
 - If you have primary and secondary databases, we recommend that you upload data of the secondary database.
 - During data upload in batches, each table requires a database connection with the maximum data transmission rate of 1 Mbit/s. If you upload 100 tables at a time, 100 database connections are established. We recommend that you determine the number of concurrent threads based on your business requirements.
 - The maximum data transmission rate of all sync nodes is 1 Mbit/s, which may fail to meet your specific requirements for data transmission efficiency.
- Database migration only supports mapping between the names, field names, and data types of the source and destination tables.

During database migration, Data Integration automatically creates MaxCompute tables. The partition field of the tables is **pt**, which is of the **string** type and is in the **yyyymmdd** format.

**Note:**

After you select the tables to be uploaded, all fields in the tables are uploaded and you cannot edit the fields.

Rules for generating sync nodes

After you set parameters for database migration, Data Integration creates MaxCompute tables and generates corresponding sync nodes in sequence based on the tables you select .

The names, field names, and data types of MaxCompute tables are determined based on the advanced settings. If you do not configure advanced settings, the names, field names,

and data types are the same as those of the source tables. The partition field of the tables is **pt**, which is in the **yyyymmdd** format.

The generated sync nodes are daily scheduled and start to run automatically in the early morning from the next day with the maximum data transmission rate of 1 Mbit/s. The nodes may vary by synchronization method and concurrency. You can expand the created workflow such as **clone_database** in DataStudio, find the target node, for example, **mysql2odps_Table name** under **Data Integration**, and then edit the node as required.

4.4.2 Migrate a MySQL database

This topic describes how to migrate a MySQL database to MaxCompute.

The database migration feature allows you to quickly upload all tables in a MySQL database to MaxCompute in an efficient and cost-effective manner. For more information, see [Rules and restrictions](#).

Procedure

1. Log on to the [DataWorks console](#). In the left-side navigation pane, click Workspaces. On the Workspaces page, find the target workspace and click **Data Integration** in the Actions column.
2. On the **Data Integration** page, click **Connection** in the left-side navigation pane. The **Data Source** page appears.
3. Click **Add Connection** in the upper-right corner and add a MySQL connection named clone_database for database migration. For more information, see [Configure a MySQL connection](#).
4. Go back to the **Data Integration** page and click **Migrate Database** in the left-side navigation pane.
5. On the Migrate Database page, find the added MySQL connection and click **Migrate Tables** in the Actions column.

The database migration settings page consists of three functional modules.

No.	Functional module	Description
1	Tables to migrate	This module lists all the tables in the MySQL connection named clone_data base. Selected tables will be migrated.

No.	Functional module	Description
2	Advanced settings	You can configure the rules for converting the table names, column names, and data types between MySQL and MaxCompute data tables.
3	Basic settings	<p>You can specify whether to synchronize full or incremental data on a daily basis, whether to upload data in one or more batches, and the synchronization frequency and efficiency. You can also view the migration progress and results of each table after committing the sync node.</p> <p>If you select Synchronize Incremental Data Daily, you must set the Incremental configuration mode parameter. Valid values:</p> <ul style="list-style-type: none"> • Incremental field: Data Integration automatically generates a WHERE clause based on the specified field. • Where condition for incremental extraction?: Explicitly specify a WHERE clause to extract incremental data.

6. 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.
7. 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 Java Database Connectivity (JDBC) and executing 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. Configure Data Integration to start data synchronization for three tables every one hour from 00:00 each day.

Click **Commit Sync Node**. Then, you can view the migration progress and results of each table.

8. 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.

The log in the following figure shows that table a1 is migrated successfully.

4.4.3 Migrate an Oracle database

This topic describes how to migrate an Oracle database to MaxCompute.

Context

The database migration feature allows you to quickly upload all tables in an Oracle database to MaxCompute in an efficient and cost-effective manner. For more information, see [Rules and restrictions](#).

Procedure

1. Log on to the [DataWorks console](#). In the left-side navigation pane, click **Workspaces**. On the **Workspaces** page, find the target workspace and click **Data Integration** in the **Actions** column.
2. On the **Data Integration** page, click **Connection** in the left-side navigation pane. The **Data Source** page appears.
3. Click **Add Connection** in the upper-right corner and add an Oracle connection named clone_database for database migration. For more information, see [Configure an Oracle connection](#).

4. Go back to the **Data Integration** page and click **Migrate Database** in the left-side navigation pane.
5. On the **Migrate Database** page, find the added Oracle connection and click **Migrate Tables** in the **Actions** column.

The database migration settings page consists of three functional modules.

No.	Functional module	Description
1	Tables to migrate	This module lists all the tables in the Oracle connection named clone_database. Selected tables will be migrated.
2	Advanced settings	You can configure the rules for converting the table names, column names, and data types between Oracle and MaxCompute data tables .
3	Basic settings	<p>You can specify whether to synchronize full or incremental data on a daily basis, whether to upload data in one or more batches, and the synchronization frequency and efficiency. You can also view the migration progress and results of each table after committing the sync node.</p> <p>If you select Synchronize Incremental Data Daily, you must set the Incremental configuration mode parameter. Valid values:</p> <ul style="list-style-type: none">• Incremental field: Data Integration automatically generates a WHERE clause based on the specified field.• Where condition for incremental extraction?: Explicitly specify a WHERE clause to extract incremental data.

6. Click **Advanced Settings** and configure conversion rules based on your needs.
7. 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. Configure Data Integration to start data synchronization for three tables every one hour from 00:00 each day.

Click **Commit Sync Node**. Then, you can view the migration progress and results of each table.

8. 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_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.

4.5 Migrate multiple tables to the cloud at a time

4.5.1 Migrate multiple tables to the cloud at a time

Data Integration allows you to migrate multiple tables to the cloud at a time in an efficient and cost-effective manner. This topic describes how to migrate multiple tables to the cloud at a time.

Context

You can quickly upload all tables in MySQL, Oracle, or SQL Server databases to MaxCompute at a time. This saves the time spent on creating multiple nodes one by one to migrate your initial data to the cloud.

Based on your business requirements, you can also configure rules to convert table names, field names, and data types, add fields to the destination tables, specify values for fields in the destination tables, filter data, and add a prefix to the names of the destination tables.

Procedure

1. Go to the **Sync Tables page of **Data Integration**.**

- a) Log on to the [DataWorks console](#) as a developer. In the left-side navigation pane, click **Workspaces**. On the **Workspaces** page, find the target workspace and click **Data Integration** in the **Actions** column.
- b) On the **Data Integration** page, click **Sync Tables** in the left-side navigation pane. On the **Sync Tables** page that appears, you can view all the configured sync nodes.

**Note:**

- On the **Sync Tables** page, you can view the related logs and synchronization rules, but you cannot modify them.
- If you do not commit the sync nodes after submitting synchronization rules, no running time appears for the nodes and the rules do not take effect.

2. Click **Sync Tables in the upper-right corner.****3. On the page that appears, select the source and destination connections.**

Select the added source connections and destination MaxCompute connection. You can select multiple source connections of the same type, such as MySQL, Oracle, or SQL Server. For more information about how to add multiple connections at a time, see [Add multiple connections at a time](#).

4. Configure the synchronization rules.

Click **Add Rule**, select rules from the drop-down list as required, and configure the rules. After configuring the rules, click **Apply Rules**. Then check the DDL statements and synchronization code to confirm the rule effects.

**Note:**

- If rules in the drop-down list do not meet your requirements, you can click **Switch to Code Editor** and configure rules in the code editor.
- After configuring the rules, you must run the rules and commit the sync nodes. Otherwise, the rules are not saved after you refresh or close this page.

Button	Parameter	Description
Add Rule	Partition Key Field in Target Table	The rule for specifying the format of partition fields in the destination tables based on scheduling parameters. For more information, see #unique_116 .

Button	Parameter	Description
	Change Table Names	The rule for converting table names. You can convert names of the source tables to required names and use them for the destination tables.
	Change Field Names	The rule for converting field names. You can convert field names in the source tables to required names and use them in the destination tables.
	Convert Field Types	The rule for converting data types. You can convert data types in the source tables to required data types and use them in the destination tables.
	Add Fields to Target Table	The rule for adding fields to the destination MaxCompute tables. You can add a field to the destination tables and set the field name as required.
	Assign Values in Target Table	The rule for assigning a value to the added field.
	Filter Data	The rule for filtering data in the source tables of the selected source connections.
	Target Table Name Prefix	The rule for adding a prefix to the names of the destination tables.
Switch to Code Editor	You can switch to the code editor to configure synchronization rules. Compared with the codeless user interface (UI), you can specify the applicable scope of each rule in the code editor. However, you cannot switch back to the codeless UI after you switch to the code editor.	
Reset Script	You can reset code only after you switch to the code editor. After you click this button, a unified code template appears in the code editor.	
Apply Rules	<p>You can click Apply Rules to view the impacts of the rules on DDL statements and synchronization code. No sync node is created after you click this button. You can only preview DDL statements and synchronization code.</p> <p>You can select some tables to check whether their DDL statements and synchronization code comply with the rules.</p>	

5. Select the tables to be synchronized and commit the sync nodes.

You can select multiple tables at a time. Data Integration generates destination MaxCompute tables based on the configured synchronization rules. If the synchronization fails, move the pointer over the synchronization result. Then the failure cause appears.

Button	Description
DDL	You can click this button to view the related DDL table creation statements, but you cannot modify them.
Sync Settings	You can click this button to view code of a sync node in the code editor.
View Table	You can click this button to view the details of a destination MaxCompute table.
Tasks	After committing a sync node, you can click this button to go to the DataStudio page and view the node under the corresponding workflow.

4.5.2 Add multiple connections at a time

This topic describes how to add multiple connections at a time.




Note:

- Currently, you can only add multiple MySQL, Oracle, or SQL Server connections at a time.
- You can only set the connection type to **JDBC Connection Mode**.
- After adding MySQL, Oracle, or SQL Server connections, you must test connectivity. If the connectivity test is passed, you can select the connections when you configure sync nodes to migrate tables to the cloud.

1. Log on to the [DataWorks console](#) as a workspace administrator. In the left-side navigation pane, click **Workspaces**. On the **Workspaces** page, find the target workspace and click **Data Integration** in the **Actions** column.
2. On the **Data Integration** page, click **Connection** in the left-side navigation pane. The **Data Source** page appears.
3. On the **Data Source** page, click **Add Connections** in the upper-right corner.

4. In the **Add Connections** dialog box that appears, select a connection type, click **Select File**, and then upload the relevant file. In this topic, MySQL connections are added.

Parameter or button	Description
Connect To	The type of the connection. You can only set the value to JDBC Connection Mode .
Upload Script	<p>Click Download Template. In the template, enter the name, description, JDBC URL, username, and password of each connection to add.</p> <div> Note: The template contains information about a default connection mysql_001_di_test. You can delete the information and add your connections.</div>
Select File	Click Select File and select the modified template.
Create	After the file is uploaded, click Create . The upload results, including the number of added connections, number of connections that fail to be added, and failure causes, appear in the text box at the bottom.

5. After the file is uploaded, click **Close**.

6. On the **Data Source** page, select the added connections and click **Test Connection**.

**Note:**

You can use the connections to migrate multiple tables to the cloud only after the connectivity test is passed.

5 Real-time synchronization


5.1 Create, commit, and manage real-time sync nodes

DataWorks supports real-time data synchronization. This topic describes how to create, commit, and manage real-time sync nodes.

Prerequisites

The real-time synchronization feature is activated. Currently, the real-time synchronization feature is in the phased release stage. To activate this feature, [submit a ticket](#).

Create a real-time sync node

1. Log on to the [DataWorks console](#).
2. In the left-side navigation pane, click **Workspaces**.
3. Select the region of the workspace, and click **Data Analytics**.
4. On the Data Analytics tab, move the pointer over the  icon and choose **Data Integration > Real-Time Sync**.

You can also find the target workflow, right-click **Data Integration**, and then choose **Create > Real-Time Sync**.

5. In the **Create Node** dialog box that appears, set **Node Name** and **Location**.

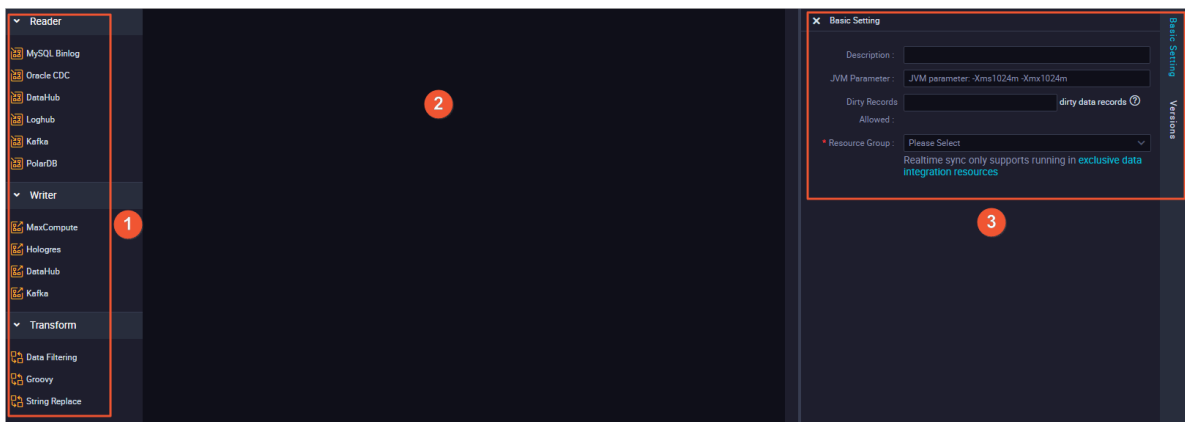
**Note:**

A node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is case-insensitive.

6. Click **Commit**.

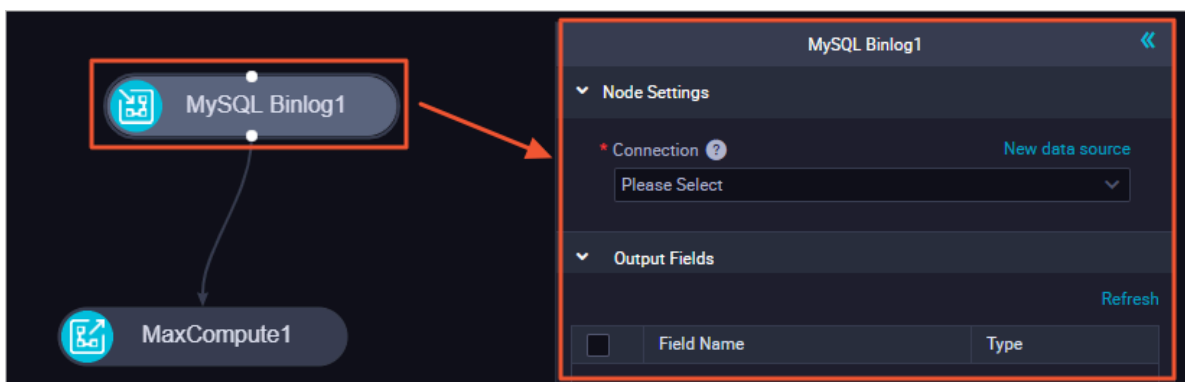
Commit a real-time sync node


1. Open the configuration tab of a real-time sync node.



Panel	Description
1	The component panel. This panel consists of the Reader , Writer , and Transform sections.
2	The editing panel of the real-time sync node. You can drag and drop components from the component panel to the editing panel.
3	The property configuration panel of a component. This panel appears after you click a component or Basic Settings on the right side of the configuration tab.

2. Drag and drop components from the component panel to the editing panel, and connect the component nodes in the editing panel. Data will be synchronized from upstream component nodes to downstream component nodes based on the connection.
3. Click a component node. In the pane that appears on the right, set the required parameters in the **Node Settings** section. For more information about the configuration of each type of component nodes, see [Supported data stores](#).



4. After you configure all the component nodes, click the  icon in the toolbar.

5. In the **Commit** dialog box that appears, select the component nodes to be committed, set **Description**, and then select **Ignore I/O Inconsistency Alerts**.

6. Click **Commit**.

In a workspace in the standard mode, you need to click **Publish** in the upper-right corner after you commit the real-time sync node. For more information, see [Deploy a node](#).

Manage a real-time sync node

1. After a real-time sync node is committed or published, click **Operation Center** in the upper-right corner.
2. On the page that appears, choose **RealTime Task Maintenance > RealTime DI** in the left-side navigation pane.
3. On the **RealTime DI** page that appears, find the target node, click the node name, and then view the O&M details about the node. You can also click **Start**, **Stop**, and **Reset Start Offset** in the Actions column of the node.

Operation	Description
Start	Start a node that is not running.
Stop	Stop a running node.
Reset Start Offset	Set the next startup time for a node.

5.2 Supported data stores

Data Integration supports the following types of plug-ins for real-time synchronization: **reader**, **writer**, and **transformation**.



Note:

Currently, the real-time synchronization feature is in phased release stage. To activate this feature, [submit a ticket](#).

Type	Plug-in	Reference
Reader	MySQL binlog reader	MySQL binlog reader
	Oracle Change Data Capture (CDC) reader	Oracle CDC reader
	Datahub reader	Datahub reader
	LogHub reader	LogHub reader

Type	Plug-in	Reference
Writer	Kafka reader	Kafka reader
	Datahub writer	Datahub writer
	Kafka writer	Kafka writer
Transformation	Data filtering	Data filtering
	Groovy	Groovy
	String replacement	String replacement


**Note:**

- You cannot run a real-time sync node on the node configuration tab. Instead, you must run a real-time sync node in the production environment after saving and committing the node.
- Real-time sync nodes can only run on exclusive resource groups for data integration. For more information, see [#unique_15/unique_15_Connect_42_section_hcx_l05_vcg](#).

Basic settings

After you specify the reader, writer, and transformation plug-ins to create a real-time sync node, you can click the **Basic Setting** tab in the right-side navigation pane to configure this sync node.

Parameter	Description
Description	The description of the real-time sync node.
JVM Parameter	The Java virtual memory (JVM) allocated for the real-time sync node. If the value is empty, Data Integration automatically allocates JVM based on your node settings.
Dirty Records Allowed	The maximum number of dirty data records allowed. If you set this parameter to 0, no dirty data records are allowed. If the value is empty, the node continues no matter whether dirty data records exist.

Parameter	Description
Resource Group	<p>The exclusive resource group for data integration on which the real-time sync node is run.</p> <div>  Note: For more information about exclusive resource groups for data integration, see #unique_15/unique_15_Connect_42_section_hcx_l05_vcg. </div>

5.3 Fields used for real-time synchronization

This topic describes the fields that Data Integration uses to synchronize data in real time.

The following table describes the format of a data record synchronized by Data Integration from a MySQL or Oracle database in real time.

_log_file_name_offset_	_operation_type_	_execute_time_	_before_image_	_after_image_	Field 1	Field 2	Field 3
The position of the data record.	The type of the operation. Valid values: I, D, and U.	The timestamp when the data record is generated.	Indicates whether the data record stores the original data. Valid values: Y and N.	Indicates whether the data record stores the updated data. Valid values: Y and N.	Field 1 in the source database.	Field 2 in the source database.	Field 3 in the source database.

When Data Integration synchronizes data from relational databases such as MySQL and Oracle databases to DataHub or Kafka in real time, Data Integration adds five fields to data records in the destination data store. These fields are used for operations such as metadata management, sorting, and de-duplication. The following table describes the fields that Data Integration adds to the destination data store.

Field	Data type	Description
_log_file_name_offset_	STRING	The position of the synchronized data record in the binary log file. It consists of the name of the binary log file and the offset of the data record.
_operation_type_	STRING	The type of the operation. Valid values: <ul style="list-style-type: none"> • I: INSERT. • D: DELETE. • U: UPDATE.
_execute_time_	LONG	The timestamp in the binlog file, indicating when the data record was generated.
_before_image_	STRING	Indicates whether the data record stores the original data. Valid values: Y and N.
_after_image_	STRING	Indicates whether the data record stores the updated data. Valid values: Y and N.

In incremental data records generated for the INSERT, UPDATE, and DELETE operations, the **_before_image_** and **_after_image_** fields are set as follows:

- **INSERT:** An incremental data record is generated after an INSERT operation. This data record stores the new data. For this data record, the value of **_before_image_** is N, whereas the value of **_after_image_** is Y.
- **UPDATE:** Two incremental data records are generated for an UPDATE operation. One stores the original data, and the other stores the updated data. The two data records have the same values for **_log_file_name_offset_**, **_operation_type_**, and **_execute_time_**.

For the data record that stores the original data, the value of **_before_image_** is Y, whereas the value of **_after_image_** is N. For the data record that stores the updated data, the value of **_before_image_** is N, whereas the value of **_after_image_** is Y.


- **DELETE:** An incremental data record is generated after the DELETE operation. This data record stores the original data. For this data record, the value of **_before_image_** is Y, whereas the value of **_after_image_** is N.

5.4 Reader

5.4.1 MySQL binlog reader

A MySQL binlog reader reads data from tables in your MySQL database in real time based on a real-time binlog subscription.

Create a MySQL binlog reader

1. Log on to the [DataWorks console](#). In the left-side navigation pane, click Workspaces. On the Workspaces page, find the target workspace and click **Data Analytics** in the Actions column.
2. On the Data Analytics tab, move the pointer over the  icon and choose **Data Integration > Real-Time Sync**.

You can also find the target workflow, right-click **Data Integration**, and choose **Create > Real-Time Sync**.
3. In the **Create Node** dialog box that appears, set **Node Name** and **Location**, and then click **Commit**.
4. On the configuration tab of the real-time sync node, drag **MySQL Binlog** under **Reader** to the editing panel.
5. Click the **MySQL Binlog** node and set parameters in the **Node Settings** section.

Parameter	Description
Connection	The connection to the MySQL binlog data store. In this example, you can only select a MySQL binlog connection. If no connection is available, click Add Connection on the right to create one on the Workspace Manage > Data Source page.
Table	The name of the table from which data is read in the MySQL database. You can click Preview on the right to preview the selected table.
Start Offset	The start time of the sync node.
Time Zone	The time zone where the MySQL database resides.


Parameter	Description
Output Fields	The fields from which data is read.

6. Click  in the toolbar.

5.4.2 Oracle CDC reader

An Oracle Change Data Capture (CDC) reader synchronizes data through the trigger created in the Oracle database. You must activate the Oracle CDC service.

Create an Oracle CDC reader

1. Log on to the [DataWorks console](#). In the left-side navigation pane, click Workspaces. On the Workspaces page, find the target workspace and click **Data Analytics** in the Actions column.
2. On the Data Analytics tab, move the pointer over the  icon and choose **Data Integration > Real-Time Sync**.

You can also find the target workflow, right-click **Data Integration**, and choose **Create > Real-Time Sync**.

3. In the **Create Node** dialog box that appears, set **Node Name** and **Location**, and then click **Commit**.
4. On the configuration tab of the real-time sync node, drag **Oracle CDC** under **Reader** to the editing panel.
5. Click the **Oracle CDC** node and set parameters in the **Node Settings** section.

Parameter	Description
Connection	The connection to the Oracle CDC data store. In this example, you can only select an Oracle CDC connection. If no connection is available, click Add Connection on the right to create one on the Workspace Manage > Data Source page.
Table	The name of the table from which data is read in the Oracle database. You can click Preview on the right to preview the selected table.
Start Offset	The start time of the sync node.

Parameter	Description
Time Zone	The time zone where the Oracle database resides.
Output Fields	The fields from which data is read.

6. Click  in the toolbar.


5.4.3 Datahub reader

A Datahub 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. A Datahub reader has the following two features:

- Reads data in real time.
- Reads data concurrently based on the number of shards in Datahub.


Create a Datahub reader

1. Log on to the [DataWorks console](#). In the left-side navigation pane, click Workspaces. On the Workspaces page, find the target workspace and click **Data Analytics** in the Actions column.
2. On the Data Analytics tab, move the pointer over the  icon and choose **Data Integration > Real-Time Sync**.

You can also find the target workflow, right-click **Data Integration**, and choose **Create > Real-Time Sync**.
3. In the **Create Node** dialog box that appears, set **Node Name** and **Location**, and then click **Commit**.
4. On the configuration tab of the real-time sync node, drag **DataHub** under **Reader** to the editing panel.

5. Click the **Datahub reader** node and set parameters in the **Node Settings** section.

Parameter	Description
Connection	The connection to Datahub. In this example, you can only select a Datahub connection. If no connection is available, click Add Connection on the right to create one on the Workspace Manage > Data Source page.
Topic	The name of the topic from which data is read in Datahub. You can click Preview on the right to preview the selected topic.
Start Offset	The start time of the sync node.
Time Zone	The time zone where Datahub resides.
Output Fields	The fields from which data is read.

6. Click  in the toolbar.

5.4.4 LogHub reader


A LogHub reader reads data from LogHub topics you specified in real time and supports shard merge and split.



Note:

After shards are merged or split, duplicate data records may exist but no data will be lost.

Create a LogHub reader

- Log on to the [DataWorks console](#). In the left-side navigation pane, click Workspaces. On the Workspaces page, find the target workspace and click **Data Analytics** in the Actions column.
- On the Data Analytics tab, move the pointer over the  icon and choose **Data Integration > Real-Time Sync**.

You can also find the target workflow, right-click **Data Integration**, and choose **Create > Real-Time Sync**.
- In the **Create Node** dialog box that appears, set **Node Name** and **Location**, and then click **Commit**.

4. On the configuration tab of the real-time sync node, drag **LogHub** under **Reader** to the editing panel.
5. Click the **LogHub** node and set parameters in the **Node Settings** section.


Parameter	Description
Connection	The connection to LogHub. In this example, you can only select a LogHub connection. If no connection is available, click Add Connection on the right to create one on the Workspace Manage > Data Source page.
Logstore	The name of the Logstore from which data is read in LogHub. You can click Preview on the right to preview the selected Logstore.
Start Offset	The start time of the sync node.
Time Zone	The time zone where LogHub resides.
Advanced Settings	Specifies whether to split data in the Logstore.
Output Fields	The fields from which data is read.

6. Click  in the toolbar.

5.4.5 Kafka reader

A Kafka reader reads data from Kafka in real time by using the Kafka SDK.

Create a Kafka reader

1. Log on to the [DataWorks console](#). In the left-side navigation pane, click Workspaces. On the Workspaces page, find the target workspace and click **Data Analytics** in the Actions column.
2. On the Data Analytics tab, move the pointer over the  icon and choose **Data Integration > Real-Time Sync**.

You can also find the target workflow, right-click **Data Integration**, and choose **Create > Real-Time Sync**.

3. In the **Create Node** dialog box that appears, set **Node Name** and **Location**, and then click **Commit**.
4. On the configuration tab of the real-time sync node, drag **Kafka** under **Reader** to the editing panel.
5. Click the **Kafka reader** node and set parameters in the **Node Settings** section.


Parameter	Description
server	The broker server address of Kafka in the format of ip:port.
topic	The name of the topic from which data is read in Kafka. Kafka maintains feeds of messages in categories called topics. Each message published to the Kafka cluster is assigned to a topic. Each topic contains a group of messages.
keyType	The type of the Kafka key.
valueType	The type of the Kafka value.
Startup Mode	The start time of data synchronization.
Configuration 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.
Start Offset	The start time of the sync node.
Time Zone	The time zone where the Kafka cluster resides.
Output Fields	The output fields, which can be customized.

6. Click  in the toolbar.

5.4.6 ApsaraDB for POLARDB reader

Currently, an ApsaraDB for POLARDB reader can only read data from ApsaraDB POLARDB for MySQL databases instead of ApsaraDB POLARDB for PostgreSQL databases.

Procedure

1. Go to the **DataStudio** page.
 - a) Log on to the [DataWorks console](#).
 - b) In the left-side navigation pane, click **Workspaces**.
 - c) On the Workspaces page that appears, select the region where the target workspace resides, find the target workspace, and then click **Data Analytics** in the Actions column. The DataStudio page appears.
2. On the Data Analytics tab, move the pointer over the  icon and choose **Data Integration > Real-Time Sync**.

You can also find the target workflow, right-click **Data Integration**, and then choose **Create > Real-Time Sync**.

3. In the **Create Node** dialog box that appears, 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 case-insensitive.

4. Click **Commit**.
5. On the configuration tab of the real-time sync node, drag **PolarDB** under **Reader** to the editing panel.
6. Click **PolarDB1**. In the pane that appears on the right, set the required parameters in the **Node Settings** section.

Parameter	Description
Connection	The connection to the ApsaraDB POLARDB for MySQL data store. In this example, you can only select an ApsaraDB POLARDB for MySQL connection. If no connection is available, click New data source on the right to add one on the Workspace Manage > Data Source page.
Table	The name of the table from which data is read in the ApsaraDB POLARDB for MySQL data store. You can click Preview on the right to preview the selected table.
Output Fields	The fields from which data is read.

7. Click the  icon in the toolbar.

5.5 Writer


5.5.1 MaxCompute writer

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

Prerequisites

The corresponding reader or transformation node is configured. For more information, see [Supported data stores](#).

Procedure

1. Go to the **DataStudio** page.
 - a) Log on to the [DataWorks console](#).
 - b) In the left-side navigation pane, click **Workspaces**.
 - c) On the Workspaces page that appears, select the region where the target workspace resides, find the target workspace, and then click **Data Analytics** in the Actions column. The DataStudio page appears.
2. On the Data Analytics tab, move the pointer over the  icon and choose **Data Integration > Real-Time Sync**.

You can also find the target workflow, right-click **Data Integration**, and then choose **Create > Real-Time Sync**.
3. In the **Create Node** dialog box that appears, 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 case-insensitive.


4. Click **Commit**.
5. On the configuration tab of the real-time sync node, drag **MaxCompute** under **Writer** to the editing panel. Connect it to the desired reader or transformation node in the editing panel.

6. Click **MaxCompute1**. In the pane that appears on the right, set the required parameters in the **Node Settings** section.

Parameter	Description
Connection	<p>The connection to the MaxCompute data store. In this example, you can only select a MaxCompute connection.</p> <p>If no connection is available, click New data source on the right to add one on the Workspace Manage > Data Source page. For more information, see #unique_70.</p>
Table	<p>The name of the MaxCompute table to which data is written.</p> <p>You can click One-Click table creation on the right to create a table, or click Preview to preview the selected table.</p> <div>  Notice: Before you create a table, connect the writer node to a reader node and make sure that the Output Fields parameter is specified for the reader node. </div>
Partition message	The information about the MaxCompute partitioned table.
Mappings	The field mappings between the source and destination data stores. Click Mappings and set field mappings. The sync 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 Table** dialog box that appears, set the parameters as required.

Parameter	Description
Table name	The name of the MaxCompute table.
Life cycle	The lifecycle of the MaxCompute table. For more information, see #unique_147 .
Data field structure	The schema of the MaxCompute table. To add a field, click Add .

Parameter	Description
Partition settings	<p>The partition information about the MaxCompute table. For more information, see #unique_148.</p> <div>  Notice: You need to configure at least two levels of partitions, that is, yearly and monthly partitions. You can configure up to five levels of partitions, that is, yearly, monthly, daily, hourly, and minutely partitions. </div>

7. Click the  icon in the toolbar.


5.5.2 Hologres writer

You can build a real-time data warehouse with the real-time writing capability of Hologres.

Prerequisites

The corresponding reader or transformation node is configured. For more information, see [Supported data stores](#).

Procedure

- Go to the **DataStudio** page.
 - Log on to the [DataWorks console](#).
 - In the left-side navigation pane, click **Workspaces**.
 - On the Workspaces page that appears, select the region where the target workspace resides, find the target workspace, and then click **Data Analytics** in the Actions column. The DataStudio page appears.
- On the Data Analytics tab, move the pointer over the  icon and choose **Data Integration > Real-Time Sync**.

You can also find the target workflow, right-click **Data Integration**, and then choose **Create > Real-Time Sync**.

- In the **Create Node** dialog box that appears, 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 case-insensitive.

- Click **Commit**.

5. On the configuration tab of the real-time sync node, drag **Hologres** under **Writer** to the editing panel. Connect it to the desired reader or transformation node in the editing panel.
6. Click **Hologres1**. In the pane that appears on the right, set the required parameters in the **Node Settings** section.

Parameter	Description
Connection	<p>The connection to the Hologres data store. In this example, you can only select a Hologres connection.</p> <p>If no connection is available, click New data source on the right to add one on the Workspace Manage > Data Source page.</p>
Table	<p>The name of the Hologres table to which data is written.</p> <p>You can click One-Click table creation on the right to create a table, or click Preview to preview the selected table.</p>
Job Type	<p>The type of the job. Valid values: Replay (replay log to restore data) and Insert (archived save).</p> <ul style="list-style-type: none">• Replay (replay log to restore data): indicates that the Hologres writer performs the same operation on the Hologres data store as that performed on the source data store. For example, if the INSERT statement is executed to add a record to the source data store, the Hologres writer executes the INSERT statement to add the same record to the Hologres data store. If the UPDATE or DELETE statement is executed in the source data store, the Hologres writer executes the UPDATE or DELETE statement in the Hologres data store accordingly.• Insert (archived save): indicates that the Hologres writer uses the Hologres data store as streaming data storage. Data is synchronized from the source data store to the Hologres data store by using the INSERT statement.

Parameter	Description
Conflict Mode	<p>The solution to data write conflicts. Valid values: Overwrite and Ignore</p> <ul style="list-style-type: none"> • Overwrite: indicates that the Hologres writer uses the new data synchronized from the source data store to overwrite the existing data in the Hologres data store. • Ignore: indicates that the Hologres writer ignores the new data synchronized from the source data store and retains the existing data in the Hologres data store.
Mappings	<p>The field mappings between the source and destination data stores. Click Mappings and set field mappings. The sync node synchronizes data based on the field mappings.</p>

7. Click the  icon in the toolbar.


5.5.3 Datahub writer

Datahub is a platform designed to process streaming data. You can publish and subscribe to applications for streaming data in Datahub and distribute the data to other platforms. Datahub allows you to analyze streaming data and build applications based on the streaming data.

The 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>
```

Create a Datahub writer

1. Log on to the [DataWorks console](#). In the left-side navigation pane, click Workspaces. On the Workspaces page, find the target workspace and click **Data Analytics** in the Actions column.
2. On the Data Analytics tab, move the pointer over the  icon and choose **Data Integration > Real-Time Sync**.

You can also find the target workflow, right-click **Data Integration**, and choose **Create > Real-Time Sync**.
3. In the **Create Node** dialog box that appears, set **Node Name** and **Location**, and then click **Commit**.

- On the configuration tab of the created real-time sync node, drag **DataHub** under **Writer** to the editing panel. Connect it to the desired reader or transformation node in the panel.
- Click the **Datahub writer** node and set parameters in the **Node Settings** section.


Parameter	Description
Connection	The connection to Datahub. In this example, you can only select a Datahub connection. If no connection is available, click Add Connection on the right to create one on the Workspace Manage > Data Source page.
Topic	The name of the topic to which data is written in Datahub. You can click Preview on the right to preview the selected topic.
Records per Batch	The number of records that are written at a time.
Mappings	The mappings between fields in the source and destination data stores. DataWorks synchronizes data based on the field mappings.

- Click  in the toolbar.

5.5.4 Kafka writer

When you configure a Kafka writer, you only need to select a table and configure the field mappings.


Create a Kafka writer

- Log on to the [DataWorks console](#). In the left-side navigation pane, click Workspaces. On the Workspaces page, find the target workspace and click **Data Analytics** in the Actions column.
- On the Data Analytics tab, move the pointer over the  icon and choose **Data Integration > Real-Time Sync**.

You can also find the target workflow, right-click **Data Integration**, and choose **Create > Real-Time Sync**.

3. In the **Create Node** dialog box that appears, set **Node Name** and **Location**, and then click **Commit**.
4. On the configuration tab of the created real-time sync node, drag **Kafka** under **Writer** to the editing panel. Connect it to the desired reader or transformation node in the panel.
5. Click the **Kafka writer** node and set parameters in the **Node Settings** section.

Parameter	Description
server	The broker server address of Kafka in the format of ip:port.
topic	The name of the topic to which data is written in Kafka. Kafka maintains feeds of messages in categories called topics. Each message published to the Kafka cluster is assigned to a topic. Each topic contains a group of messages.
keyColumn	The column that is specified as the key.
valueColumn	The column that is specified as the value. If this parameter is not specified, all columns are concatenated by using the delimiter specified by fieldDelimiter to form the value.
keyType	The type of the Kafka key.
valueType	The type of the Kafka value.
batchSize	The number of data records that are written at a time. Default value: 1024.
Configuration 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.

6. Click  in the toolbar.

5.6 Transformation


5.6.1 Data filtering

The data filtering plug-in can filter data based on specified rules, such as the field size.

Only data that meets the rules is retained.

Create a data filtering node

1. Log on to the [DataWorks console](#). In the left-side navigation pane, click Workspaces. On the Workspaces page, find the target workspace and click **Data Analytics** in the Actions column.

2. On the Data Analytics tab, move the pointer over the  icon and choose **Data**

Integration > Real-Time Sync.

You can also find the target workflow, right-click **Data Integration**, and choose **Create > Real-Time Sync.**

3. In the **Create Node** dialog box that appears, set **Node Name** and **Location**, and then click **Commit**.

4. On the configuration tab of the created real-time sync node, drag **Data Filtering** under **Transform** to the editing panel. Connect it to the desired reader in the panel.

5. Click the **data filtering** node and set parameters in the **Node Settings** section.

- **Node Settings**

Rule: You can configure rules to filter data in data stores. Only data that meets the rules is retained.

- **Output Fields**

The names and types of output fields after filtering.

6. Click  in the toolbar.

5.6.2 Groovy


Groovy is a custom transformation plug-in that allows you to write Groovy code to transform the fields in each record to meet specific requirements.

For example, you can use Groovy to transform uppercase letters in values of a specific field to lowercase letters, add a prefix or suffix to all field values, or add attributes parsed from JavaScript Object Notation (JSON) data to other fields.

Implementation

The Groovy plug-in compiles and runs the custom Groovy code through the Groovy compiler. The code needs to inherit the `com.alibaba.di.plugin.center.transformer.Transformer` class and implement the `Record evaluate(Record record)` method.

Create a Groovy node

1. Log on to the [DataWorks console](#). In the left-side navigation pane, click Workspaces. On the Workspaces page, find the target workspace and click **Data Analytics** in the Actions column.
2. On the Data Analytics tab, move the pointer over the  icon and choose **Data Integration > Real-Time Sync**.

You can also find the target workflow, right-click **Data Integration**, and choose **Create > Real-Time Sync**.
3. In the **Create Node** dialog box that appears, set **Node Name** and **Location**, and then click **Commit**.
4. On the configuration tab of the created real-time sync node, drag **Groovy** under **Transform** to the editing panel. Connect it to the desired reader in the panel.
5. Click the **Groovy** node and set parameters in the **Node Settings** section.

- **Node Settings**

Groovy Code: the Groovy code to run. For example, to read data from a MySQL database, add the GROOVY suffix to the first field in the data, and write the transformed data to Datahub, enter the following code:

```
import com.alibaba.di.plugin.center.element.Column;
import com.alibaba.di.plugin.center.element.StringColumn;
import com.alibaba.di.plugin.center.record.Record;
import com.alibaba.di.plugin.center.transformer.Transformer;

class MyTransformer extends Transformer {
    @Override
    public Record evaluate(Record record) {
        Column column = record.getColumn(0);
        String newValue = column.asString() + "GROOVY";
        record.setColumn(0, new StringColumn(newValue));
        return record;
    }
}
```


```
}
```

- **Output Fields**

The output fields of the parent node that are used as the input fields of the current node.

- **Custom Field**


The output fields of the Groovy node after transformation.

6. Click  in the toolbar.

5.6.3 String replacement

String replacement is a transformation plug-in used to replace field values of the STRING type.

1. Log on to the [DataWorks console](#). In the left-side navigation pane, click Workspaces. On the Workspaces page, find the target workspace and click **Data Analytics** in the Actions column.

2. On the Data Analytics tab, move the pointer over the  icon and choose **Data**

Integration > Real-Time Sync.

You can also find the target workflow, right-click **Data Integration**, and choose **Create > Real-Time Sync.**

3. In the **Create Node** dialog box that appears, set **Node Name** and **Location**, and then click **Commit**.

4. On the configuration tab of the created real-time sync node, drag **String Replace** under **Transform** to the editing panel. Connect it to the desired reader in the panel.

5. Click the **string replacement** node and set parameters in the **Node Settings** section.

- **Node Settings**


Rule:

- **Field:** the field of the parent node to be used as the input field.
- **Regular Expression Match:** specifies whether a regular expression is used to search for the original string.
- **Original String:** the original string to search.
- **New String:** the new string to replace the original string.
- **Case Sensitive:** specifies whether the value is case sensitive during the search.

Add Condition: Click the button to add more string replacement rules.

- **Output Fields**

The output fields after string replacement.

6. Click  in the toolbar.

6 Node optimization

6.1 Optimize synchronization performance

This topic describes the factors that affect the speed of data synchronization, and how to adjust the concurrency of sync nodes to maximize the synchronization speed. This topic also describes bandwidth throttling settings, scenarios of slow data synchronization, and how to deal with slow data synchronization.

Data Integration is a one-stop platform that supports real-time and batch data synchronization between data stores in any location and in any network environment. You can synchronize 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 heterogeneous data stores. 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, solid-state drive (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 higher 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 the 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 small bandwidth limit.
 - Whether to create an index for query statements.
- Destination
 - Performance: the performance of the CPU, memory, 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 sections describe the optimal settings of a sync node.

Concurrency

You can configure the concurrency for a node on the codeless user interface (UI). The following example shows 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 transmission rate 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 transmission rate to 30 Mbit/s. The following example shows how to configure an upper limit for synchronization speed in the code editor, in which the transmission bandwidth is 1 Mbit/s:

```
"setting": {
```



```
"speed": {  
  "throttle": true // Specifies that bandwidth throttling is enabled.  
  "mbps": 1, // The synchronization speed.  
}
```

**Note:**

- When the **throttle** parameter is set to false, bandwidth 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 transmission rate of a node to increase the synchronization speed. However, the maximum transmission rate is limited by the number of files.

Assume that the maximum transmission rate can be set to n Mbit/s for n files.

- If you set the maximum transmission rate to $(n+1)$ Mbit/s, the files are still synchronized at a speed of n Mbit/s.
- If you set the maximum transmission rate to $(n-1)$ Mbit/s, the files are synchronized at a speed of $(n-1)$ Mbit/s.
- A table in a relational database can be split based on the maximum transmission rate only after you set the maximum transmission rate and shard key. Usually, relational databases support only numeric-type shard keys. However, Oracle databases support numeric- and string-type 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 ApsaraDB for Relational Database Service (RDS) to MaxCompute. The node has waited about 800 seconds before it is run. 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 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%
```

- Handling method

The default resource group is not exclusively used by a single user. Many nodes, not just two or three nodes of a single user, run on the default resource group. If resources are insufficient after you start to run a node, the node needs to wait for resources. In this case, the node is delayed for 800 seconds, and it only takes 18 seconds for the node to be run.

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. You can avoid this time period to prevent your nodes from waiting for resources.

- Scenario 2: Accelerate nodes that synchronize data from multiple source tables to the same destination table.

- Example

Multiple sync nodes are configured to run in sequence to synchronize data from tables of multiple data stores to the same destination table. However, the synchronization takes a long time.

- Handling method

To start multiple concurrent nodes that write data to the same destination database, pay attention to the following points:

- Make sure that the destination database can support the running 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: A full table scan slows down the data synchronization because no index is added in the WHERE clause.

- 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';
```

The sync node started to run at **11:01:24.875 on October 25, 2016** and started to return results from **11:11:05.489 on October 25, 2016**. The synchronization program is waiting for the database to return SQL query results. However, it takes a long time before MaxCompute can respond.

- Cause

When the WHERE clause is used for a query, the createTime column is not indexed, resulting in a full table scan.

- Handling method

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.

6.2 Optimize a sync node

When a sync node is scheduled, the instance may take longer than expected to run. This topic describes how to optimize a sync node when the instance runs slowly or the node start time greatly differs from the scheduling time.

Prerequisites

The operational logs and attribute information of a sync node are obtained before you optimize the node.

DataWorks provides level-1 scheduling resources and level-2 running resources for sync nodes.

- Level-1 scheduling resources: Go to **Operation Center** and choose **Cycle Task Maintenance > Cycle Task** in the left-side navigation pane. On the page that appears, right-click the sync node in the directed acyclic graph (DAG) on the right and select **View Node Details**. On the **Node Information** page that appears, you can view the attribute information and level-1 scheduling resources of the node.

- Level-2 running resources: Go to the **Data Integration** module and click **Custom Resource Group** in the left-side navigation pane. On the page that appears, you can view and add level-2 running resources.

Context

Generally, a sync node may be considered slow in the following scenarios:

- The node start time greatly differs from the scheduling time.
- The sync node remains in the WAIT state for a long time.
- The sync node runs at a low speed.

Scenario 1 where the start time of a sync node greatly differs from the scheduling time

In this scenario, you must first obtain the operational logs and attribute information of the node and compare the operational logs with the attribute information. The comparison result shows that the node start time in the operational logs differs from the scheduling time in the attribute information. Most of the time is consumed while waiting for scheduling .

Problem example

1. Go to **Operation Center** and choose **Cycle Task Maintenance > Cycle Task** in the left-side navigation pane. On the page that appears, right-click the sync node in the DAG on the right and select **View Node Details**. On the **Node Information** page that appears, check the scheduling time of the node, which is 00:00. However, the node actually starts at 00:29. It is inferred that most of the time is consumed while waiting for scheduling.
2. Choose **Cycle Task Maintenance > Cycle Instance** in the left-side navigation pane. On the page that appears, right-click the node instance and select **View Runtime Log**. The operational logs show that the node starts at 00:29 and ends at 00:30. That is, it takes only 1 minute to complete the task. This indicates that the node is able to run normally.

Troubleshooting

1. Check whether multiple nodes are scheduled at the same time in your workspace. The default resource group contains a limited number of level-1 scheduling resources. If multiple nodes are scheduled at the same time, other nodes must queue up and wait for scheduling.
2. The peak hours for business scheduling range from 00:00 to 02:00. We recommend that you run your business during off-peak hours.

Scenario 2 where a sync node runs at a steady speed of 0

The operational logs of the node show that the node runs at a steady speed of 0. This is usually because the source database has a high CPU load or network traffic usage, and therefore the SQL statement is executed slowly. Or, **truncate** operations are performed before the SQL statement is executed, and therefore the SQL statement is processed for a long time.

Problem example

1. The operational logs of the node show that the node keeps running from 18:00 to 21:13 at a steady speed of 0.
2. The operational logs also record a truncate operation that lasts from 18:00 to 21:13.

Troubleshooting

It can be inferred that the sync node is slow due to the truncate operation. You need to check the cause of the slow truncate operation in the source database.

Scenario 3 where a sync node runs at a low speed

The operational logs of the node show that the node runs at a low speed slightly greater than 0.

Problem example

1. The operational logs of the node show that the node runs at a low speed of about 1.93 Kbit/s.
2. The value of the **WaitReaderTime** field is greater than that of the **WaitWriterTime** field, indicating that more time is consumed for waiting to read data.

Troubleshooting

If the node runs at a low speed, check whether the value of WaitReaderTime or WaitWriterTime is large. If reading or writing consumes more time, check the load of the corresponding source or destination database.

7 Resource groups

7.1 Overview

This topic describes the basic concepts and types of resource groups, and how the connectivity and performance of resource groups affect data synchronization. This topic also compares different types of resource groups. You can select appropriate resource groups based on your needs.

Basic concepts

A resource group is a collection of computing resources where batch sync nodes of Data Integration run. Generally, a resource group refers to a server that consists of CPU, memory, and network resources.

In the process of running a sync node, the resource group pulls data from the source data store and pushes the data to the destination data store.

Connectivity and performance

When you use resource groups, you must pay attention to their connectivity and performance.

- Connectivity

To make sure that data can be properly synchronized, a resource group must be properly connected to the source and destination data stores. Connectivity is a most important factor that affects data synchronization.

Data Integration cannot build networks. You must make sure that Data Integration is properly connected to data stores before you use it to synchronize data. If Data Integration is disconnected from data stores, batch sync nodes cannot be run.

- Performance

Batch sync 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 start, wait for resources for a prolonged period after startup, transmit data at a low rate, or fail to generate results in a timely manner.

To guarantee smooth running of batch sync nodes, you must allocate adequate resources for them. We recommend that you use exclusive resource groups to run

batch sync nodes so that the nodes do not need to compete for resources in the public resource pool.

Types and comparison of resource groups

Data Integration supports the following types of resource groups:

- [Use the default resource group](#)
- [Use exclusive resource groups for data integration](#)
- [Custom resource group](#)

The three types of resource groups are applicable to different scenarios. You can select a resource group as needed to run a sync node.

Type	Default resource group	Exclusive resource group	Custom resource group
Ownership of resources	The resources are maintained by DataWorks and shared among all tenants.	The resources are maintained by DataWorks and exclusively used by the tenant that purchases the exclusive resource group.	The resources are maintained by yourself and are located in your IDC.
Network	Supports non-Alibaba Cloud databases deployed on classic networks or the Internet and Alibaba Cloud databases on any type of networks	Supports non-Alibaba Cloud databases deployed on Virtual Private Clouds (VPCs) or the Internet and Alibaba Cloud databases on any type of networks	Supports non-Alibaba Cloud databases deployed on VPCs or the Internet and Alibaba Cloud databases on any type of networks
Billing method	Tiered pricing based on the number of node instances	Subscription based on the server specifications	Monthly billing in the pay-as-you-go mode based on the DataWorks edition
Supported data stores	Some data stores	All data stores	All data stores
Security	High	High	Depending on the environment where your server resides

Type	Default resource group	Exclusive resource group	Custom resource group
<p>Node running efficiency</p> <p>Node running efficiency refers to whether nodes can be allocated with sufficient computing resources to achieve the highest performance .</p>	Low	High	Depending on the environment where your server resides
<p>Reliability</p> <p>Reliability refers to whether nodes can be started as scheduled and generate results in a timely manner in the case that network resources are occupied by other tenants.</p>	Low	High	Depending on the environment where your server resides

Type	Default resource group	Exclusive resource group	Custom resource group
Scenario	Suitable for running a small number of non-important, non-urgent, or testing nodes	Suitable for running a large number of important production nodes	<ul style="list-style-type: none"> Suitable if you want to make full use of the computing resources you have purchased Suitable if both the source and destination data stores are in the same IDC as the custom resource group
Recommendation index	★★	★★★★★	★

Based on the preceding table, we recommend that you use exclusive resource groups to run batch sync nodes.

7.2 Use the default resource group

Data Integration of DataWorks provides a default resource group for you to create and run nodes.

The default resource group is created and maintained by Data Integration.

- The default resource group is a public resource pool. Nodes that use this resource group may not be run as scheduled due to insufficient resources. If you want your nodes to be run as expected, use an exclusive resource group. For more information, see [Use exclusive resource groups for data integration](#).
- You need to provide information about the default resource group when you configure network connectivity. For more information, see [Configure a security group](#) and [Configure a whitelist](#).
- By default, the default resource group is used to test the connectivity to data stores. If your nodes are run on another resource group, click **Advanced Mode** and select the corresponding resource group during the connectivity test.

7.3 Use exclusive resource groups for data integration

When data integration nodes run in high concurrency and load shifting is not an option, enterprises need exclusive computing resources to guarantee that data is transmitted quickly and reliably. To address such problems, DataWorks provides exclusive resource groups for data integration.

Context

An exclusive resource group for data integration can only access the data stores in the same region and zone as the resource group.

- You must bind a purchased exclusive resource group for data integration to a zone of your Virtual Private Cloud (VPC). The exclusive resource group for data integration must be in the same zone as the data stores that you want to access.
- When you bind an exclusive resource group for data integration or an exclusive resource group for scheduling to a VPC, select the VSwitch associated with the data stores that you want to access.
- After an exclusive resource group for data integration is bound to a VPC, the exclusive resource group for data integration can only access the data stores in the specified zone of the VPC. We recommend that you make sure that the same zone as that the VPC to be bound is selected when you add an exclusive resource group for data integration.
- An exclusive resource group for data integration cannot access data stores deployed on a classic network of Alibaba Cloud. If your data stores are on a classic network, we recommend that you run sync nodes on the default resource group.

Purchase an exclusive resource group for data integration

1. Log on to the [DataWorks console](#).
2. In the left-side navigation pane, click **Resource Groups**. The **Exclusive Resource Groups** tab appears by default.
3. Click **Add Exclusive Resource Group**.
4. In the **Add Exclusive Resource Group** dialog box that appears, click **Purchase** next to **Order Number**.
5. On the purchase page, set **Region**, **Type**, **Exclusive Resource Groups for Data Integration**, **Units**, and **Duration**, and click **Buy Now**.



Note:


Here, **Type** must be set to **Exclusive Resource Groups for Data Integration**.

Exclusive resources cannot be used across regions. For example, the exclusive resources in the China (Shanghai) region can only be used by workspaces in the China (Shanghai) region.

6. After confirming that the order information is correct, select the check box for **DataWorks Exclusive Resources Agreement of Service** and click **Pay**.

Add an exclusive resource group for data integration

1. On the **Exclusive Resource Groups** tab, click **Add Exclusive Resource Group**.
2. In the **Add Exclusive Resource Group** dialog box that appears, set the parameters.

Parameter	Description
Resource Group Type	The type of the exclusive resource group. The valid values are Exclusive Resource Groups and Exclusive scheduling resources . The two types of resource groups are applicable to general node scheduling and data synchronization, respectively.
Resource Group Name	The name of the exclusive resource group, which must be unique within all resource groups of a tenant.  Note: A tenant account indicates an Alibaba Cloud account. Multiple Resource Access Management (RAM) users may exist under a tenant account.
Resource Group Description	The description of the exclusive resource group.
Order Number	The order number of the exclusive resource group. If you have not purchased any exclusive resource groups, click Purchase next to Order Number to go to the purchase page and purchase an exclusive resource group.
Zone	The zone of servers available in the region. Select a zone based on your requirements.

3. After the configuration is completed, click **Create**.



Note:

The exclusive resource group is initialized within 20 minutes. Wait until its status changes to **Running**.

Bind an exclusive resource group for data integration to a VPC

Exclusive resource groups are deployed in a VPC managed by DataWorks. To allow exclusive resource groups to access data stores in your own VPC, you must bind the exclusive resource groups to your VPC.

1. Click **Add VPC Binding** in the Actions column of the exclusive resource group for data integration.

**Note:**

Before binding the exclusive resource group for data integration to your VPC, you must authorize DataWorks to access your cloud resources in the RAM console.

2. Click **Add Binding** in the upper-right corner. In the **Add VPC Binding** dialog box that appears, set the parameters.

- If no VPC is available, click **Create VPC** to create a VPC on the **VPCs** page of the VPC console.

Click **Create VPC**. In the **Create VPC** dialog box that appears, set the parameters and click **OK**.

After a VPC is created, you can view it on the VPCs page.

- If no VSwitch is available, click **Create VSwitch** to create a VSwitch on the **VSwitches** page of the VPC console.

Click **Create VSwitch**. In the **Create VSwitch** dialog box that appears, set the parameters and click **OK**.

After a VSwitch is created, you can view it on the VSwitches page.

- If no security group is available, click **Create Security Group** to create a security group on the **Security Groups** page of the ECS console.

Click **Create Security Group**. In the **Create Security Group** dialog box that appears, set the parameters and click **OK**.

After a security group is created, you can view it on the Security Groups page.

3. After the configuration is completed, click **Create**.

Purchase an ApsaraDB for RDS instance

1. Move the pointer over the icon in the upper-left corner of the DataWorks console and click **Relational Database Service** to log on to the ApsaraDB for RDS console. The **Instances** page appears by default.
2. Click **Create Instance** in the upper-right corner.
3. On the purchase page, set the parameters in **Basic Configurations** step and click **Next:Instance Configuration**.
4. Set the parameters in the **Instance Configuration** step and click **Next:Confirm Order**.



Note:

The values of **Version**, **Zone**, and **Network Type** must be the same as those specified for the exclusive resource group for data integration.

5. Confirm the settings in the **Parameters** section, specify **Purchase Plan** and **Duration**, and read and select the check box for **Terms of Service**. Note that you only need to specify Duration if you create a subscription instance.
6. Click **Pay Now**.

Configure a whitelist for an ApsaraDB for RDS instance

1. On the **Instances** page, click the ID of the created ApsaraDB for RDS instance.
2. On the page that appears, click **Data Security** in the left-side navigation pane.
3. On the **Whitelist Settings** tab, click **Create Whitelist**.
4. In the **Create Whitelist** dialog box that appears, enter the IP addresses or Classless Inter-Domain Routing (CIDR) blocks of the region and the IP address of the created VPC.
5. In the left-side navigation pane, click **Accounts** and **Databases** to create an account and a database. For more information, see [#unique_158](#).


Create a MySQL connection

1. Move the pointer over the icon in the upper-left corner of the ApsaraDB for RDS console and click **DataWorks** to log on to the DataWorks console.
2. Find the target workspace and click **Data Integration** in the Actions column.
3. In the left-side navigation pane, click **Connection** to go to the **Workspace Manage > Data Source** page.
4. On the **Data Source** page, click **Add Connection** in the upper-right corner.

5. In the **Add Connection** dialog box that appears, click **MySQL** in the Relational Database section.
6. In the **Add MySQL Connection** dialog box that appears, set the parameters.

The MySQL connection type can be set to **ApsaraDB for RDS** or **JDBC Connection Mode**.

In this example, set **Connect To** to **ApsaraDB for RDS**.

Parameter	Description
Connect To	The type of the connection. Here, set the value to ApsaraDB for RDS .
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_) and must start with a letter.
Description	The description of the connection. The description cannot exceed 80 characters in length.
Applicable Environment	<div>The environment in which the connection is used. Valid values: Development and Production.</div> <div> Note: This parameter is available only when the workspace is in standard mode.</div>
Region	The region of the ApsaraDB for RDS instance.
RDS Instance ID	The ID of the ApsaraDB for RDS instance. You can view the ID in the ApsaraDB for RDS console.
RDS Instance Account ID	The ID of the Alibaba Cloud account used to purchase the ApsaraDB for RDS instance. You can view your account ID on the Security Settings page after logging on to the Alibaba Cloud console with your Alibaba Cloud account.
Database Name	The name of the database.
Username	The username for logging on to the database.
Password	The password for logging on to the database.

**Note:**

You must add the IP addresses or CIDR blocks that you use to access the ApsaraDB for RDS instance to a whitelist of the instance. For more information, see [Configure a whitelist](#).

7. Click **Test Connection**.
8. After the connectivity test is passed, click **Complete**.

Change the workspace to which an exclusive resource group is bound

You must bind an exclusive resource group to a workspace so that the exclusive resource group can be used by nodes in the workspace. You can bind an exclusive resource group to multiple workspaces.

1. Log on to the DataWorks console. In the left-side navigation pane, click **Resource Groups**.
2. Click **Change Workspace** in the Actions column of the exclusive resource group for data integration.
3. In the **Change Workspace** dialog box that appears, select the desired workspace and click **OK**.

After you bind the exclusive resource group for data integration to a workspace, you can use the exclusive resource group for data integration to run sync nodes in the workspace.

7.4 Add a custom resource group

DataWorks provides the default resource group for you to migrate large amounts of data to the cloud for free. However, the default resource group does not work if a high transmission speed is required or your data stores are deployed in complex environments. You can use custom resource groups to run your sync nodes. This guarantees connections to your data stores and enables a higher transmission speed.

A workspace administrator can add or modify custom resource groups on the **Custom Resource Group** page of Data Integration.

If the default resource group cannot access your data stores deployed in complex network environments, you can add custom resource groups to enable data synchronization between any network environments.



Note:

- Custom resource groups added on the **Custom Resource Group** page of Data Integration can only run sync nodes in the current workspace. They do not appear in the

[resource group list](#). Currently, custom resource groups added on the Custom Resource Group page cannot run sync nodes in a manually triggered workflow.

- You can add only one custom resource group on an Elastic Compute Service (ECS) instance. You can select only one network type for each custom resource group.
- When you register an ECS instance for hosting a custom resource group, you can set the network type to classic network only when the ECS instance is in the China (Shanghai) region. In this case, you must enter the hostname of the ECS instance. We recommend that you set the network type to Virtual Private Cloud (VPC) preferentially. You can only set the network type to VPC for ECS instances in other regions. In this case, you must enter the universally unique identifier (UUID) of the ECS instance to be registered.
- 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 SQL files on the ECS instance when you write a shell script for a node.
- A resource group for scheduling is mainly used to schedule nodes. They have limited resources and are not suitable for computing nodes. Therefore, we recommend that you do not add custom resource groups on 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.

Limits

- The difference between the time of the ECS instance where a custom resource group 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.
- If the timeout error message `response code is not 200` exists in the log file of `alisatasknode`, the custom resource group was inaccessible within the specific time period. The ECS server hosting the custom resource group can continue working 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.

Purchase an ECS instance

Purchase an ECS instance



Note:

- CentOS V6, CentOS V7, or AliOS is recommended.

- If the added ECS instance needs to run MaxCompute nodes or sync nodes, verify that the current Python version of the ECS instance is V2.6 or V2.7. (The Python version of CentOS V5 is V2.4 while those of other operating systems are later than V2.6.)
- Make sure that the ECS instance can access the Internet. Ping `www.alibabacloud.com` on the ECS instance and check whether the URL can be pinged.
- We recommend that you configure the ECS instance with an 8-core CPU and 16 GB memory.

View the hostname and internal IP address of the ECS instance

Log on to the ECS console. In the left-side navigation pane, choose **Instances & Images > Instances**. On the page that appears, view the hostname and IP address of the purchased ECS instance.

Enable port 8000 for reading logs



Note:

You do not need to enable port 8000 if your ECS instance is in a VPC. Steps in this section apply to ECS instances on classic networks only.

1. Add a security group rule.

Log on to the **ECS console**. In the left-side navigation pane, choose **Network & Security > Security Groups**. On the page that appears, find the target security group and click **Add Rules** in the Actions column.

2. On the **Security Group Rules > Inbound** page that appears, click **Add Security Group Rule** in the upper-right corner.

3. In the **Add Security Group Rule** dialog box that appears, set the parameters. Set Authorization Object to the fixed IP address of Data Integration and Port Range to 8000/8000.

Add a custom resource group

1. In the left-side navigation pane, click **Custom Resource Group**. The **Custom Resource Group** page appears.
2. Click **Add Resource Group** in the upper-right corner.



Note:

By default, the Custom Resource Group page lists only your custom resource groups, but not the default resource group.

3. In the **Add Resource Group** dialog box that appears, set **Resource Group Name** and click **Next**.
4. In the **Add Server** dialog box that appears, set the parameters and click **Next**.

Parameter	Description
Network Type	The network type of the ECS instance. Currently, the classic network type is supported only for ECS instances in the China (Shanghai) region. You can only set this parameter to VPC for ECS instances in other regions.
Server Name or ECS UUID	<ul style="list-style-type: none"> The hostname of the ECS instance when Network Type is set to Classic Network. To obtain the hostname, log on to the ECS instance and run the hostname command. The UUID of the ECS instance when Network Type is set to VPC. To obtain the UUID, log on to the ECS instance and run the dmidecode grep UUID command.
Server IP Address	The internal 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 hosting 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 hosting a custom resource group.

**Note:**

To add an ECS instance that resides in a VPC, enter the UUID of the ECS instance. To obtain the UUID, log on to the ECS instance and run the dmidecode | grep UUID command.

For example, if the dmidecode | grep UUID command returns UUID: 713F4718-8446-4433-A8EC-6B5B62D7****, the UUID is 713F4718-8446-4433-A8EC-6B5B62D7****.

5. Install and initialize the Agent.

If an ECS instance is newly purchased and has never been used, follow these steps:

- a. Log on to the ECS instance through Secure Shell (SSH) as the root user.
- b. Run the following commands:

```
chown admin:admin /opt/taobao //Grant the admin user the permission to access the /opt/taobao directory.
```

```
wget https://alisaproxy.shuju.aliyun.com/install.sh --no-check-certificate
sh install.sh --user_name=****19d --password=****h1bm --enable_uid=false
```

- c. Wait for a while, click **Refresh** in the Add Server dialog box, and check whether the service status changes to **Available**.
- d. Enable port 8000 on the ECS instance.

**Note:**

If an error occurs when you run the `install.sh` script or you have 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 that need to be run during the installation and initialization process differ for each user. Run relevant commands according to instructions on the initialization interface.

If the service status remains **Stopped** after the preceding steps, the hostname may not be bound to an IP address, as shown in the following figure.

```
at org.springframework.beans.factory.support.DefaultSingletonBeanRegistry
y.getSingleton(DefaultSingletonBeanRegistry.java:222)
at org.springframework.beans.factory.support.AbstractBeanFactory.doGetBe
an(AbstractBeanFactory.java:298)
at org.springframework.beans.factory.support.AbstractBeanFactory.getBean
(AbstractBeanFactory.java:192)
at org.springframework.beans.factory.support.DefaultListableBeanFactory.
preInstantiateSingletons(DefaultListableBeanFactory.java:585)
at org.springframework.context.support.AbstractApplicationContext.finish
BeanFactoryInitialization(AbstractApplicationContext.java:895)
at org.springframework.context.support.AbstractApplicationContext.refres
h(AbstractApplicationContext.java:425)
at org.springframework.context.support.ClassPathXmlApplicationContext.<i
nit>(ClassPathXmlApplicationContext.java:139)
at org.springframework.context.support.ClassPathXmlApplicationContext.<i
nit>(ClassPathXmlApplicationContext.java:93)
at com.alibaba.alisa.node.server.StartUp.main(StartUp.java:24)
Caused by: java.util.MissingResourceException: Can't find resource for bundle ja
va.util.PropertyResourceBundle, key alisa.node.host.name
at java.util.ResourceBundle.getObject(ResourceBundle.java:458)
at java.util.ResourceBundle.getString(ResourceBundle.java:487)
at com.alibaba.alisa.common.util.PropertyUtils.getProperty(PropertyUtils
.java:32)
... 24 more
"alisatasknode.log" 3937L, 445471C 3937,2-9 Bot
```

To bind the hostname to an IP address, follow these steps:

1. Switch to 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 service status and check whether the ECS instance is registered.

**Note:**

- If the ECS instance is still in the Stopped status after you refresh the page, you can restart alisatasknode.

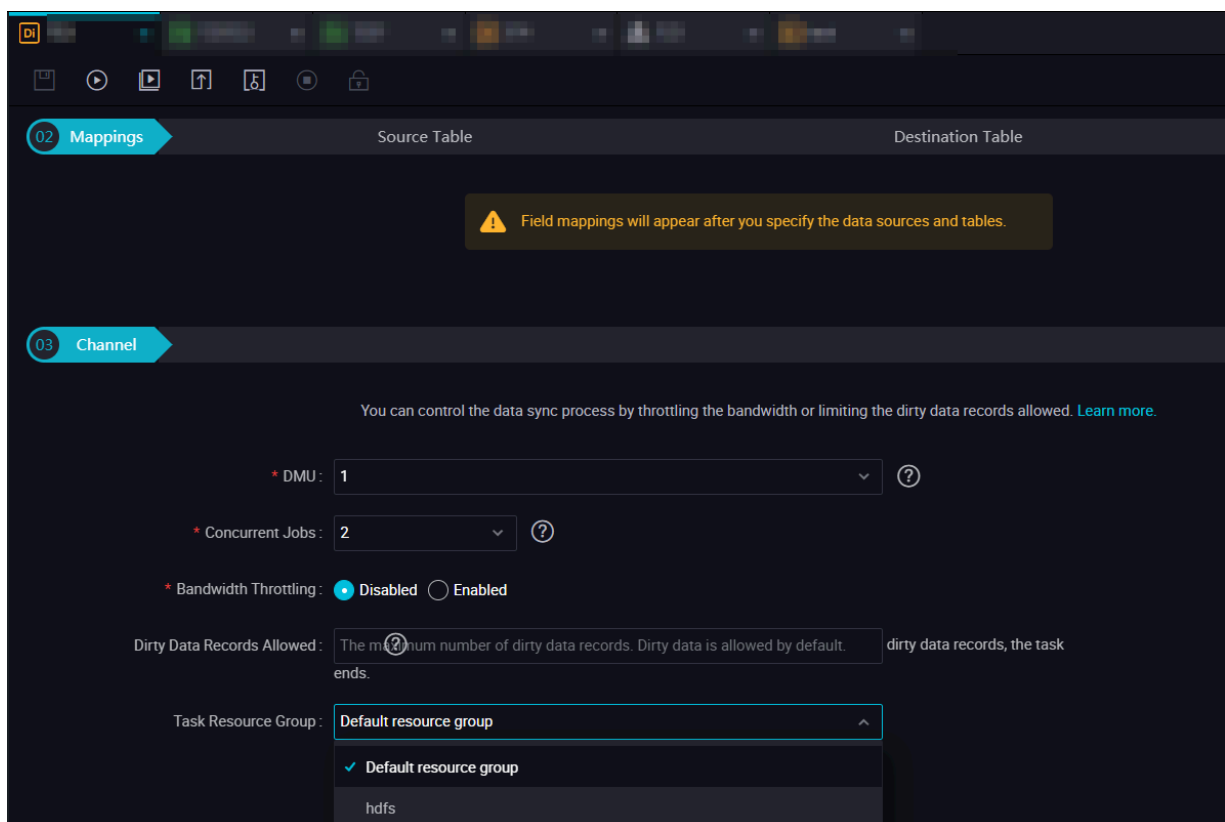
Switch to the admin user and run the following command:

```
/home/admin/alisatasknode/target/alisatasknode/bin/serverctl restart
```

- You need to enter your AccessKey when running this command. Keep the AccessKey information secure.

Select the custom resource group for a sync node

On the configuration tab of a sync node, select the custom resource group for **Resource Group** in the **Channel** section.



8 Appendixes

8.1 Connection configuration

8.1.1 Configure an AnalyticDB for MySQL 2.0 connection

An AnalyticDB for MySQL 2.0 connection allows you to read data from and write data to AnalyticDB for MySQL 2.0 by using AnalyticDB for MySQL 2.0 Reader and Writer. You can configure sync nodes for AnalyticDB for MySQL 2.0 by using the codeless user interface (UI) or code editor.

Context

Workspaces in the standard mode support the connection isolation feature. You can add connections for the development and production environments separately and isolate the connections to protect your data security. For more information, see [Connection isolation](#).

Procedure

1. Go to the **Data Source** page.
 - a) Log on to the [DataWorks console](#).
 - b) In the left-side navigation pane, click **Workspaces**.
 - c) In the top navigation bar, select the region where the target workspace resides. Find the target workspace and click **Data Integration** in the Actions column.
 - d) On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.
2. On the **Data Source** page, click **Add a Connection** in the upper-right corner.
3. In the **Add Connection** dialog box that appears, select **AnalyticDB for MySQL 2.0**.

4. In the **Add AnalyticDB for MySQL 2.0 Connection** dialog box that appears, set relevant parameters.

Add AnalyticDB for MySQL2.0 Connection

* Connection Name :
Enter a name.

Description :

* Applicable :
☒ Development
☐ Production

Environment

* Connection URL :
The format is IP address:port.

* Database :

* AccessKey ID :

* AccessKey Secret :

Resource Connectivity :


Resource Group Name	Type	Status	Test Time	Actions
	Custom Resource Group	Not Supported		Not Supported
	Custom Resource Group	Not Supported		Not Supported
	Custom Resource Group	Not Supported		Not Supported
Public (Default) Resource Group				Test Connection.

Attention : If the test fails, the possible reason is:

- The database is not started, please confirm that it has started normally.
- DataWorks cannot access the network where the database is located. Please make sure the network is connected with Aliyun.

Previous

Complete

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_) and must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Environment	The environment in which the connection is used. Valid values: Development and Production . <div>  Note: This parameter is available only when the workspace is in the standard mode. </div>

Parameter	Description
Connection URL	The URL for connecting to the AnalyticDB for MySQL 2.0 database. Specify the parameter in the IP address:Port format.
Database	The name of the AnalyticDB for MySQL 2.0 database.
AccessKey ID	The AccessKey ID used as the logon credential. You can view the AccessKey ID on the User Info page.
AccessKey Secret	The AccessKey secret used as the logon credential.

5. Click **Test Connection** in the Actions column of each resource group.

A sync node only uses one resource group. Therefore, you must test the connectivity of all the resource groups for Data Integration that your sync nodes use to connect to the data store so that sync nodes can run properly. For more information, see [Test data store connectivity](#).

6. After the connection passes the connectivity test, click **Complete**.

What's next

Now you have learned how to configure an AnalyticDB for MySQL 2.0 connection. You can proceed with the next tutorial. In the next tutorial, you will learn how to configure AnalyticDB for MySQL 2.0 Reader and Writer. .

8.1.2 Configure an SQL Server connection

An SQL Server connection allows you to read data from and write data to SQL Server by using SQL Server Reader and Writer. You can configure sync nodes for SQL Server by using the codeless user interface (UI) or code editor.

Workspaces in standard mode support the [connection isolation](#) feature. You can add connections for the development and production environments separately and isolate the connections to protect your data security.



Note:

Currently, only SQL Server 2005 and later are supported. When you add connections for SQL Server data stores deployed in Virtual Private Clouds (VPCs), note the following information:

- User-created SQL Server data store
 - You can configure sync nodes that involve such connections. However, connectivity testing is not supported. You can click **Complete** without testing the connectivity.
 - You must run such sync nodes on custom resource groups. Make sure that the custom resource groups can connect to the data stores.
- ApsaraDB RDS for SQL Server data store


You do not need to select the network type, because DataWorks automatically identifies the network type based on the information you enter for the ApsaraDB RDS for SQL Server instance.

Procedure

1. On the **Data Source** page that appears, click **Add Connection** in the upper-right corner.
2. In the **Add Connection** dialog box that appears, click **SQL Server** in the Relational Database section.
3. In the **Add SQL Server Connection** dialog box that appears, set the parameters.

The SQL Server connection type can be set to **ApsaraDB for RDS** or **JDBC Connection Mode**. You can select a type as required.

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


Parameter	Description
Connect To	The type of the connection. Here, set the value to ApsaraDB for RDS .
Connection Name	The name of the connection. The name can contain letters , digits, and underscores (_) and must start with a letter.
Description	The description of the connection. The description cannot exceed 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: Development and Production . <div>  Note: This parameter is available only when the workspace is in standard mode. </div>
Region	The region of the ApsaraDB RDS for SQL Server instance.

Parameter	Description
RDS Instance ID	The ID of the ApsaraDB RDS for SQL Server instance. You can view the ID in the ApsaraDB for RDS console.
RDS Instance Account ID	The ID of the Alibaba Cloud account used to purchase the ApsaraDB RDS for SQL Server instance
Database Name	The name of the database.
Username	The username for logging on to the database.
Password	The password for logging on to the database.

**Note:**

You must add the IP addresses or Classless Inter-Domain Routing (CIDR) blocks that you use to access the ApsaraDB RDS for SQL Server instance to a whitelist of the instance. For more information, see [Configure a whitelist](#).

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

Parameter	Description
Connect To	The type of the connection. Here, set the value to JDBC Connection Mode .
Connection Name	The name of the connection. The name can contain letters , digits, and underscores (_) and must start with a letter.
Description	The description of the connection. The description cannot exceed 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: Development and Production . <div>  Note: This parameter is available only when the workspace is in standard mode. </div>
JDBC URL	The Java Database Connectivity (JDBC) URL of the database, in the format of jdbc:sqlserver://ServerIP:Port; DatabaseName=Database.
Username	The username for logging on to the database.
Password	The password for logging on to the database.

4. Click **Test Connection**.

5. After the connectivity test is passed, click **Complete**.

Note on connectivity testing

- If the data store is a user-created one deployed on Elastic Compute Service (ECS) instances that reside on a classic network, we recommend that you use a custom resource group to run sync nodes that use the connection. The default resource group does not guarantee that it can connect to the data store over the network.
- If the data store is deployed in a VPC and you configure the connection in instance mode, the connectivity test checks whether the entered information is correct.
- In a VPC, if you use an internal address as the JDBC URL to configure the connection, the connectivity test will fail.
- If you use a public address as the JDBC URL to configure the connection, the connectivity test checks whether the entered information is correct.

What to do next

Now you have learned how to configure an SQL Server connection. You can proceed with the next tutorial. In the next tutorial, you will learn how to configure SQL Server Reader and Writer. For more information, see [Configure SQL Server Reader](#) and [Configure SQL Server Writer](#).

8.1.3 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.



Note:

Workspaces in standard mode support the [connection isolation](#) feature. You can add connections for the development and production environments separately and isolate the connections to protect your data security.

Procedure



1. On the **Data Source** page that appears, click **Add Connection** in the upper-right corner.
2. In the **Add Connection** dialog box that appears, click **MongoDB** in the NoSQL section.

3. In the Add MongoDB Connection dialog box that appears, set the parameters.

The MongoDB connection type can be set to **ApsaraDB for RDS** or **JDBC Connection Mode**.



- **ApsaraDB for RDS:** Classic networks are generally used. Classic networks in the same region can be connected, while classic networks in different regions cannot be connected.
- **JDBC Connection Mode:** The public network is generally used, which may cost you certain fees.

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

Parameter	Description
Connect To	The type of the connection. Here, 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 Alibaba Cloud account and perform authorization. Then, refresh this configuration page.
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_) and must start with a letter.
Description	The description of the connection. The description cannot exceed 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 of the MongoDB instance.
Instance ID	The ID of the ApsaraDB for MongoDB instance. You can view the ID in the ApsaraDB for MongoDB console.

Parameter	Description
Database Name	The name of the database you created in the ApsaraDB for MongoDB console. You can also specify the database username and password in the console.
Username	The username for logging on to the database.
Password	The password for logging on to the database.

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

Parameter	Description
Connect To	The type of the connection. Here, set the value to JDBC Connection Mode .
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_) and must start with a letter.
Description	The description of the connection. The description cannot exceed 80 characters in length.
Applicable Environment	<p>The environment in which the connection is used. Valid values: Development and Production.</p> <div>  Note: This parameter is available only when the workspace is in standard mode. </div>
Address	<p>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.</p> <div>  Note: You must add either public endpoints or internal endpoints. Do not mix public endpoints with internal endpoints. </div>
Database Name	The name of the database.
Username	The username for logging on to the database.

Parameter	Description
Password	The password for logging on to the database.

**Note:**

To configure a MongoDB connection in connection mode, follow these steps:

- a. Set Connect To to **JDBC Connection Mode**.
- b. In the **Add MongoDB Connection** dialog box that appears, set the parameters. Note that you must specify an internal endpoint as the address.
- c. Click **Complete** without testing the connectivity.
- d. Add a custom resource group for running sync nodes. For more information, see [Add a custom resource group](#).

4. Click **Test Connection**.

5. After the connectivity test is passed, click **Complete**.

**Note:**

- If the data store is deployed in a Virtual Private Cloud (VPC), set Connect To to JDBC Connection Mode.
- If the data store is deployed in a VPC, the connectivity test is not supported.

What to do next

Now you have learned how to configure a MongoDB connection. You can proceed with the next tutorial. In the next tutorial, you will learn how to configure MongoDB Reader and Writer.

8.1.4 Configure a Datahub connection

A Datahub connection allows you to write data in other data stores to Datahub by using Datahub Writer.


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

**Note:**

Workspaces in standard mode support the [connection isolation](#) feature. You can add connections for the development and production environments separately and isolate the connections to protect your data security.

Procedure

1. On the **Data Source** page that appears, click **Add Connection** in the upper-right corner.
2. In the **Add Connection** dialog box that appears, click **DataHub** in the Big Data Storage section.
3. In the Add DataHub Connection dialog box that appears, set the parameters.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_) and must start with a letter.
Description	The description of the connection. The description cannot exceed 80 characters in length.
Applicable Environment	<div>The environment in which the connection is used. Valid values: Development and Production.</div> <div> Note: This parameter is available only when the workspace is in standard mode.</div>
DataHub Endpoint	The endpoint of Datahub. This parameter is read-only, which is automatically obtained from system configurations.
DataHub Project	The ID of the Datahub project.
AccessKey ID	The AccessKey ID used as the logon credential. You can view the AccessKey ID on the User Info page.
AccessKey Secret	The AccessKey secret used as the logon credential.

4. Click **Test Connection**.
5. After the connectivity test is passed, click **Complete**.

The connectivity test checks whether the entered information is correct.

What to do next

Now you have learned how to configure a Datahub connection. You can proceed with the next tutorial. In the next tutorial, you will learn how to configure Datahub Writer.


8.1.5 Configure a DM connection

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

Workspaces in standard mode support the [connection isolation](#) feature. You can add connections for the development and production environments separately and isolate the connections to protect your data security.

Procedure

1. On the **Data Source** page that appears, click **Add Connection** in the upper-right corner.
2. In the **Add Connection** dialog box that appears, click **DM** in the Relational Database section.
3. In the **Add DM Connection** dialog box that appears, set the parameters.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_) and must start with a letter.
Description	The description of the connection. The description cannot exceed 80 characters in length.
Applicable Environment	<div>The environment in which the connection is used. Valid values: Development and Production.</div> <div> Note: This parameter is available only when the workspace is in standard mode.</div>
JDBC URL	The Java Database Connectivity (JDBC) URL of the database, in the format of jdbc:dm://ServerIP:Port/Database.
Username	The username for logging on to the database.
Password	The password for logging on to the database.

4. Click **Test Connection**.
5. After the connectivity test is passed, click **Complete**.

Note on connectivity testing

- If the data store is a user-created one deployed on Elastic Compute Service (ECS) instances that reside on a classic network, we recommend that you use a custom

resource group to run sync nodes that use the connection. The default resource group does not guarantee that it can connect to the data store over the network.

- If the data store is deployed in a VPC and you configure the connection in connection string mode, the connectivity test is not supported. You can click **Complete** without testing the connectivity.

8.1.6 Configure a DRDS connection

A Distributed Relational Database Service (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 user interface (UI) or code editor.



Note:

Workspaces in standard mode support the [connection isolation](#) feature. You can add connections for the development and production environments separately and isolate the connections to protect your data security.


Procedure

1. On the **Data Source** page that appears, click **Add Connection** in the upper-right corner.
2. In the **Add Connection** dialog box that appears, click **DRDS** in the Relational Database section.
3. In the Add DRDS Connection dialog box that appears, set the parameters.


The DRDS connection type can be set to **ApsaraDB for DRDS** or **JDBC Connection Mode**. You can select a type as required.

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

Parameter	Description
Connect To	The type of the connection. Here, set the value to ApsaraDB for DRDS .
Connection Name	The name of the connection. The name can contain letters , digits, and underscores (_) and must start with a letter.
Description	The description of the connection. The description cannot exceed 80 characters in length.

Parameter	Description
Applicable Environment	<p>The environment in which the connection is used. Valid values: Development and Production.</p> <div>  Note: This parameter is available only when the workspace is in standard mode. </div>
Instance ID	The ID of the DRDS instance. You can view the ID in the DRDS console.
Tenant Account ID	The ID of the Alibaba Cloud account used to purchase the DRDS instance. You can view your account ID on the Security Settings after logging on to the Alibaba Cloud console with your Alibaba Cloud account.
Database Name	The name of the database.
Username	The username for logging on to the database.
Password	The password for logging on to the database.

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

Parameter	Description
Connect To	The type of the connection. Here, set the value to JDBC Connection Mode .
Connection Name	The name of the connection. The name can contain letters , digits, and underscores (_) and must start with a letter.
Description	The description of the connection. The description cannot exceed 80 characters in length.
Applicable Environment	<p>The environment in which the connection is used. Valid values: Development and Production.</p> <div>  Note: This parameter is available only when the workspace is in standard mode. </div>
JDBC URL	The Java Database Connectivity (JDBC) URL of the database, in the format of jdbc:mysql://ServerIP:Port/Database.
Username	The username for logging on to the database.

Parameter	Description
Password	The password for logging on to the database.

**Note:**

You must add the IP addresses or Classless Inter-Domain Routing (CIDR) blocks that you use to access the DRDS instance to a whitelist of the instance. For more information, see [Configure a whitelist](#).

4. Click **Test Connection**.

5. After the connectivity test is passed, click **Complete**.

Note on connectivity testing

- If the data store is a user-created one deployed on Elastic Compute Service (ECS) instances that reside on a classic network, we recommend that you use a custom resource group to run sync nodes that use the connection. The default resource group does not guarantee that it can connect to the data store over the network.
- If the data store is deployed in a Virtual Private Cloud (VPC) and you configure the connection in instance mode, the connectivity test checks whether the entered information is correct.
- In a VPC, if you use an internal address as the JDBC URL to configure the connection, the connectivity test will fail.
- If you use a public address as the JDBC URL to configure the connection, the connectivity test checks whether the entered information is correct.

What to do 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.

8.1.7 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 user interface (UI) or code editor.

**Note:**

Workspaces in standard mode support the [Connection isolation](#) feature. You can add connections for the development and production environments separately and isolate the connections to protect your data security.

Procedure

1. On the **Data Source** page that appears, click **Add Connection** in the upper-right corner.
2. In the **Add Connection** dialog box that appears, click **FTP** in the Semi-structured storage section.
3. In the Add FTP Connection dialog box that appears, set the parameters.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_) and must start with a letter.
Description	The description of the connection. The description cannot exceed 80 characters in length.
Applicable Environment	<div>The environment in which the connection is used. Valid values: Development and Production.</div> <div> Note: This parameter is available only when the workspace is in standard mode.</div>
Protocol	The protocol used by the FTP server. Currently, only FTP and Secure File Transfer Protocol (SFTP) are supported.
Host	The address of the FTP server.
Port	The port of the FTP server. This parameter defaults to 21 if you select FTP as the adopted protocol, and defaults to 22 if you select SFTP.
Username	The username for logging on the FTP server.
Password	The password for logging on the FTP server.

4. Click **Test Connection**.
5. After the connectivity test is passed, click **Complete**.

Note on connectivity testing

- If the data store is a user-created one deployed on Elastic Compute Service (ECS) instances that reside on a classic network, we recommend that you use a custom resource group to run sync nodes that use the connection. The default resource group does not guarantee that it can connect to the data store over the network.

- If the data store is deployed in a Virtual Private Cloud (VPC), the connectivity test is not supported. You can click **Complete** without testing the connectivity.

What to do next

Now you have learned how to configure an FTP connection. You can proceed with the next tutorial. In the next tutorial, you will learn how to configure FTP Reader and Writer.


8.1.8 Configure an HDFS connection

A Hadoop Distributed File System (HDFS) connection allows you to read data from and write data to HDFS by using HDFS Reader and Writer. You can configure sync nodes for HDFS by using the code editor.

Workspaces in standard mode support the [connection isolation](#) feature. You can add connections for the development and production environments separately and isolate the connections to protect your data security.

Procedure

1. On the **Data Source** page that appears, click **Add Connection** in the upper-right corner.
2. In the **Add Connection** dialog box that appears, click **HDFS** in the Semi-structured storage section.
3. In the **Add HDFS Connection** dialog box that appears, set the parameters.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_) and must start with a letter.
Description	The description of the connection. The description cannot exceed 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.
DefaultFS	The address of the NameNode of HDFS, in the format of hdfs://ServerIP:Port.

4. Click **Test Connection**.

5. After the connectivity test is passed, click **Complete**.

The connectivity test checks whether the entered information is correct.

Note on connectivity testing

- If the data store is a user-created one deployed on Elastic Compute Service (ECS) instances that reside on a classic network, we recommend that you use a custom resource group to run sync nodes that use the connection. The default resource group does not guarantee that it can connect to the data store over the network.
- If the data store is deployed in a Virtual Private Cloud (VPC), the connectivity test is not supported. You can click **Complete** without testing the connectivity.

What to do next

Now you have learned how to configure an HDFS connection. You can proceed with the next tutorial. In the next tutorial, you will learn how to configure HDFS Reader and Writer. For more information, see [Configure HDFS Reader](#) and [Configure HDFS Writer](#).

8.1.9 Configure a LogHub connection

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



Note:

Workspaces in standard mode support the [connection isolation](#) feature. You can add connections for the development and production environments separately and isolate the connections to protect your data security.

Procedure

1. On the **Data Source** page that appears, click **Add Connection** in the upper-right corner.
2. In the **Add Connection** dialog box that appears, click **LogHub (Log Service)** in the Message Queue section.

3. In the Add LogHub (Log Service) Connection dialog box that appears, set the parameters.

Add LogHub Connection
✕

* Connection Name : test

Description :

* LogHub Endpoint : http://cn-hangzhou-intranet.log.aliyuncs.com ?


* Project : mgqtest

* AccessKey ID : ?

* AccessKey Secret :

Test Connection : Test Connection

Previous
Complete

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_) and must start with a letter.
Description	The description of the connection. The description cannot exceed 80 characters in length.
Applicable Environment	<p>The environment in which the connection is used. Valid values: Development and Production.</p> <div>  Note: This parameter is available only when the workspace is in standard mode. </div>
LogHub Endpoint	The LogHub endpoint, in the format of http://cn-shanghai.log.aliyun.com. For more information, see Service endpoints .
Project	The name of the LogHub project.
AccessKey ID and AccessKey Secret	The AccessKey ID and AccessKey secret used as logon credentials.

4. Click **Test Connection**.
5. After the connectivity test is passed, click **Complete**.

The connectivity test checks whether the entered information is correct.

What to do next

Now you have learned how to configure a LogHub connection. You can proceed with the next tutorial. In the next tutorial, you will learn how to configure LogHub Reader and Writer. For more information, see [Configure LogHub Reader](#).

8.1.10 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.




Note:

Workspaces in standard mode support the [connection isolation](#) feature. You can add connections for the development and production environments separately and isolate the connections to protect your data security.

Procedure

1. On the **Data Source** page that appears, click **Add Connection** in the upper-right corner.
2. In the **Add Connection** dialog box that appears, click **Memcache (OCS)** in the NoSQL section.

3. In the Add Memcache (OCS) Connection dialog box that appears, set the parameters.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_) and must start with a letter.
Description	The description of the connection. The description cannot exceed 80 characters in length.
Applicable Environment	<p>The environment in which the connection is used. Valid values: Development and Production.</p> <div>  Note: This parameter is available only when the workspace is in standard mode. </div>
Proxy Host	The IP address of the host or Memcache proxy, which can be viewed on the basic information page of the ApsaraDB for Memcache console.
Port	The port for accessing the ApsaraDB for Memcache instance. Default value: 11211 .
Username	The username for logging on to the ApsaraDB for Memcache instance.
Password	The password for logging on to the ApsaraDB for Memcache instance.

4. Click **Test Connection**.
5. After the connectivity test is passed, click **Complete**.

What to do next

Now you have learned how to configure a Memcache connection. You can proceed with the next tutorial. In the next tutorial, you will learn how to configure Memcache Writer.

8.1.11 Configure a MySQL connection

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

Workspaces in standard mode support the [connection isolation](#) feature. You can add connections for the development and production environments separately and isolate the connections to protect your data security.



Note:

When you add connections for MySQL data stores deployed in Virtual Private Clouds (VPCs), note the following information:

- User-created MySQL data store
 - You can configure sync nodes that involve such connections. However, connectivity testing is not supported. You can click **Complete** without testing the connectivity.
 - You must run such sync nodes on custom resource groups. Make sure that the custom resource groups can connect to the data stores.
- ApsaraDB RDS for MySQL data store

You do not need to select the network type, because DataWorks automatically identifies the network type based on the information you enter for the ApsaraDB RDS for MySQL instance.


Procedure

1. On the **Data Source** page that appears, click **Add Connection** in the upper-right corner.
2. In the **Add Connection** dialog box that appears, click **MySQL** in the Relational Database section.

3. In the Add MySQL Connection dialog box that appears, set the parameters.

The MySQL connection type can be set to **ApsaraDB for RDS** or **JDBC Connection Mode**.

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


Parameter	Description
Connect To	The type of the connection. Here, set the value to ApsaraDB for RDS .
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_) and must start with a letter.
Description	The description of the connection. The description cannot exceed 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 of the ApsaraDB RDS for MySQL instance.
RDS Instance ID	The ID of the ApsaraDB RDS for MySQL instance. You can view the ID in the ApsaraDB for RDS console.
RDS Instance Account ID	The ID of the Alibaba Cloud account used to purchase the ApsaraDB RDS for MySQL instance. You can view your account ID on the Security Settings page after logging on to the Alibaba Cloud console with your Alibaba Cloud account.
Database Name	The name of the database.
Username	The username for logging on to the database.
Password	The password for logging on to the database.



Note:

You must add the IP addresses or Classless Inter-Domain Routing (CIDR) blocks that you use to access the ApsaraDB RDS for MySQL instance to a whitelist of the instance. For more information, see [Configure a whitelist](#).

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

Parameter	Description
Connect To	The type of the connection. Here, set the value to JDBC Connection Mode .
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_) and must start with a letter.
Description	The description of the connection. The description cannot exceed 80 characters in length.
Applicable Environment	<div>The environment in which the connection is used. Valid values: Development and Production.</div> <div> Note: This parameter is available only when the workspace is in standard mode.</div>
JDBC URL	The Java Database Connectivity (JDBC) URL of the database, in the format of jdbc:mysql://ServerIP:Port/Database.
Username	The username for logging on to the database.
Password	The password for logging on to the database.

4. Click **Test Connection**.

5. After the connectivity test is passed, click **Complete**.

Note on connectivity testing

- If the data store is a user-created one deployed on Elastic Compute Service (ECS) instances that reside on a classic network, we recommend that you use a custom resource group to run sync nodes that use the connection. The default resource group does not guarantee that it can connect to the data store over the network.
- If the data store is deployed in a VPC and you configure the connection in instance mode, the connectivity test checks whether the entered information is correct.

- In a VPC, if you use an internal address as the JDBC URL to configure the connection, the connectivity test will fail.
- If you use a public address as the JDBC URL to configure the connection, the connectivity test checks whether the entered information is correct.

What to do next

Now you have learned how to configure a MySQL connection. You can proceed with the next tutorial. In the next tutorial, you will learn how to configure MySQL Reader and Writer. For more information, see [Configure MySQL Reader](#).


8.1.12 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 user interface (UI) or code editor.

Workspaces in standard mode support the [connection isolation](#) feature. You can add connections for the development and production environments separately and isolate the connections to protect your data security.

Procedure

1. On the **Data Source** page that appears, click **Add Connection** in the upper-right corner.
2. In the **Add Connection** dialog box that appears, click **Oracle** in the Relational Database section.
3. In the **Add Oracle Connection** dialog box that appears, set the parameters.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_) and must start with a letter.
Description	The description of the connection. The description cannot exceed 80 characters in length.
Applicable Environment	<div>The environment in which the connection is used. Valid values: Development and Production.</div> <div> Note: This parameter is available only when the workspace is in standard mode.</div>
JDBC URL	The Java Database Connectivity (JDBC) URL of the database, in the format of jdbc:oracle:thin:@ServerIP:Port:Database.

Parameter	Description
Username	The username for logging on to the database.
Password	The password for logging on to the database.

4. Click **Test Connection**.

5. After the connectivity test is passed, click **Complete**.

Note on connectivity testing

- If the data store is a user-created one deployed on Elastic Compute Service (ECS) instances that reside on a classic network, we recommend that you use a custom resource group to run sync nodes that use the connection. The default resource group does not guarantee that it can connect to the data store over the network.
- If the data store is deployed in a Virtual Private Cloud (VPC), the connectivity test is not supported. You can click **Complete** without testing the connectivity.

What to do next

Now you have learned how to configure an Oracle connection. You can proceed with the next tutorial. In the next tutorial, you will learn how to configure Oracle Reader and Writer. For more information, see [Configure Oracle Reader](#).

8.1.13 Configure an OSS connection

Alibaba Cloud Object Storage Service (OSS) is a secure and reliable service that allows you to store large amounts of objects.



Note:

- Workspaces in standard mode support the [Connection isolation](#) feature. You can add connections for the development and production environments separately and isolate the connections to protect your data security.
- For more information about OSS, see [OSS product introduction](#).
- For more information about OSS Java SDK, see [OSS Java SDK](#).

Procedure

1. On the **Data Source** page that appears, click **Add Connection** in the upper-right corner.
2. In the **Add Connection** dialog box that appears, click **OSS** in the Semi-structured storage section.

3. In the Add OSS Connection dialog box that appears, set the parameters.

Add OSS Connection

* Connection Name :

Description :

* Applicable :
☒ Development
☐ Production

Environment

* Endpoint :
?

* Bucket :
?

* AccessKey ID :
?

* AccessKey Secret :

Resource Connectivity :

Resource Group Name	Type	Status	Test Time	Actions
Public Resource Group		Not Tested		Test Connection.



?

Attention : If the test fails, the possible reason is:

- The database is not started, please confirm that it has started normally.
- DataWorks cannot access the network where the database is located. Please make sure the network is connected with Alivun.

Previous

Complete

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_) and must start with a letter.
Description	The description of the connection. The description cannot exceed 80 characters in length.
Applicable Environment	<p>The environment in which the connection is used. Valid values: Development and Production.</p> <div>  Note: This parameter is available only when the workspace is in standard mode. </div>
Endpoint	<p>The OSS endpoint, in the format of <code>http://oss.aliyuncs.com</code>. The OSS endpoint varies with the region.</p> <div>  Note: If you add the bucket name before the domain name, for example, <code>http://xxx.oss.aliyuncs.com</code>, the connection can pass the connectivity test but data synchronization will fail. </div>

Parameter	Description
Bucket	<p>The name of the OSS bucket. A bucket is a storage space that serves as a container for storing objects.</p> <p>You can create one or more buckets and add one or more objects to each bucket.</p> <p>DataWorks can only search for objects in the bucket specified here during data synchronization.</p>
AccessKey ID and AccessKey Secret	The AccessKey ID and AccessKey secret used as logon credentials.

4. Click **Test Connection**.

5. After the connectivity test is passed, click **Complete**.

**Note:**

When data in OSS is stored as Comma-Separated Values (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 parsed incorrectly.

Note on connectivity testing

- If the data store is a user-created one deployed on Elastic Compute Service (ECS) instances that reside on a classic network, we recommend that you use a custom resource group to run sync nodes that use the connection. The default resource group does not guarantee that it can connect to the data store over the network.
- If the data store is deployed in a Virtual Private Cloud (VPC), the connectivity test is not supported. You can click **Complete** without testing the connectivity.

What to do next

Now you have learned how to configure an OSS connection. You can proceed with the next tutorial. In the next tutorial, you will learn how to configure OSS Reader and Writer. For more information, see [Configure OSS Reader](#) and [Configure OSS Writer](#).

8.1.14 Configure a Table Store connection

Table Store is a Not only SQL (NoSQL) database service built on Apsara distributed operating system. It allows you to store and access large amounts of structured data in real time.


Workspaces in standard mode support the [connection isolation](#) feature. You can add connections for the development and production environments separately and isolate the connections to protect your data security.

**Note:**

For more information about Table Store, see [#unique_164](#).

Procedure

1. On the **Data Source** page that appears, click **Add Connection** in the upper-right corner.
2. In the **Add Connection** dialog box that appears, click **Table Store (OTS)** in the NoSQL section.
3. In the **Add Table Store (OTS) Connection** dialog box that appears, set the parameters.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_) and must start with a letter.
Description	The description of the connection. The description cannot exceed 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: Development and Production . <div> Note: This parameter is available only when the workspace is in standard mode.</div>
Endpoint	The endpoint of the Table Store service.
Table Store Instance ID	The ID of the Table Store instance.
AccessKey ID	The AccessKey ID used as the logon credential. You can view the AccessKey ID on the User Info page.
Access Key Secret	The AccessKey secret used as the logon credential.

4. Click **Test Connection**.
5. After the connectivity test is passed, click **Complete**.

Note on connectivity testing

- If the data store is a user-created one deployed on Elastic Compute Service (ECS) instances that reside on a classic network, we recommend that you use a custom resource group to run sync nodes that use the connection. The default resource group does not guarantee that it can connect to the data store over the network.
- If the data store is deployed in a VPC, the connectivity test is not supported. You can click **Complete** without testing the connectivity.

What to do next

Now you have learned how to configure a Table Store connection. You can proceed with the next tutorial. In the next tutorial, you will learn how to configure Table Store Reader. For more information, see [Configure Table Store Reader](#).

8.1.15 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 user interface (UI) or code editor.



Note:

Workspaces in standard mode support the [connection isolation](#) feature. You can add connections for the development and production environments separately and isolate the connections to protect your data security.

When you add connections for PostgreSQL data stores deployed in Virtual Private Clouds (VPCs), note the following information:

- User-created PostgreSQL data store
 - You can configure sync nodes that involve such connections. However, connectivity testing is not supported. You can click **Complete** without testing the connectivity.
 - You must run such sync nodes on custom resource groups. Make sure that the custom resource groups can connect to the data stores.
- ApsaraDB RDS for PostgreSQL data store


You do not need to select the network type, because DataWorks automatically identifies the network type based on the information you enter for the ApsaraDB RDS for PostgreSQL instance.

Procedure

1. On the **Data Source** page that appears, click **Add Connection** in the upper-right corner.
2. In the **Add Connection** dialog box that appears, click **PostgreSQL** in the Relational Database section.
3. In the Add PostgreSQL Connection dialog box that appears, set the parameters.


The PostgreSQL connection type can be set to **ApsaraDB for RDS** or **JDBC Connection Mode**. You can select a type as required.

The following table describes the parameters that appear after you set **Connect To** 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 (_) and must start with a letter.
Description	The description of the connection. The description cannot exceed 80 characters in length.
Applicable Environment	<p>The environment in which the connection is used. Valid values: Development and Production.</p> <div>  Note: This parameter is available only when the workspace is in standard mode. </div>
Region	The region of the ApsaraDB RDS for PostgreSQL instance.
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 Alibaba Cloud account used to purchase the ApsaraDB RDS for PostgreSQL instance. You can view your account ID on the Security Settings page after logging on to the Alibaba Cloud console with your Alibaba Cloud account.
Database Name	The name of the database.
Username	The username for logging on to the database.

Parameter	Description
Password	The password for logging on to the database.

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

Parameter	Description
Connect To	The type of the connection. In this example, set the value to JDBC Connection Mode .
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_) and must start with a letter.
Description	The description of the connection. The description cannot exceed 80 characters in length.
Applicable Environment	<p>The environment in which the connection is used. Valid values: Development and Production.</p> <div>  Note: This parameter is available only when the workspace is in standard mode. </div>
JDBC URL	The Java Database Connectivity (JDBC) URL of the database, in the format of jdbc:postgresql://ServerIP:Port/Database.
Username	The username for logging on to the database.
Password	The password for logging on to the database.

4. Click **Test Connection**.

5. After the connectivity test is passed, click **Complete**.

Note on connectivity testing

- If the data store is a user-created one deployed on Elastic Compute Service (ECS) instances that reside on a classic network, we recommend that you use a custom resource group to run sync nodes that use the connection. The default resource group does not guarantee that it can connect to the data store over the network.
- If the data store is deployed in a VPC and you configure the connection in instance mode, the connectivity test checks whether the entered information is correct.
- In a VPC, if you use an internal address as the JDBC URL to configure the connection, the connectivity test will fail.

- If you use a public address as the JDBC URL to configure the connection, the connectivity test checks whether the entered information is correct.

What to do next

Now you have learned how to configure a PostgreSQL connection. You can proceed with the next tutorial. In the next tutorial, you will learn how to configure PostgreSQL Reader and Writer.

8.1.16 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 codeless user interface (UI) or code editor.

Remote Directory Server (Redis) is a document-based Not only SQL (NoSQL) database that provides both persistent and in-memory database services. Based on the highly reliable two-node hot standby architecture and the cluster architecture that can be seamlessly scaled, Redis can provide high read and write performance and flexible capacity to meet changing business needs.

Workspaces in standard mode support the [connection isolation](#) feature. You can add connections for the development and production environments separately and isolate the connections to protect your data security.


Procedure

1. On the **Data Source** page that appears, click **Add Connection** in the upper-right corner.
2. In the **Add Connection** dialog box that appears, click **Redis** in the NoSQL section.
3. In the **Add Redis Connection** dialog box that appears, set the parameters.


The Redis connection type can be set to **ApsaraDB for RDS** or **JDBC Connection Mode**. You can select a type as required.

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

Parameter	Description
Connect To	The type of the connection. Here, set the value to ApsaraDB for RDS .
Connection Name	The name of the connection. The name can contain letters , digits, and underscores (_) and must start with a letter.

Parameter	Description
Description	The description of the connection. The description cannot exceed 80 characters in length.
Applicable Environment	<p>The environment in which the connection is used. Valid values: Development and Production.</p> <div>  Note: This parameter is available only when the workspace is in standard mode. </div>
Region	The region of the ApsaraDB for Redis instance.
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 for accessing ApsaraDB for Redis. Leave it blank if no password is required.

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

Parameter	Description
Connect To	<p>The type of the connection. Here, set the value to JDBC Connection Mode.</p> <p>If you select this type of connection, you must run sync nodes on custom resource groups. For more information, click here in the wizard.</p>
Connection Name	The name of the connection. The name can contain letters , digits, and underscores (_) and must start with a letter.
Description	The description of the connection. The description cannot exceed 80 characters in length.
Applicable Environment	<p>The environment in which the connection is used. Valid values: Development and Production.</p> <div>  Note: This parameter is available only when the workspace is in standard mode. </div>
Server Address	The server address in the host:port format.
Add Server Address	Click the button to add a server address in the host:port format.

Parameter	Description
Redis Password	The password for accessing the Redis service.

4. Click **Test Connection**.

5. After the connectivity test is passed, click **Complete**.

What to do next

Now you have learned how to configure a Redis connection. You can proceed with the next tutorial. In the next tutorial, you will learn how to configure Redis Writer.

8.1.17 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 user interface (UI) or code editor.



Note:

Workspaces in standard mode support the [connection isolation](#) feature. You can add connections for the development and production environments separately and isolate the connections to protect your data security.

When you add connections for HybridDB for MySQL data stores deployed in Virtual Private Clouds (VPCs), note the following information:

- User-created MySQL data store
 - You can configure sync nodes that involve such connections. However, connectivity testing is not supported. You can click **Complete** without testing the connectivity.
 - You must run such sync nodes on custom resource groups. Make sure that the custom resource groups can connect to the data stores.
- HybridDB for MySQL data store: You do not need to select the network type, because DataWorks automatically identifies the network type based on the information you enter for the HybridDB for MySQL instance.

Procedure

1. On the **Data Source** page, click **Add Connection** in the upper-right corner.
2. In the **Add Connection** dialog box that appears, click **HybridDB for MySQL** in the Relational Database section.

3. In the Add HybridDB for MySQL connection dialog box that appears, set the parameters.

The HybridDB for MySQL connection type can be set to **ApsaraDB for AnalyticDB** or **JDBC Connection Mode**.

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

Add HybridDB for MySQL Connection

* Connect To: ApsaraDB for AnalyticDB

* Connection Name: Enter a name.

Description:

* Applicable: ☒ Development ☐ Production

Environment

* Instance ID: ?

* Tenant Account ID: ?

* Database Name:

* Username:

* Password:

Test Connection: Test Connection

i To establish a connection to the database, add the IP addresses that you use to access the database to the whitelist. [Learn more about the process.](#)
Ensure that the database is available.

Previous Complete

Parameter	Description
Connect To	The type of the connection. Here, set the value to ApsaraDB for AnalyticDB .
Connection Name	The name of the connection. The name can contain letters , digits, and underscores (_) and must start with a letter.
Description	The description of the connection.

Parameter	Description
Applicable Environment	The environment in which the connection is used. Valid values: Development and Production.
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 Alibaba Cloud account used to purchase the HybridDB for MySQL instance. You can view your account ID on the Security Settings page after logging on to the Alibaba Cloud console with your Alibaba Cloud account.
Username	The username for logging on to the database.
Password	The password for logging on to the database.

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

Add HybridDB for MySQL Connection

* Connect To : ☐ ApsaraDB for AnalyticDB ☒ JDBC Connection Mode

* Connection Name : Enter a name.

Description :

* Applicable : ☒ Development ☐ Production

Environment

* JDBC URL : jdbc:mysql://ServerIP:Port/Database

* Username :

* Password :

Resource Connectivity :

Resource Group Name	Type	Status	Test Time	Actions
Public (Default) Resource Group				Test Connection.


Attention : If the test fails, the possible reason is:

1. The database is not started, please confirm that it has started normally.
2. DataWorks cannot access the network where the database is located. Please make sure the network is connected with Aliyun.

Previous

Complete

Parameter	Description
Connect To	The type of the connection. Here, set the value to JDBC Connection Mode .

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_) and must start with a letter.
Description	The description of the connection. The description cannot exceed 80 characters in length.
Applicable Environment	<p>The environment in which the connection is used. Valid values: Development and Production.</p> <div>  Note: This parameter is available only when the workspace is in standard mode. </div>
JDBC URL	The Java Database Connectivity (JDBC) URL of the database, in the format of <code>jdbc:mysql://ServerIP:Port/Database</code> .
Username	The username for logging on to the database.
Password	The password for logging on to the database.

4. Click **Test Connection**.

5. After the connectivity test is passed, click **Complete**.



Note:

You must add the IP addresses or Classless Inter-Domain Routing (CIDR) blocks that you use to access the HybridDB for MySQL instance to a whitelist of the instance. For more information, see [Configure a whitelist](#).

Note on connectivity testing

- If the data store is a user-created one deployed on Elastic Compute Service (ECS) instances that reside on a classic network, we recommend that you use a custom resource group to run sync nodes that use the connection. The default resource group does not guarantee that it can connect to the data store over the network.
- If the data store is deployed in a VPC and you configure the connection in instance mode, the connectivity test checks whether the instance ID, account ID, username, and password you entered are correct.
- In a VPC, if you use an internal address as the JDBC URL to configure the connection, the connectivity test will fail.

- If you use a public address as the JDBC URL to configure the connection, the connectivity test checks whether the JDBC URL, username, and password you entered are correct.

What to do next

Now you have learned how to configure a HybridDB for MySQL connection. You can proceed with the next tutorial. In the next tutorial, you will learn how to configure HybridDB for MySQL Reader and Writer. For more information, see [Configure HybridDB for MySQL Reader](#).

8.1.18 Configure an AnalyticDB for PostgreSQL Connection

An AnalyticDB for PostgreSQL connection allows you to read data from and write data to AnalyticDB for PostgreSQL by using AnalyticDB for PostgreSQL Reader and Writer. You can configure sync nodes for AnalyticDB for PostgreSQL by using the codeless user interface (UI) or code editor.



Note:

Workspaces in standard mode support the [connection isolation](#) feature. You can add connections for the development and production environments separately and isolate the connections to protect your data security.

When you add connections for AnalyticDB for PostgreSQL data stores deployed in Virtual Private Clouds (VPCs), note the following information:

- User-created PostgreSQL data store
 - You can configure sync nodes that involve such connections. However, connectivity testing is not supported. You can click **Complete** without testing the connectivity.
 - You must run such sync nodes on custom resource groups. Make sure that the custom resource groups can connect to the data stores.
- AnalyticDB for PostgreSQL data store

You do not need to select the network type, because DataWorks automatically identifies the network type based on the information you enter for the AnalyticDB for PostgreSQL instance.


Procedure

1. On the **Data Source** page, click **Add Connection** in the upper-right corner.
2. In the **Add Connection** dialog box that appears, click **AnalyticDB for PostgreSQL** in the Relational Database section.

3. In the Add AnalyticDB for PostgreSQL Connection dialog box that appears, set the parameters.


The AnalyticDB for PostgreSQL connection type can be set to **ApsaraDB for AnalyticDB** or **JDBC Connection Mode**.

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

Parameter	Description
Connect To	The type of the connection. Here, set the value to ApsaraDB for AnalyticDB .
Connection Name	The name of the connection. The name can contain letters , digits, and underscores (_) and must start with a letter.
Description	The description of the connection. The description cannot exceed 80 characters in length.
Applicable Environment	<p>The environment in which the connection is used. Valid values: Development and Production.</p> <div>  Note: This parameter is available only when the workspace is in standard mode. </div>
Instance ID	The ID of the AnalyticDB for PostgreSQL instance. You can view the ID in the AnalyticDB for PostgreSQL console.
Tenant Account ID	The ID of the Alibaba Cloud account used to purchase the AnalyticDB for PostgreSQL instance. You can view your account ID on the Security Settings page after logging on to the Alibaba Cloud console with your Alibaba Cloud account.
Database Name	The name of the database.
Username	The username for logging on to the database.
Password	The password for logging on to the database.

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

Parameter	Description
Connect To	The type of the connection. Here, set the value to JDBC Connection Mode .

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_) and must start with a letter.
Description	The description of the connection. The description cannot exceed 80 characters in length.
Applicable Environment	<p>The environment in which the connection is used. Valid values: Development and Production.</p> <div> Note: This parameter is available only when the workspace is in standard mode.</div>
JDBC URL	The Java Database Connectivity (JDBC) URL of the database, in the format of <code>jdbc:postgresql://ServerIP:Port/Database</code> .
Username	The username for logging on to the database.
Password	The password for logging on to the database.

4. Click **Test Connection**.

5. After the connectivity test is passed, click **Complete**.

**Note:**

You must add the IP addresses or Classless Inter-Domain Routing (CIDR) blocks that you use to access the AnalyticDB for PostgreSQL instance to a whitelist of the instance. For more information, see [Configure a whitelist](#).

Note on connectivity testing

- If the data store is a user-created one deployed on Elastic Compute Service (ECS) instances that reside on a classic network, we recommend that you use a custom resource group to run sync nodes that use the connection. The default resource group does not guarantee that it can connect to the data store over the network.
- If the data store is deployed in a VPC and you configure the connection in instance mode, the connectivity test checks whether the entered information is correct.
- In a VPC, if you use an internal address as the JDBC URL to configure the connection, the connectivity test will fail.
- If you use a public address as the JDBC URL to configure the connection, the connectivity test checks whether the entered information is correct.

What to do next

Now you have learned how to configure an AnalyticDB for PostgreSQL connection. You can proceed with the next tutorial. In the next tutorial, you will learn how to configure AnalyticDB for PostgreSQL Reader and Writer. For more information, see [Configure AnalyticDB for PostgreSQL Reader](#).

8.1.19 Configure a POLARDB connection

A POLARDB connection allows you to read data from and write data to ApsaraDB for POLARDB by using POLARDB Reader and Writer. You can configure sync nodes for ApsaraDB for POLARDB by using the codeless user interface (UI) or code editor.


**Note:**

Currently, POLARDB connections do not support custom resource groups. Use the default resource group. If you want to use a custom resource group, add a MySQL connection in **connection mode**. For more information, see [Configure a MySQL connection](#).

Procedure

1. On the **Data Source** page, click **Add Connection** in the upper-right corner.
2. In the **Add Connection** dialog box that appears, click **POLARDB** in the Relational Database section.
3. In the **Add POLARDB Connection** dialog box that appears, set the parameters.

Parameter	Description
Connect To	The type of the connection. Set the value to ApsaraDB for POLARDB .
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_) and must start with a letter.
Description	The description of the connection. The description cannot exceed 80 characters in length.

Parameter	Description
Applicable Environment	<p>The environment in which the connection is used. Valid values: Development and Production.</p> <div>  Note: This parameter is available only when the workspace is in standard mode. Workspaces in standard mode support the connection isolation feature. You can add connections for the development and production environments separately and isolate the connections to protect your data security. </div>
Cluster ID	The ID of the ApsaraDB for POLARDB cluster. You can view the ID in the ApsaraDB for POLARDB console.
POLARDB Instance Tenant Account ID	The ID of the Alibaba Cloud account used to purchase the ApsaraDB for POLARDB instance. You can view your account ID on the Security Settings page after logging on to the Alibaba Cloud console with your Alibaba Cloud account.
Database Name	The name of the database.
Username	The username for logging on to the database.
Password	The password for logging on to the database.

When you add connections for POLARDB data stores deployed in Virtual Private Clouds (VPCs), note the following information:

- User-created POLARDB data store
 - You can configure sync nodes that involve such connections. However, connectivity testing is not supported. You can click **Complete** without testing the connectivity.
 - You must run such sync nodes on custom resource groups. Make sure that the custom resource groups can connect to the data stores.
- ApsaraDB for POLARDB data store

You do not need to select the network type, because DataWorks automatically identifies the network type based on the information you enter for the ApsaraDB for POLARDB instance.

4. Click **Test Connection**.

5. After the connectivity test is passed, click **Complete**.

**Note:**

You must add the IP addresses or Classless Inter-Domain Routing (CIDR) blocks that you use to access the ApsaraDB for POLARDB instance to a whitelist of the instance. For more information, see [Configure a whitelist](#).

Note on connectivity testing

- If the data store is a user-created one deployed on Elastic Compute Service (ECS) instances that reside on a classic network, we recommend that you use a custom resource group to run sync nodes that use the connection. The default resource group does not guarantee that it can connect to the data store over the network.
- You can add a connection for a POLARDB data store deployed in a VPC based on the ID of the ApsaraDB for POLARDB instance. In this case, the reverse proxy feature is provided to guarantee that the data store can be connected.

What to do next

Now you have learned how to configure a POLARDB connection. You can proceed with the next tutorial. In the next tutorial, you will learn how to configure POLARDB Reader and Writer. For more information, see [Configure POLARDB Reader](#) and [Configure POLARDB Writer](#).

8.1.20 Configure an AnalyticDB for MySQL 3.0 connection

An AnalyticDB for MySQL 3.0 connection allows you to read data from and write data to AnalyticDB for MySQL 3.0. You can configure sync nodes for AnalyticDB for MySQL 3.0 by using the codeless user interface (UI) or code editor.

Workspaces in standard mode support the [connection isolation](#) feature. You can add connections for the development and production environments separately and isolate the connections to protect your data security.


Procedure

1. On the **Data Source** page, click **Add Connection** in the upper-right corner.
2. In the **Add Connection** dialog box that appears, click **AnalyticDB for MySQL 3.0** in the Relational Database section.

3. In the **Add AnalyticDB for MySQL 3.0 Connection** dialog box that appears, set the parameters.


The AnalyticDB for MySQL 3.0 connection type can be set to **AnalyticDB for MySQL** or **JDBC Connection Mode**.

- The following table describes the parameters that appear after you set **Connect To** to **AnalyticDB for MySQL**.

Parameter	Description
Connect To	The type of the connection. Here, set the value to AnalyticDB for MySQL .
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_) and must start with a letter.
Description	The description of the connection. The description cannot exceed 80 characters in length.
Applicable Environment	<p>The environment in which the connection is used. Valid values: Development and Production.</p> <div>  Note: This parameter is available only when the workspace is in standard mode. </div>
Region	The region of the AnalyticDB for MySQL 3.0 instance.
ADS Instance ID	The ID of the AnalyticDB for MySQL 3.0 instance. You can view the ID on the AnalyticDB for MySQL 3.0 console.
Database Name	The name of the database.
Username	The username for logging on to the database.
Password	The password for logging on to the database.

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

Parameter	Description
Connect To	The type of the connection. Here, set the value to JDBC Connection Mode .
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_) and must start with a letter.

Parameter	Description
Description	The description of the connection. The description cannot exceed 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 Java Database Connectivity (JDBC) URL of the database, in the format of jdbc:mysql://ServerIP:Port/Database.
Username	The username for logging on to the database.
Password	The password for logging on to the database.

4. Click **Test Connection**.

5. After the connectivity test is passed, click **Complete**.

Note on connectivity testing

- If the data store is a user-created one deployed on Elastic Compute Service (ECS) instances that reside on a classic network, we recommend that you use a custom resource group to run sync nodes that use the connection. The default resource group does not guarantee that it can connect to the data store over the network.
- If the data store is deployed in a Virtual Private Cloud (VPC) and you configure the connection in instance mode, the connectivity test checks whether the entered information is correct.

8.1.21 Configure a DLA connection

This topic describes how to configure a Data Lake Analytics (DLA) connection.




Note:

Workspaces in standard mode support the [connection isolation](#) feature. You can add connections for the development and production environments separately and isolate the connections to protect your data security.

Procedure

1. On the **Data Source** page, click **Add Connection** in the upper-right corner.

2. In the **Add Connection** dialog box that appears, click **Data Lake Analytics(DLA)** in the Big Data Storage section.
3. In the Add Data Lake Analytics(DLA) Connection dialog box that appears, set the parameters.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_) and must start with a letter.
Description	The description of the connection. The description cannot exceed 80 characters in length.
Applicable Environment	<p>The environment in which the connection is used. Valid values: Development and Production.</p> <div>  Note: This parameter is available only when the workspace is in standard mode. </div>
Connection URL	The URL of the connection, in the format of Address:Port.
Database	The name of the database.
Username	The username for logging on to the database.
Password	The password for logging on to the database.

4. Click **Test Connection**.
5. After the connectivity test is passed, click **Complete**.

Note on connectivity testing

- If the data store is a user-created one deployed on Elastic Compute Service (ECS) instances that reside on a classic network, we recommend that you use a custom resource group to run sync nodes that use the connection. The default resource group does not guarantee that it can connect to the data store over the network.
- If the data store is deployed in a Virtual Private Cloud (VPC), the connectivity test is not supported. You can click **Complete** without testing the connectivity.

If the data store is deployed in a VPC, you must use an exclusive resource group to run sync nodes that use the connection. For more information, see [#unique_166](#) and [#unique_167](#).

8.1.22 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

Workspaces in the standard mode support the [connection isolation](#) feature. You can add connections for the development and production environments separately and isolate the connections to protect your data security.

MaxCompute offers a comprehensive data import scheme to support fast computing for large amounts of data. DataWorks generates the default connection `odps_first` for a workspace after you bind a MaxCompute computing engine to the workspace for the first time. From then on, DataWorks generates a computing engine connection named in the format of **0_Region ID_Engine name** each time you bind a new MaxCompute computing engine to the workspace.

The MaxCompute project names of the default connection and the default computing engine connections are the same as the names of the MaxCompute projects that serve as computing engines. You can change the AccessKey of the default connection. To change the AccessKey, follow these steps: In the DataWorks console, move the pointer over your username in the top navigation bar and select User Info. On the Personal Account page that appears, click Save AccessKey. In the dialog box that appears, enter the new AccessKey ID and AccessKey secret, and then click Save AccessKey. Note the following rules when you change the AccessKey:

- You can only switch from the AccessKey of one Alibaba Cloud account to that of another Alibaba Cloud account.
- Before changing the AccessKey, make sure that no data integration or data analytics nodes are running in DataWorks. A custom MaxCompute connection can use the AccessKey of a Resource Access Management (RAM) user.

Procedure

1. Go to the **Data Source** page.
 - a) Log on to the [DataWorks console](#).
 - b) In the left-side navigation pane, click **Workspaces**.
 - c) In the top navigation bar, select the region where the target workspace resides. Find the target workspace and click **Data Integration** in the Actions column.
 - d) On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.
2. On the **Data Source** page, click **Add a Connection** in the upper-right corner.
3. In the **Add Connection** dialog box that appears, select **ODPS**.
4. In the **Add ODPS Connection** dialog box that appears, set relevant parameters.

Add ODPS Connection

* Connection Name :

Enter a name.

Description :

* Applicable :

☒ Development ☐ Production

Environment

* ODPS Endpoint :

http://service.odps.aliyun.com/api

Tunnel Endpoint :

* MaxCompute Project :

Enter a MaxCompute project name.

Name

* AccessKey ID :

?

* AccessKey Secret :

Resource Connectivity :

Resource Group Name	Type	Status	Test Time	Actions
Public Resource Group		Not Tested		Test Connection.

?


Attention :

If the test fails, the possible reason is:

Previous

Complete

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

Parameter	Description
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 the standard mode.
ODPS Endpoint	The endpoint of the MaxCompute project. This parameter is read-only, which is automatically obtained from system configurations.
Tunnel Endpoint	The endpoint of the MaxCompute Tunnel service. For more information, see #unique_169 .
MaxCompute Project Name	The name of the MaxCompute project.
AccessKey ID	The AccessKey ID used as the logon credential. You can view the AccessKey ID on the User Info page.
AccessKey Secret	The AccessKey secret used as the logon credential.

5. Click **Test Connection** in the Actions column of each resource group.

A sync node only uses one resource group. Therefore, you must test the connectivity of all the resource groups for Data Integration that your sync nodes use to connect to the data store so that sync nodes can run properly. For more information, see [Test data store connectivity](#).

6. After the connection passes the connectivity test, click **Complete**.

What's next

Now you have learned how to configure a MaxCompute connection. You can proceed with the next tutorial. In the next tutorial, you will learn how to configure MaxCompute Reader and Writer. For more information, see [Configure MaxCompute Reader](#) and [Configure MaxCompute Writer](#).

8.1.23 Configure a GDB connection

A GDB connection allows you to write data to Graph Database (GDB). You can configure sync nodes for GDB by using the code editor.

GDB is a real-time and reliable online database service that makes it easy to store and navigate the relationships between highly connected datasets. GDB supports the Property

Graph model and its query language Apache TinkerPop Gremlin, allowing you to easily build queries that efficiently navigate highly connected datasets.




Note:

Workspaces in standard mode support the [Connection isolation](#) feature. You can add connections for the development and production environments separately and isolate the connections to protect your data security.

Procedure

1. On the **Data Source** page, click **Add Connection** in the upper-right corner.
2. In the **Add Connection** dialog box that appears, click **Graph Database** in the NoSQL section.
3. In the Add GDB Connection dialog box that appears, set the parameters.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_) and must start with a letter.
Description	The description of the connection. The description cannot exceed 80 characters in length.
Applicable Environment	<p>The environment in which the connection is used. Valid values: Development and Production.</p> <div>  <p>Note: This parameter is available only when the workspace is in standard mode.</p> </div>
Figure instance domain name	The endpoint of the GDB instance. Log on to the GDB console and click Instances in the left-side navigation pane. On the Instances page that appears, find the target instance and click View Instance Details in the Actions column. On the page that appears, view the value of Internal Endpoint in the Basic Information section.
Graph instance Port	The port for accessing GDB instance. Log on to the GDB console and click Instances in the left-side navigation pane. On the Instances page that appears, find the target instance and click View Instance Details in the Actions column. On the page that appears, view the value of Internal Port in the Basic Information section.

Parameter	Description
Figure example account	The username for accessing the GDB instance. Log on to the GDB console and click Instances in the left-side navigation pane. On the Instances page that appears, find the target instance and click View Instance Details in the Actions column. On the page that appears, click Account Management in the left-side navigation pane and view the value of Username .
Figure instance password	The password for accessing the GDB instance. Log on to the GDB console and click Instances in the left-side navigation pane. On the Instances page that appears, find the target instance and click View Instance Details in the Actions column. On the page that appears, click Account Management in the left-side navigation pane and view the value of Password .

4. Click **Test Connection**.



Note:

The connectivity test checks whether the default resource group can connect to the data store. In this case, the data store cannot be connected over the network. As a result, the connectivity test fails. When you commit a sync node that uses the connection, you must use an exclusive resource group.

5. After the connectivity test is passed, click **Complete**.

What to do next

Now you have learned how to configure a GDB connection. You can proceed with the next tutorial. In the next tutorial, you will learn how to configure GDB Writer.

8.1.24 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 user interface (UI) or code editor.

Context

Workspaces in the standard mode support the connection isolation feature. You can add connections for the development and production environments separately and isolate the connections to protect your data security. For more information, see [Connection isolation](#).

Procedure

1. Go to the **Data Source** page.
 - a) Log on to the [DataWorks console](#).
 - b) In the left-side navigation pane, click **Workspaces**.
 - c) In the top navigation bar, select the region where the target workspace resides. Find the target workspace and click **Data Integration** in the Actions column.
 - d) On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.
2. On the **Data Source** page, click **Add a Connection** in the upper-right corner.
3. In the **Add Connection** dialog box that appears, click **ApsaraDB for OceanBase** in the Relational Database section.

4. In the **Add ApsaraDB for OceanBase Connection** dialog box that appears, set the parameters as required.

The values that are optional for the Connect To parameter include **ApsaraDB for Oceanbase** and **Connection Mode**.

- The following table describes the parameters that you need to set if you select **ApsaraDB for Oceanbase** for **Connect To**.

Add ApsaraDB for OceanBase Connection

* Connect To : ☒ ApsaraDB for Oceanbase ☐ Connection Mode

* Connection Name : Enter a name.

Description :

* Environment : ☒ Development ☐ Production

地区 : Please Select

* 集群 : ?

* 租户 : ?

* 主帐号ID : ?

* 数据库名 :

* 用户名 :

* 密码 :

Resource Connectivity :

Resource Group Name	Type	Status	Test Time	Actions
Public Resource Group	Not Supported	Not Supported		Not Supported

?


Attention : If the test fails, the possible reason is:

- The database is not started, please confirm that it has started normally.
- DataWorks cannot access the network where the database is located. Please make sure the network is connected with Aliyun.
- DataWorks is prohibited by the network firewall where the database is located. Please add [whitelist](#).

Previous

Complete

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

Parameter	Description
Environment	<p>The environment in which the connection is used. Valid values: Development and Production.</p> <div> Note: This parameter is available only when the workspace is in the standard mode.</div>
Region	The region of the ApsaraDB for OceanBase cluster.
Cluster	The ID of the ApsaraDB for OceanBase cluster. You can view the cluster ID in the ApsaraDB for OceanBase console .
Tenant	The ID of the tenant. You can view the tenant ID in the ApsaraDB for OceanBase console.
Account ID	The ID of the Alibaba Cloud account. To obtain the ID, follow these steps: Log on to the DataWorks console , move the pointer over the profile picture in the upper-right corner, and then click Security Settings . On the Security Settings page that appears, view the value of Account ID , which is the ID of your Alibaba Cloud account.
Database Name	The name of the ApsaraDB for OceanBase database.
Username	The username for logging on to the database.

Parameter	Description
Password	The password for logging on to the database.

- The following table describes the parameters that you need to set if you select **Connection Mode** for **Connect To**.

Add ApsaraDB for OceanBase Connection

* Connect To : ☐ ApsaraDB for Oceanbase ☒ Connection Mode

* Connection Name : Enter a name.

Description :

* Environment : ☒ Development ☐ Production

* JDBC URL :

* 用户名 :

* 密码 :

Resource Connectivity :


Resource Group Name	Type	Status	Test Time	Actions
Public Resource Group	Not Supported ?	Not Supported		Not Supported

Attention : If the test fails, the possible reason is:

- The database is not started, please confirm that it has started normally.
- DataWorks cannot access the network where the database is located. Please make sure the network is connected with Aliyun.
- DataWorks is prohibited by the network firewall where the database is located. Please add [whitelist](#).
- The database domain name cannot be resolved correctly. Please confirm that the domain name can be accessed by normal resolution.

Previous

Complete

Parameter	Description
Connect To	The type of the connection. In this example, select Connection Mode .
Connection Name	The name of the connection. The name can contain letters , digits, and underscores (_) and must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Environment	The environment in which the connection is used. Valid values: Development and Production . <div>  Note: This parameter is available only when the workspace is in the standard mode. </div>

Parameter	Description
JDBC URL	The Java Database Connectivity (JDBC) URL of the ApsaraDB for OceanBase database, in the format jdbc:oceanbase://ip:port/database.
Username	The username for logging on to the database.
Password	The password for logging on to the database.

5. Click **Test Connection** in the Actions column of each resource group.

A sync node only uses one resource group. Therefore, you must test the connectivity of all the resource groups for Data Integration that your sync nodes use to connect to the data store so that sync nodes can run properly. For more information, see [Test data store connectivity](#).

6. After the connection passes the connectivity test, click **Complete**.

What's next

Now you have learned how to configure an ApsaraDB for OceanBase connection. You can proceed with the next tutorial to configure ApsaraDB for OceanBase Reader and Writer. For more information, see [Configure ApsaraDB for OceanBase Reader](#) and [Configure ApsaraDB for OceanBase Writer](#).

8.1.25 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 codeless user interface (UI) or code editor.

Context

Workspaces in the standard mode support the connection isolation feature. You can add connections for the development and production environments separately and isolate the connections to protect your data security. For more information, see [Connection isolation](#).

Procedure

1. Go to the **Data Source** page.
 - a) Log on to the [DataWorks console](#).
 - b) In the left-side navigation pane, click **Workspaces**.
 - c) In the top navigation bar, select the region where the target workspace resides. Find the target workspace and click **Data Integration** in the Actions column.
 - d) On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.
2. On the **Data Source** page, click **Add a Connection** in the upper-right corner.
3. In the **Add Connection** dialog box that appears, click **Vertica** in the Big Data Storage section.
4. In the **Add Vertica Connection** dialog box that appears, set the parameters as required.

Add Vertica Connection ×

* Connection Name :

Description :

* Environment : ☒ Development ☐ Production

* JDBC URL :

* Username :

* Password :

Resource Connectivity :

Resource Group Name	Type	Status	Test Time	Actions
Public Resource Group		Not Tested		Test Connection...


Attention : If the test fails, the possible reason is:

1. The database is not started, please confirm that it has started normally.
2. DataWorks cannot access the network where the database is located. Please make sure the network is connected with Aliyun.
3. DataWorks is prohibited by the network firewall where the database is located. Please add [whitelist](#).
4. The database domain name cannot be resolved correctly. Please confirm that the domain name can be accessed by normal resolution.

Previous

Complete

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

Parameter	Description
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 the standard mode.
JDBC URL	The Java Database Connectivity (JDBC) URL of the Vertica database, in the format <code>jdbc:vertica://Server IP:Port/Database</code> .
Username	The username for logging on to the database.
Password	The password for logging on to the database.

5. Click **Test Connection** in the Actions column of each resource group.

A sync node only uses one resource group. Therefore, you must test the connectivity of all the resource groups for Data Integration that your sync nodes use to connect to the data store so that sync nodes can run properly. For more information, see [Test data store connectivity](#).

6. After the connection passes the connectivity test, click **Complete**.

What's next

Now you have learned how to configure a Vertica connection. You can proceed with the next tutorial to configure Vertica Reader and Writer. For more information, see [Configure Vertica Reader](#) and [Configure Vertica Writer](#).

8.2 Reader configuration

8.2.1 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 is applicable only when the MySQL engine is used. DRDS is a distributed MySQL database, and most of the communication protocols are the same as those used by MySQL.

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 sends 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

Parameter	Description	Required	Default value
column	<p>The columns to be synchronized from the source table. The columns are described in a JSON array. The default value is <code>[*]</code>, which indicates all columns.</p> <ul style="list-style-type: none"> 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 DRDS, for example, <code>["id", "`table`", "1", "bazhen.csy", "null", "to_char(a + 1)", "2.3", "true"]</code>. <ul style="list-style-type: none"> 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
where	<p>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 <code>STRTODATE('\${bdp.system.bizdate}','%Y%m%d') <= today AND today < DATEADD(STRTODATE('\${bdp.system.bizdate}','%Y%m%d'), interval 1 day)</code>.</p> <ul style="list-style-type: none"> 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

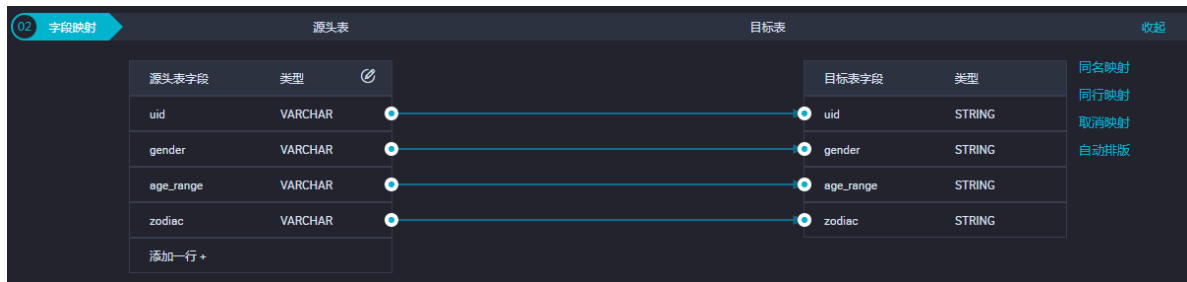
1. Configure the connections.

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	<p>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.</p> <p>If data sharding is performed based on the configured shard key, data can be read concurrently to improve data synchronization efficiency.</p> <div>  Note: The Shard Key parameter appears only when you configure the source connection for a sync node. </div>

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.
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: <ul style="list-style-type: none"> 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.

03 Channel

You can control the sync process by throttling the bandwidth or limiting the dirty data records allowed. [Learn more.](#)

* Expected Maximum ?
Concurrency

* Bandwidth Throttling ☒ Disable ☐ Enable ?

Dirty Data Records Allowed ? The node automatically ends when the number of dirty data records reaches XX.

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read 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. If a large number of nodes including this sync node are deployed on the default resource group, the sync node may need to wait for resources. We recommend that you purchase an exclusive resource group for data integration or add a custom resource group. For more information, see #unique_15 and Add a custom resource group .

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,
    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 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, 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.

8.2.2 Configure HBase Reader

HBase Reader allows you to read data from HBase.

HBase Reader connects to a remote HBase database through a Java client. Then, it uses the scan method to read data based on the specified rowkey range, converts the data to a dataset based on Data Integration data types, and sends the dataset to a writer.

Supported features

- Supports HBase 0.94.x, HBase 1.1.x, and HBase 2.x.
 - If you use HBase 0.94.x, set the plugin parameter to **094x**.

```
"reader": {  
  "plugin": "094x"
```

```
}
```

- If you use HBase 1.1.x or HBase2.x, set the plugin parameter to **11x**.

```
"reader": {
  "plugin": "11x"
}
```

**Note:**

Currently, HBase 1.1.x Reader is compatible with HBase 2.0 Reader. If you have any issues in using HBase Reader, submit a ticket.

- Supports the normal and multiVersionFixedColumn modes.
 - In **normal** mode, HBase Reader reads only the latest version of data from an HBase table and converts it to a wide table, that is, a binary table.

```
hbase(main):017:0> scan 'users'
ROW COLUMN+CELL
lisi column=address:city, timestamp=1457101972764, value=
beijing
lisi column=address:contry, timestamp=1457102773908, value=
china
lisi column=address:province, timestamp=1457101972736, value=
beijing
lisi column=info:age, timestamp=1457101972548, value=27
lisi column=info:birthday, timestamp=1457101972604, value=
1987-06-17
lisi column=info:company, timestamp=1457101972653, value=
baidu
xiaoming column=address:city, timestamp=1457082196082, value=
hangzhou
xiaoming column=address:contry, timestamp=1457082195729,
value=china
xiaoming column=address:province, timestamp=1457082195773,
value=zhejiang
xiaoming column=info:age, timestamp=1457082218735, value=29
xiaoming column=info:birthday, timestamp=1457082186830, value=
1987-06-17
xiaoming column=info:company, timestamp=1457082189826, value
=alibaba
2 row(s) in 0.0580 seconds }
```

HBase Reader converts the data read from HBase to the following table.

rowKey	address: city	address: contry	address: province	info: age	info: birthday	info: company
lisi	beijing	china	beijing	27	1987-06-17	baidu
xiaoming	hangzhou	china	zhejiang	29	1987-06-17	alibaba

- In **multiVersionFixedColumn** mode, HBase Reader reads data from an HBase table and converts it to a narrow table. Each data record consists of the four columns:

rowKey, **family:qualifier**, **timestamp**, and **value**. You need to specify the columns from which HBase Reader reads data, and HBase Reader converts each version of a table cell to a data record.

```
hbase(main):018:0> scan 'users',{VERSIONS=>5}
ROW                COLUMN+CELL
lisi               column=address:city, timestamp=1457101972764, value=
beijing
lisi               column=address:contry, timestamp=1457102773908, value=
china
lisi               column=address:province, timestamp=1457101972736, value=
beijing
lisi               column=info:age, timestamp=1457101972548, value=27
lisi               column=info:birthday, timestamp=1457101972604, value=
1987-06-17
lisi               column=info:company, timestamp=1457101972653, value=
baidu
xiaoming           column=address:city, timestamp=1457082196082, value=
hangzhou
xiaoming           column=address:contry, timestamp=1457082195729,
value=china
xiaoming           column=address:province, timestamp=1457082195773,
value=zhejiang
xiaoming           column=info:age, timestamp=1457082218735, value=29
xiaoming           column=info:age, timestamp=1457082178630, value=24
xiaoming           column=info:birthday, timestamp=1457082186830, value=
1987-06-17
xiaoming           column=info:company, timestamp=1457082189826, value
=alibaba
2 row(s) in 0.0260 seconds }
```

HBase Reader converts the data read from HBase to the following table.

rowKey	column:qualifier	timestamp	value
lisi	address:city	1457101972764	beijing
lisi	address:contry	1457102773908	china
lisi	address:province	1457101972736	beijing
lisi	info:age	1457101972548	27
lisi	info:birthday	1457101972604	1987-06-17
lisi	info:company	1457101972653	beijing
xiaoming	address:city	1457082196082	hangzhou
xiaoming	address:contry	1457082195729	china
xiaoming	address:province	1457082195773	zhejiang
xiaoming	info:age	1457082218735	29
xiaoming	info:age	1457082178630	24
xiaoming	info:birthday	1457082186830	1987-06-17


rowKey	column:qualifier	timestamp	value
xiaoming	info:company	1457082189826	alibaba


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
Date and time	DATE	DATE
Byte	BYTES	BYTES
Boolean	BOOLEAN	BOOLEAN

Parameters

Parameter	Description	Required	Default value
haveKerberos	<p>Specifies whether Kerberos authentication is required. A value of true indicates that Kerberos authentication is required.</p> <div>  Note: <ul style="list-style-type: none"> If the value is true, the following five Kerberos-related parameters must be specified: <ul style="list-style-type: none"> 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. </div>	No	false

Parameter	Description	Required	Default value
hbaseConfig	<p>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.</p> <div>  Note: You must use the internal endpoint to access an ApsaraDB for HBase database. </div>	Yes	None
mode	The mode in which data is read from HBase. 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 to strings. Valid values: UTF-8 and GBK.	No	UTF-8

Parameter	Description	Required	Default value
column	<p>The HBase columns from which data is read. This parameter is required in both normal and multiVersionFixedColumn modes.</p> <ul style="list-style-type: none"> In normal mode: <p>The name parameter specifies the name of the column in the HBase table. The format must be columnFamily:columnName except for the rowkey.</p> <p>The type parameter specifies the source data type.</p> <p>The format parameter specifies the date format.</p> <p>The value parameter specifies the column value if the column is a constant column. An example is provided as follows:</p> <pre>"column": [{ "name": "rowkey", "type": "string" }, { "value": "test", "type": "string" }]</pre> <p>For the column parameter, you must specify the type parameter and specify one of the name and value parameters.</p> In multiVersionFixedColumn mode: <p>The name parameter specifies the name of the column in the HBase table. The format must be columnFamily:columnName except for the rowkey.</p> <p>The type parameter specifies the source data type. The format parameter specifies the date format. You cannot create constant columns in multiVersionFixedColumn mode. An example is provided as follows:</p> <pre>"column": [{ "name": "rowkey", "type": "string" }, { "name": "info:age"</pre> 	Yes	None

Parameter	Description	Required	Default value
range	<p>The rowkey range that HBase Reader reads.</p> <ul style="list-style-type: none"> 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. An example is provided as follows: <pre>"range": { "startRowkey": "aaa", "endRowkey": "ccc", "isBinaryRowkey": false }</pre>	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 HBase in normal mode.

```
{
  "type": "job",
  "version": "2.0", // The version number.
  "steps": [
    {
      "stepType": "hbase", // The reader type.
      "parameter": {
        "mode": "normal", // The mode in which data is read from the HBase database.
        Valid values: normal and multiVersionFixedColumn.
        "scanCacheSize": 256, // The number of rows read by an HBase client with each
        RPC connection.
        "scanBatchSize": 100, // The number of columns read by an HBase client with
        each RPC connection.
        "hbaseVersion": "094x/11x", // The HBase version.
        "column": [ // The columns to be synchronized.
          {
            "name": "rowkey", // The rowkey name.
            "type": "string" // The data type.
          },
          {
            "name": "columnFamilyName1:columnName1",
            "type": "string"
          }
        ]
      }
    }
  ]
}
```

```

    },
    {
      "name": "columnFamilyName2:columnName2",
      "format": "yyyy-MM-dd",
      "type": "date"
    },
    {
      "name": "columnFamilyName3:columnName3",
      "type": "long"
    }
  ],
  "range": { // The rowkey range that HBase Reader reads.
    "endRowkey": "", // The end rowkey.
    "isBinaryRowkey": true // The method used to convert the specified start and
    end rowkeys to the byte[] format. Default value: false.
    "startRowkey": "" // The start rowkey.
  },
  "maxVersion": "", // The number of versions read by HBase Reader when multiple
  versions are available.
  "encoding": "UTF-8", // The encoding format.
  "table": "", // The name of the table to be synchronized.
  "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": "Reader",
  "category": "reader"
},
{ // The following template is used to configure Stream Writer. For more information,
  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"
      }
    ]
  }
}

```

```
}
```

8.2.3 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 to a format that is readable by Data Integration, and then sends the converted data to a writer.

HDFS Reader reads data from files in an HDFS and converts the data to a format that is readable by Data Integration.

Example:

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.

Supported 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, BZ2, 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

Hive maintains the metadata of HDFS files and stores the metadata in its own metadatabase, such as a MySQL database. Currently, HDFS Reader cannot access or query the metadata in the Hive metadatabase. Therefore, you must specify the data types for them to be converted to those readable to Data Integration.

The following table lists the default mapping between data types in RCFile, ParquetFile, ORCFile, TextFile, and SequenceFile files in Hive and the data types supported by Data Integration.

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

The data types are described as follows:



- LONG: data of the integer type in HDFS files, such as 123456789.
- DOUBLE: data of the floating-point type in HDFS files, such as 3.1415.


- **BOOLEAN:** data of the Boolean type in HDFS files, such as true or false. The value is case -insensitive.
- **DATE:** data of the date and time type in HDFS files, such as 2014-12-31 00:00:00.


**Note:**



The data type **TIMESTAMP** supported by Hive can be accurate to nanoseconds, so the **TIMESTAMP** data stored in **TextFile** and **ORCFile** files is similar to 2015-08-21 22:40:47.397898389. After the data of the **TIMESTAMP** type in Hive is converted to data of the **DATE** type in Data Integration, the nanoseconds in the data will be lost. Therefore, you must specify the type of converted date to **STRING** to make sure that the nanoseconds are retained after conversion.

Parameters


Parameter	Description	Required	Default value
path	<p>The path of the file to read. To read multiple files, use a regular expression such as /hadoop/data_201704*.</p> <ul style="list-style-type: none"> 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. <div>  Note: The actual number of threads is determined by both the number of HDFS files to be read and the specified transmission rate. </div> <ul style="list-style-type: none"> 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. <div>  Note: <ul style="list-style-type: none"> 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. 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 HDFS. <p>Therefore, if you want HDFS Reader to read the data of a partition, specify the file path of the partition. For example, if you want 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:</p> </div>	Yes	None
180			Issue: 20200628

Parameter	Description	Required	Default value
fileType	<p>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.</p> <p>The valid values of the fileType parameter are described as follows:</p> <ul style="list-style-type: none"> 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. <div>  Note: <p>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 to 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:</p> <ul style="list-style-type: none"> HDFS Reader converts MAP-type ORCFile data to a string: {job=80, team=60, person=70}. HDFS Reader converts MAP-type TextFile data to a string: {job:80, team:60, person:70}. <p>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</p> </div>	Yes	None

Parameter	Description	Required	Default value
column	<p>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": ["*"].</p> <p>You can also specify the column parameter (either index or value) in the following way:</p> <pre>{ "type": "long", "index": 0 // The first INT-type column of the source file. The index starts from 0. }, { "type": "string", "value": "alibaba" // The value of the current column, that is, a constant "alibaba". }</pre> <div>  Note: We recommend that you set this parameter by specifying the type and index parameters, rather than using "column": ["*"]. </div>	Yes	None
fieldDelimiter	<p>The column delimiter. You need to specify the column delimiter for text files, and the default delimiter is a comma (,). You do not need to specify the column delimiter for ORC files, and the default delimiter is \u0001.</p> <ul style="list-style-type: none"> If you want each row to be converted to 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

Parameter	Description	Required	Default value
nullFormat	<p>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.</p> <p>For example, if you specify nullFormat="null", Data Integration considers null as a null pointer.</p> <div>  Note: Pay attention to the difference between the string NULL and null pointer. </div>	No	None
compress	<p>The compression format. Available compression formats for CSV files are GZIP, BZ2, ZIP, LZO, LZO_DEFLATE, HADOOP-SNAPPY, and FRAMING-SNAPPY.</p> <div>  Note: <ul style="list-style-type: none"> 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. </div>	No	None

Parameter	Description	Required	Default value
parquetSchema	<p>The description of the data schema in Parquet files. This parameter is required when you set fileType to parquet. Make sure that the value of the parquetSchema parameter complies with the JSON syntax. The parquetSchema parameter contains the following fields:</p> <pre>message messageType { required, dataType, columnName; ; }</pre> <ul style="list-style-type: none"> messageTypeName: the name of the MessageType object. required: specifies whether the field is required. We recommend that you set all the fields to optional. dataType: the type of the field. Valid values: BOOLEAN, INT32, INT64, INT96, FLOAT, DOUBLE, BINARY, and FIXED_LEN_BYTE_ARRAY. Set this field to BINARY if the file stores strings. Note that each line, including the last one, must end with a semicolon (;). <p>An example is provided as follows:</p> <pre>"parquetSchema": "message m { optional int32 minute_id; optional int32 dsp_id; optional int32 adx_pid; optional int64 req; optional int64 res; optional int64 suc; optional int64 imp; optional double revenue; }"</pre>	No	None
csvReaderConfig	<p>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.</p> <p>An example is provided as follows:</p> <pre>"csvReaderConfig": { "safetySwitch": false, "skipEmptyRecords": false, "useTextQualifier": false }</pre>	No	None
184	You can use the following parameters and their default values:		Issue: 20200628

Parameter	Description	Required	Default value
kerberosKeytabFilePath	The absolute path of the keytab file for Kerberos authentication. This parameter is required if the haveKerberos parameter is set to true.	No	None
kerberosPrincipal	<p>The Kerberos principal to which Kerberos can assign tickets, such as <code>****/hadoopclient@**. ***</code>. This parameter is required if the haveKerberos parameter is set to true.</p> <div>  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. An example is provided as follows: <pre>"haveKerberos":true, "kerberosKeytabFilePath":"/opt/datax/**.keytab", "kerberosPrincipal":"/hadoopclient@**. ***"</pre> </div>	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 time format.
            "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 Stream Writer. For more information,
see 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
        parameter to true.
        "dmu": 1// The DMU value.
    }
},
"order": {
    "hops": [
        {
            "from": "Reader",
            "to": "Writer"
        }
    ]
}
}
}

```

The following is an example of the HDFS Reader configuration with the `parquetSchema` parameter.



Note:

- The `fileType` parameter must be set to `parquet`.

- If you want HDFS Reader to read specific columns from a Parquet file, you must specify the `parquetSchema` parameter and specify the columns to be synchronized through the `index` field in the `column` parameter.

```
"reader": {
  "name": "hdfsreader",
  "parameter": {
    "path": "/user/hive/warehouse/addata.db/dw_ads_rtb_monitor_minute/thedate=
20170103/hour_id=22/*",
    "defaultFS": "h10s010.07100.149:8020",
    "column": [
      {
        "index": 0,
        "type": "string"
      },
      {
        "index": 1,
        "type": "long"
      },
      {
        "index": 2,
        "type": "double"
      }
    ],
    "fileType": "parquet",
    "encoding": "UTF-8",
    "parquetSchema": "message m { optional int32 minute_id; optional int32 dsp_id
; optional int32 adx_pid; optional int64 req; optional int64 res; optional int64 suc;
optional int64 imp; optional double revenue; }"
  }
}
```

8.2.4 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. With 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 only supports access to a MongoDB database by using a MongoDB database account. When adding 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 according to 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
Byte	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	odps_column3
a	b	{x_1,x_2}
a	b	{x_1,x_2,x_3}
a	b	{x_5}

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
collectionName	The name of the MongoDB collection.	Yes	None
column	The columns in MongoDB. <ul style="list-style-type: none"> 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

Parameter	Description	Required	Default value
query	The filter condition for obtaining data from MongoDB. Only data of 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 conditional operators (\$gt, \$lt, \$gte, \$lte), logical operators (and, or), and functions (max, min, sum, avg, ISODate) supported by MongoDB as needed. For more information, see Configure MongoDB Reader in the code editor.	No	None

Configure MongoDB Reader by using the codeless UI

Currently, the codeless user interface (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": [
    {
      "category": "reader",
      "name": "Reader",
      "parameter": {
        "datasource": "datasourceName", // The connection name.
        "collectionName": "tag_data", // The name of the MongoDB collection.
        "query": "", // The filter condition for obtaining data from MongoDB.
        "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": "mongodb"
},
{
  // The following template is used to configure Stream Writer. For more information,
  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"
      }
    ]
  }
}

```

**Note:**

Currently, you cannot retrieve data elements from arrays.

8.2.5 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 sends 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	<p>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.</p> <ul style="list-style-type: none"> 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"]. <ul style="list-style-type: none"> 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	<p>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.</p> <ul style="list-style-type: none"> 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	""

Parameter	Description	Required	Default value
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 <code>gmt_create>\$bizdate</code> . 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. 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 <code>select a,b from table_a join table_b on table_a.id = table_b.id</code> . 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 splitPk
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"
  },
  { // The following template is used to configure Stream Writer. For more information,
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"
    }
  ]
}
}
}

```

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, 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 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 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, 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.

8.2.6 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 sends 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:


- Except for the preceding data types, other types are not supported.
- MySQL Reader considers `tinyint(1)` as the INTEGER type.

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	<p>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.</p> <ul style="list-style-type: none"> 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"] . <ul style="list-style-type: none"> 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: <ul style="list-style-type: none"> ■ " " 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	<p>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.</p> <ul style="list-style-type: none"> 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	<p>The WHERE clause. For example, set this parameter to <code>gmt_create>\$bizdate</code>.</p> <ul style="list-style-type: none"> 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

Parameter	Description	Required	Default value
querySql (only available in the code editor)	<p>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 <code>select a,b from table_a join table_b on table_a.id = table_b.id</code>. 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.</p> <div>  Note: The querySql parameter is case-sensitive. For example, if you enter querysql, it does not take effect. </div>	No	None
singleOrMulti (only applicable to database and table sharding)	<p>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: <code>"singleOrMulti": "multi"</code>. 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.</p>	Yes	multi

Configure MySQL Reader by using the codeless UI

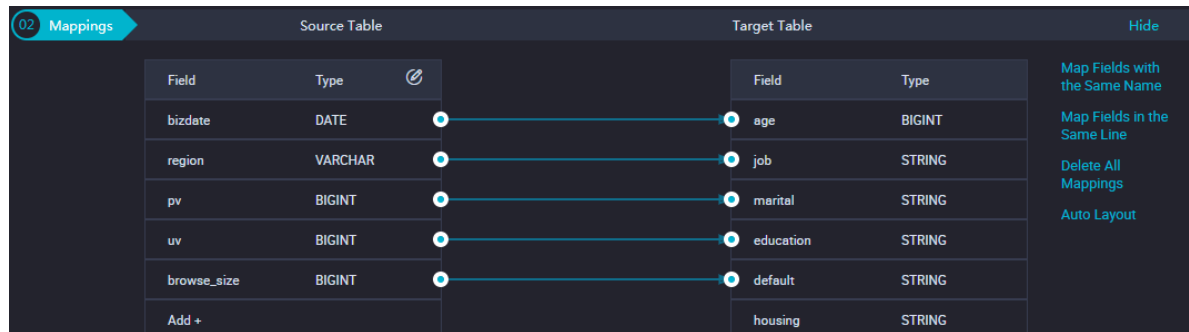
1. Configure the connections.

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	<p>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.</p> <p>If data sharding is performed based on the configured shard key, data can be read concurrently to improve data synchronization efficiency.</p> <div>  Note: The Shard Key parameter appears only when you configure the source connection for a sync node. </div>

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.
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	<ul style="list-style-type: none"> 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 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. If a large number of nodes including this sync node are deployed on the default resource group, the sync node may need to wait for resources. We recommend that you purchase an exclusive resource group for data integration or add a custom resource group. For more information, see #unique_15 and Add a custom resource group .

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": [// The name of the table to be synchronized. The table name must
be enclosed in brackets ([ ]), even if only one table exists.
            "xxx"
          ]
        }
      ],
      "where": "", // The WHERE clause.
      "splitPk": "", // The shard key.
      "encoding": "UTF-8" // The encoding format.
    },
    "name": "Reader",
    "category": "reader"
  },
  { // The following template is used to configure Stream Writer. For more information,
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"
      }
    ]
  }
}

```

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": {
  }
}

```

8.2.7 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.



Note:

Currently, Relational Database Service (RDS) and Distributed Relational Database Service (DRDS) do not support the Oracle storage engine.

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 sends 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	<p>The columns to be synchronized from the source table. The columns are described in a JSON array. The default value is <code>[*]</code>, which indicates all columns.</p> <ul style="list-style-type: none"> Column pruning is supported. 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 JSON format. <pre>["id", "1", "'mingya.wmy'", "null", "to_char(a + 1)", "2.3", "true"]</pre> <ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> The column parameter must be specified. 	Yes	None
splitPk	<p>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.</p> <ul style="list-style-type: none"> 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

Parameter	Description	Required	Default value
where	<p>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().</p> <ul style="list-style-type: none"> 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 (only available in the code editor)	<p>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 <code>select a,b from table_a join table_b on table_a.id = table_b.id</code>. If you specify the querySql parameter, Oracle Reader ignores the table, column, and where parameters that you have configured.</p>	No	None
fetchSize	<p>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.</p> <div>  Note: A value greater than 2048 may lead to out of memory (OOM) during the data synchronization process. </div>	No	1024

Configure Oracle Reader by using the codeless UI

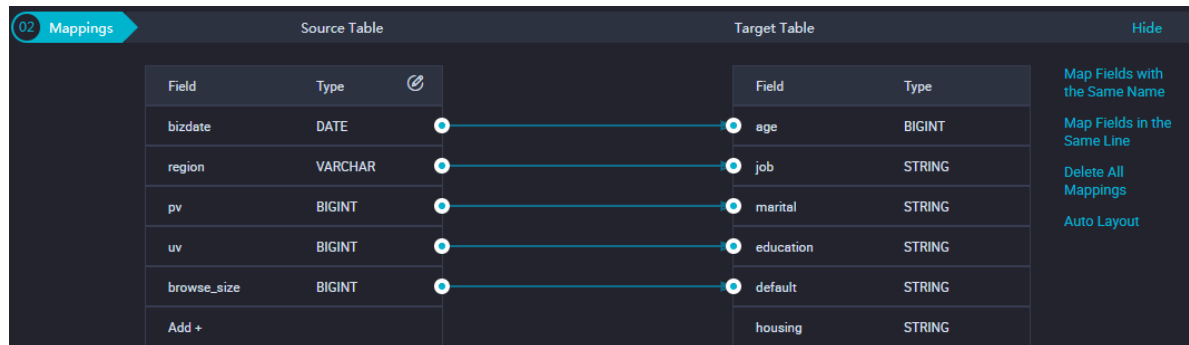
1. Configure the connections.

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	<p>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.</p> <p>If data sharding is performed based on the configured shard key, data can be read concurrently to improve data synchronization efficiency.</p> <div>  Note: The Shard Key parameter appears only when you configure the source connection for a sync node. </div>

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.
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	<ul style="list-style-type: none"> 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.

03 Channel

You can control the sync process by throttling the bandwidth or limiting the dirty data records allowed. [Learn more.](#)

* Expected Maximum ?
Concurrency

* Bandwidth Throttling ☒ Disable ☐ Enable ?

Dirty Data Records Allowed ? The node automatically ends when the number of dirty data records reaches XX.

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read 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. If a large number of nodes including this sync node are deployed on the default resource group, the sync node may need to wait for resources. We recommend that you purchase an exclusive resource group for data integration or add a custom resource group. For more information, see #unique_15 and Add a custom resource group .

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,
  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 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, 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.

8.2.8 Configure OSS Reader

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

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

- For more information about OSS, see [OSS overview](#).
- For more information about the OSS Java SDK, see [Alibaba Cloud OSS Java SDK](#).
- For more information about how to process unstructured data such as OSS data, see [Process unstructured data](#).

OSS stores unstructured data only. Currently, 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. 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.

Currently, 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, DATETIME, and BOOLEAN.



Data types

Category	Data Integration data type	OSS data type
Integer	LONG	LONG
String	STRING	STRING
Floating point	DOUBLE	DOUBLE
Boolean	BOOLEAN	BOOLEAN
Date and time	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

Parameter	Description	Required	Default value
Object	<p>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.</p> <ul style="list-style-type: none"> 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. <div>  Note: <ul style="list-style-type: none"> 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 too many objects, an OOM error may be returned. In this case, store the objects in different directories and then synchronize data. </div>	Yes	None

Parameter	Description	Required	Default value
column	<p>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.</p> <p>By default, OSS Reader reads all data as strings. You can specify the column parameter in the following way:</p> <pre>json "column": ["*"]</pre> <p>You can also specify the column parameter in the following way:</p> <pre>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, that is, a constant "alibaba". }</pre> <div>  Note: For the column parameter, you must specify the type parameter and specify one of the index and value parameters. </div>	Yes	By default, OSS Reader reads all data as strings.
fieldDelimiter	<p>The column delimiter.</p> <div>  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. </div>	Yes	,
compress	<p>The compression format of the object. By default, this parameter is left empty, that is, objects are not compressed. OSS Reader supports the following object compression formats: GZIP, BZIP2, and ZIP.</p>	No	By default, objects are not compressed.

Parameter	Description	Required	Default value
encoding	The encoding format of the object to read.	No	UTF-8
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 source and destination connections for the sync node.

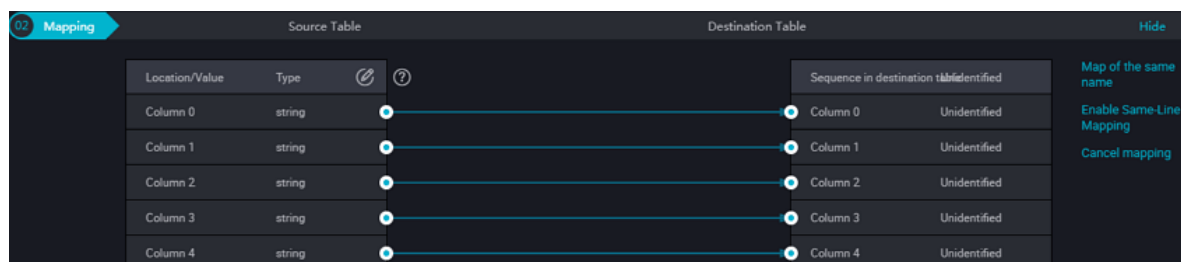
The screenshot shows the 'Data Source' configuration window. It has two main sections: 'Source' and 'Destination'. The 'Source' section is currently selected and contains the following fields: 'Data Source' (set to OSS), 'Object Prefix' (input field), 'File Type' (set to csv), 'Column Separator' (input field), 'Encoding' (set to UTF-8), 'Null String' (input field), 'Compression' (set to None), and 'Include Header' (set to No). The 'Destination' section contains: 'Data Source' (set to OSS), 'Object Prefix' (input field), 'File Type' (set to csv), 'Column Separator' (input field), 'Encoding' (set to UTF-8), 'Null String' (input field), 'Time Format' (input field), and 'Solution to Duplicate' (set to Replace the Original File). A 'Preview' button is located at the bottom of the 'Source' section.

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.

Parameter	Description
Object Name Prefix	<p>The object parameter in the preceding parameter description.</p> <div>  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. </div>
Field Delimiter	The fieldDelimiter parameter in the preceding parameter description. The default delimiter is comma (,).
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. 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. By default, objects 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.

Parameter	Description
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.

03 Channel

You can control the sync process by throttling the bandwidth or limiting the dirty data records allowed. [Learn more.](#)

* Expected Maximum ?
Concurrency

* Bandwidth Throttling ☒ Disable ☐ Enable ?

Dirty Data Records Allowed ? The node automatically ends when the number of dirty data records reaches XX.

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read 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. If a large number of nodes including this sync node are deployed on the default resource group, the sync node may need to wait for resources. We recommend that you purchase an exclusive resource group for data integration or add a custom resource group. For more information, see #unique_15 and Add a custom resource group .

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 CSV-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"
    },
    { // The following template is used to configure Stream Writer. For more information,
see the corresponding topic.
      "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 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"
    }
  ]
}
}
}

```

Read ORC or Parquet files from OSS

Currently, OSS Reader uses HDFS Reader to read ORC or Parquet files from OSS. In addition to the original parameters, OSS Reader provides extended parameters **Path** and **FileFormat**. For more information about the extended parameters, see [Configure HDFS Reader](#).

- In the following code, OSS Reader is configured to read ORC files from OSS.

```

{
  "stepType": "oss",
  "parameter": {
    "datasource": "",
    "fileFormat": "orc",
    "path": "/tests/case61/orc__691b6815_9260_4037_9899_aa8e61dc7e4b",
    "column": [
      {
        "index": 0,
        "type": "long"
      },
      {
        "index": "1",
        "type": "string"
      },
      {
        "index": "2",
        "type": "string"
      }
    ]
  }
}

```

- In the following code, OSS Reader is configured to read Parquet files from OSS.

```

{
  "stepType": "oss",
  "parameter": {
    "datasource": "",
    "fileFormat": "parquet",
    "path": "/tests/case61/parquet",
    "parquetSchema": "message test { required int64 int64_col;\n required binary
str_col (UTF8);\nrequired group params (MAP) {\nrepeated group key_value {\n

```

```

nrequired binary key (UTF8);\nrequired binary value (UTF8);\n}\n\nrequired group
params_arr (LIST) {\n repeated group list {\n required binary element (UTF8);\n }
\n}\nrequired group params_struct {\n required int64 id;\n required binary name (
UTF8);\n }\nrequired group params_arr_complex (LIST) {\n repeated group list {\n
required group element {\n required int64 id;\n required binary name (UTF8);\n }\n
\n}\nrequired group params_complex (MAP) {\nrepeated group key_value {\nrequired
binary key (UTF8);\nrequired group value {\n required int64 id;\n required binary
name (UTF8);\n }\n}\n\n}\nrequired group params_struct_complex {\n required int64
id;\n required group detail {\n required int64 id;\n required binary name (UTF8);\n }
\n }\n}\n",
  "column": [
    {
      "index": 0,
      "type": "long"
    },
    {
      "index": "1",
      "type": "string"
    },
    {
      "index": "2",
      "type": "string"
    },
    {
      "index": "3",
      "type": "string"
    },
    {
      "index": "4",
      "type": "string"
    },
    {
      "index": "5",
      "type": "string"
    },
    {
      "index": "6",
      "type": "string"
    },
    {
      "index": "7",
      "type": "string"
    }
  ]
}
}

```

8.2.9 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 to 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

Parameter	Description	Required	Default value
path	<p>The path of the FTP file to read. You can specify multiple FTP file paths.</p> <ul style="list-style-type: none"> 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. FTP Reader can flexibly generate node names based on custom parameters. <div>  Note: <ul style="list-style-type: none"> 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. </div>	Yes	None

Parameter	Description	Required	Default value
column	<p>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.</p> <p>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:</p> <pre>{ "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". }</pre> <p>For the column parameter, you must specify the type parameter and specify one of the index and value parameters.</p>	Yes	By default, FTP Reader reads all data as strings.
fieldDelimiter	<p>The column delimiter.</p> <div>  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. </div>	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	<p>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.</p> <p>For example, if you specify nullFormat:"null", Data Integration considers null as a null pointer.</p>	No	None
markDoneFile	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	<p>The format of the file saved by FTP Reader. By default, FTP Reader converts the data to a two-dimensional table and stores the table in a CSV file. If you specify binary as the file format, Data Integration converts data to the binary format for replication and transmission.</p> <p>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).</p>	No	None

Configure FTP Reader by using the codeless UI

1. Configure the connections.

Configure the source and destination connections for the sync node.

The data sources can be default data sources or data sources created by you. Click [here](#) to check the supported data source types.

Source

- Data Source: FTP
- File Path: /home/workshop/user_log.txt
- File Type: text
- Column: 1
- Separator
- Encoding: UTF-8
- Null String: Enter the string that represents null
- Compression: None
- Format
- Include Header: No

Destination

- Data Source: ODPS
- Table: ods_raw_log_d
- Partition: dt = \${bizdate}
- Clearance Rule: Clear Existing Data Before Writing (Insert Overwrit...)
- Compression: Disable
- Consider Empty String as Null: Yes

Generate Destination Table

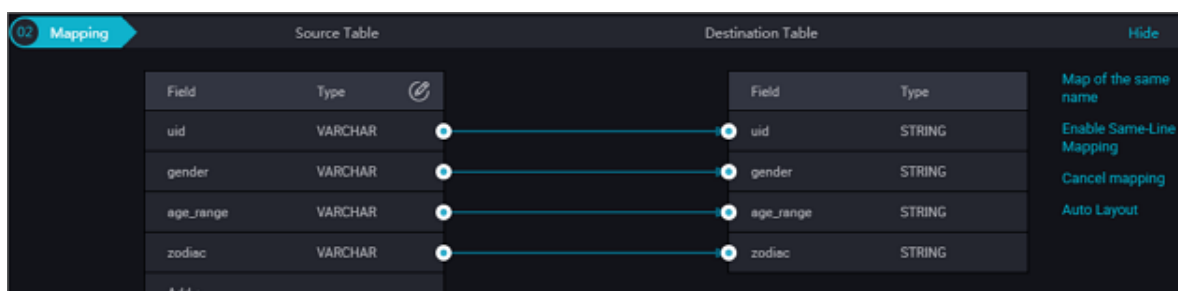
Preview

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. Default value: CSV.
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, which defines a string that represents the null value.
Compression Format	The compression format. By default, files are not compressed.

Parameter	Description
Include Header	The skipHeader parameter in the preceding parameter description. Default value: 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.
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.

03 Channel

You can control the sync process by throttling the bandwidth or limiting the dirty data records allowed. [Learn more.](#)

* Expected Maximum ?
Concurrency

* Bandwidth Throttling ☒ Disable ☐ Enable ?

Dirty Data Records Allowed ? The node automatically ends when the number of dirty data records reaches XX.

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read 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. If a large number of nodes including this sync node are deployed on the default resource group, the sync node may need to wait for resources. We recommend that you purchase an exclusive resource group for data integration or add a custom resource group. For more information, see #unique_15 and Add a custom resource group .

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 Stream Writer. For more information,
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"
        }
    ]
}
}

```

8.2.10 Configure Table Store Reader

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

Table Store Reader can read incremental data from Table Store based on the specified range. Currently, Table Store 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.

Table Store 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. Table

Store organizes data into instances and tables. Using data sharding and load balancing technologies, Table Store seamlessly expands the data scale.

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

Table Store Reader splits a sync node to concurrent tasks based on the table range to synchronize data in a Table Store table. Each thread is responsible for running a task.

Table Store Reader supports all Table Store data types. The following table lists the data types supported by Table Store Reader.

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

**Note:**

Table Store does not support data of the DATE type. Applications use the LONG-type UNIX timestamp to indicate the time.

Parameters

Parameter	Description	Required	Default value
endpoint	The endpoint of the Table Store server. For more information, see Endpoint .	Yes	None
accessId	The AccessKey ID for accessing Table Store.	Yes	None
accessKey	The AccessKey secret for accessing Table Store.	Yes	None

Parameter	Description	Required	Default value
instanceName	<p>The name of the Table Store instance. The instance is an entity for you to use and manage Table Store.</p> <p>After you activate the Table Store service, you must create an instance in the console before creating and managing tables.</p> <p>Instances are the basic unit for managing Table Store resources. All access control and resource measurement for applications are completed at the instance level.</p>	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 Table Store.	Yes	None

Parameter	Description	Required	Default value
column	<p>The columns to be synchronized from the source table. The columns are described in a JSON array. Table Store is a NoSQL database service. You must specify column names for Table Store Reader to read data.</p> <ul style="list-style-type: none"> You can specify common columns. For example, you can specify {"name":"col1"} for Table Store Reader to read data in column 1. You can specify certain columns to read. Table Store Reader only reads specified columns. You can specify constant columns. For example, you can specify {"type":"STRING", "value":"DataX"} to read 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 Table Store, and INF_MAX indicates the maximum value specified by Table Store. If you set the type to INF_MIN or INF_MAX, do not set the value. Otherwise, errors may occur. You cannot specify a function or custom expression, because Table Store does not provide functions or expressions similar to those of SQL. Table Store Reader cannot read columns that contain functions or expressions. 	Yes	None

Parameter	Description	Required	Default value
begin and end	<p>The Table Store table range from which data is to be read. 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 Table Store 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 certain data from a Table Store table with the primary key of [DeviceID, SellerID], specify the begin and end parameters in the following way:</p> <pre> "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.] } </pre> <p>To read all data from the table, specify the begin and end parameters in the following way:</p> <pre> "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.] } </pre>	Yes	None
split	<p>The custom rule for data sharding. This parameter is an advanced setting. We recommend that you do not set this parameter.</p> <p>If data is unevenly distributed in a Table Store table and the automatic sharding feature of Table Store Reader</p>	No	None
Issue: 20200628	<p>fails to work, you can customize a sharding rule.</p> <p>The sharding rule specified by the split parameter</p>		237

Configure Table Store Reader by using the code editor

In the following code, a node is configured to read data from a Table Store table.

```
{
  "type": "job",
  "version": "2.0", // The version number.
  "steps": [
    {
      "stepType": "ots", // The reader type.
      "parameter": {
        "datasource": "", // The connection name.
        "column": [ // The columns to be synchronized.
          {
            "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 table to be synchronized.
},
"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
    parameter to true.
    "concurrent": 1, // The maximum number of concurrent threads.
  }
},
"order": {
  "hops": [
    {
      "from": "Reader",
      "to": "Writer"
    }
  ]
}
}

```

8.2.11 Configure AnalyticDB for MySQL 3.0 Reader

This topic describes the data types and parameters supported by AnalyticDB for MySQL 3.0 Reader and how to configure it by using the codeless user interface (UI) and code editor.

AnalyticDB for MySQL 3.0 Reader allows you to read data from AnalyticDB for MySQL 3.0.

AnalyticDB for MySQL 3.0 Reader connects to a remote AnalyticDB for MySQL 3.0 database

through Java Database Connectivity (JDBC) and runs a SELECT statement to select and read data from the database.

Data types

The following table lists the data types supported by AnalyticDB for MySQL 3.0 Reader.

Category	AnalyticDB for MySQL 3.0 data type
Integer	INT, INTEGER, TINYINT, SMALLINT, and BIGINT
Floating point	FLOAT, DOUBLE, and DECIMAL
String	VARCHAR
Date and time	DATE, DATETIME, TIMESTAMP, and TIME
Boolean	BOOLEAN

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	<p>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.</p> <ul style="list-style-type: none"> 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 AnalyticDB for MySQL, for example, ["id", "table", "1", "bazhen.csy", "null", "to_char(a + 1)", "2.3", "true"]. <ul style="list-style-type: none"> id: a column name. table: the name of a column that contains reserved keywords. 	Yes	None

Parameter	Description	Required	Default value
splitPk	<p>The field used for data sharding when AnalyticDB for MySQL 3.0 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.</p> <ul style="list-style-type: none"> 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, AnalyticDB for 	No	None
	MySQL 3.0 Reader ignores the splitPk parameter and synchronizes		Issue: 20200628


Parameter	Description	Required	Default value
where	<p>The WHERE clause. For example, set this parameter to <code>gmt_create>\$bizdate</code>.</p> <ul style="list-style-type: none"> 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 AnalyticDB for MySQL on the SQL WHERE clause. 	No	None

Configure AnalyticDB for MySQL 3.0 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.

Parameter	Description
Shard Key	<p>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.</p> <p>If data sharding is performed based on the configured shard key, data can be read concurrently to improve data synchronization efficiency.</p> <div>  Note: The Shard Key parameter is displayed only when you configure the source connection for a sync node. </div>

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.
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.

Parameter	Description
Add	<ul style="list-style-type: none"> 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 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. If a large number of nodes including this sync node are deployed on the default resource group, the sync node may need to wait for resources. We recommend that you purchase an exclusive resource group for data integration or add a custom resource group. For more information, see #unique_15 and Add a custom resource group .

Configure AnalyticDB for MySQL 3.0 Reader by using the code editor

In the following code, a node is configured to read data from an AnalyticDB for MySQL 3.0 database. For more information about the parameters, see the preceding parameter description.

```
{
  "type": "job",
  "steps": [
    {
      "stepType": "analyticdb_for_mysql", // The reader type.
      "parameter": {
        "column": [ // The columns to be synchronized.
          "id",
          "value",
          "table"
        ],
        "connection": [
          {
            "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"
    },
    { // The following template is used to configure Stream Writer. For more information,
      // see the corresponding topic.
      "stepType": "stream",
      "parameter": {},
      "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": {
      "concurrent": 2, // 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
        // parameter to true.
    }
  }
}
```

```
}
```

8.2.12 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 sends 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
column	<p>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.</p> <ul style="list-style-type: none"> 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 SQL Server, for example, ["id", "table", "1", "mingya.wmy", "null", "to_char(a+1)", "2.3", "true"] . <ul style="list-style-type: none"> 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': 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

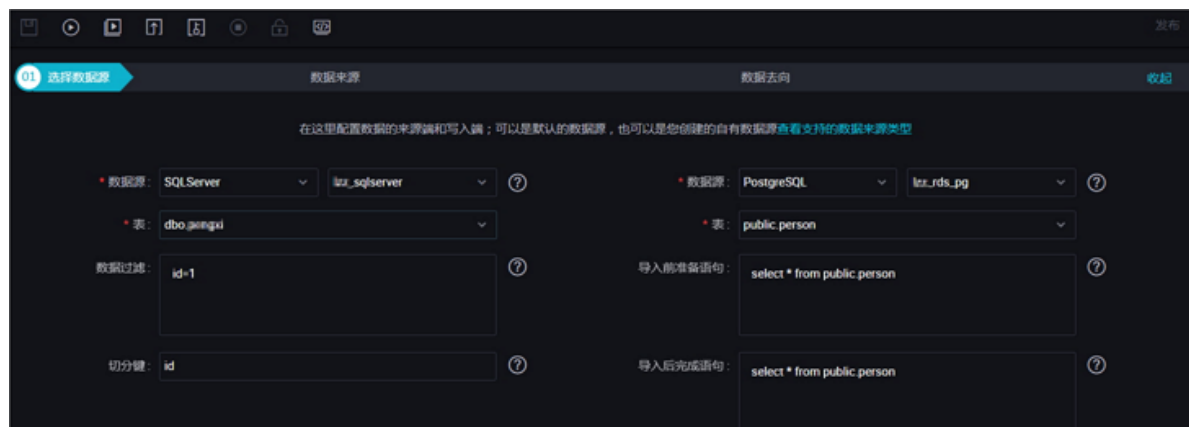
Parameter	Description	Required	Default value
splitPk	<p>The field used for data sharding when SQL Server Reader extracts data. If you specify the splitPk parameter, the table is sharded based on the shard key specified by this parameter. Data Integration then runs concurrent threads to synchronize data. This improves efficiency.</p> <ul style="list-style-type: none"> 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
where	<p>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 or <code>gmt_create > \$bizdate</code>.</p> <ul style="list-style-type: none"> 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	<p>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 <code>select a,b from table_a join table_b on table_a.id = table_b.id</code>. If you specify the querySql parameter, SQL Server Reader ignores the table, column, and where parameters that you have configured.</p>	No	None

Parameter	Description	Required	Default value
fetchSize	<p>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.</p> <div>  Note: A value larger than 2048 may lead to the out of memory (OOM) error during the data synchronization process. </div>	No	1024

Configure SQL Server Reader by using the codeless UI

1. Configure the connections.

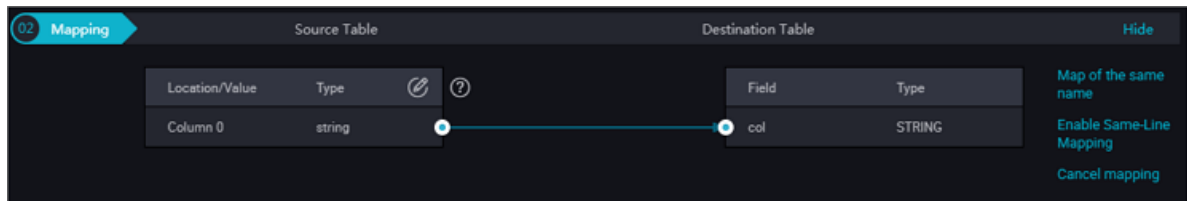
Configure the source and destination connections for the sync node.



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.

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.
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	<ul style="list-style-type: none"> 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 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. If a large number of nodes including this sync node are deployed on the default resource group, the sync node may need to wait for resources. We recommend that you purchase an exclusive resource group for data integration or add a custom resource group. For more information, see #unique_15 and Add a custom resource group .

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"
  },
  {
    "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"
      }
    ]
  }
}

```

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, SQL Server 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 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, 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.

8.2.13 Configure LogHub Reader

This topic describes the data types and parameters supported by LogHub Reader and how to configure it by using the codeless user interface (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 is readable 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 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, 2000 times/s.
- Read: 10 Mbit/s, 100 times/s.

LogHub Reader consumes log data in shards by following this process that uses the GetCursor and BatchGetLog API operations:

- 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 depending on the Alibaba Cloud region where the project resides and the project name. For more information about the endpoint of each region, see #unique_173 .	Yes	None
accessId	The AccessKey ID for accessing Log Service.	Yes	None
accessKey	The AccessKey secret for accessing 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

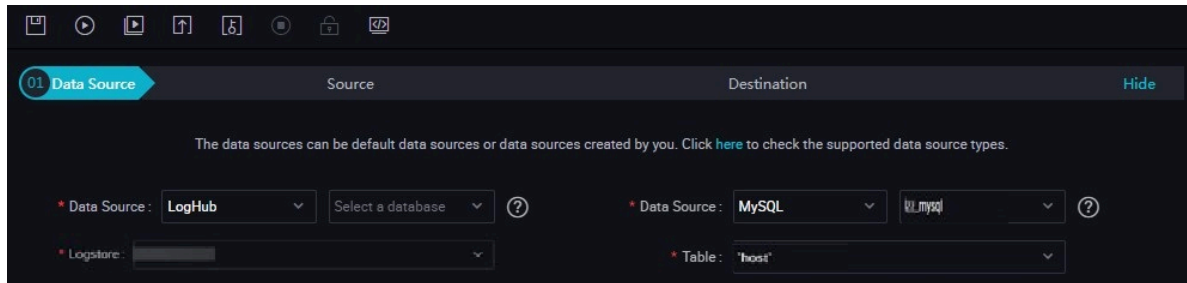
Parameter	Description	Required	Default value
column	<p>The name of the columns to be synchronized. You can set this parameter to the metadata in Log Service. In Log Service, the column names can be the log topic, unique identifier of the host, hostname, path, and log time.</p> <div>  Note: The column name is case-sensitive. For more information about column names in Log Service, see Log Service server group. </div>	Yes	None
beginDateTime	<p>The start time of data consumption, that is, the time when log data arrives at LogHub. This parameter defines the left boundary of an interval (left-closed and right-open) in the format of yyyyMMddHHmmss, for example, 20180111013000. The parameter can work with the scheduling time parameter in DataWorks.</p> <div>  Note: The beginDateTime and endDateTime parameters must be used in pairs. </div>	You must specify either beginDateTime or beginTimestampMillis, but not both.	None
endTime	<p>The end time of data consumption in the format of yyyyMMddHHmmss, such as 20180111013010. This parameter defines the right boundary of an interval (left-closed and right-open) and can work with the scheduling time parameter in DataWorks.</p> <div>  Note: Make sure that the intervals overlap. That is, the time specified by the endTime 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 not be pulled in some regions. </div>	You must specify either endTime or endTimeMillis, but not both.	None

Parameter	Description	Required	Default value
beginTimestampMillis	<p>The start time of data consumption. This parameter specifies the left boundary of the interval (left-closed and right-open), measured in milliseconds.</p> <div>  Note: The beginTimestampMillis and endTimestampMillis parameters must be used in pairs. The value -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. </div>	You must specify either beginTimestampMillis or beginDateTime, but not both.	None
endTimestampMillis	<p>The end time of data consumption, measured in milliseconds. This parameter defines the right boundary of the left-closed and right-open interval.</p> <div>  Note: The endTimestampMillis and beginTimestampMillis parameters must be used in pairs. The value -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. </div>	You must specify either endTimestampMillis or endDateTime, but not both.	None

Configure LogHub Reader by using the codeless UI

1. Configure the connections.

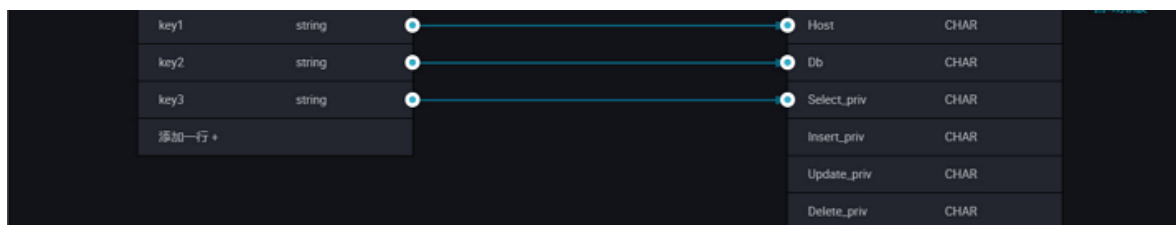
Configure the source and destination connections for the sync node.



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.
Logstore	The name of the target Logstore.
Start Timestamp	The start time of data consumption, that is, the time when log data arrives at LogHub. This parameter defines the left boundary of an interval (left-closed and right-open) 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 an interval (left-closed and right-open) 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, 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.
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.

3. Configure channel control policies.

03 Channel

You can control the sync process by throttling the bandwidth or limiting the dirty data records allowed. [Learn more.](#)

* Expected Maximum ?
Concurrency

* Bandwidth Throttling ☒ Disable ☐ Enable ?

Dirty Data Records Allowed ? The node automatically ends when the number of dirty data records reaches XX.

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read 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. If a large number of nodes including this sync node are deployed on the default resource group, the sync node may need to wait for resources. We recommend that you purchase an exclusive resource group for data integration or add a custom resource group. For more information, see #unique_15 and Add a custom resource group .

Configure LogHub Reader by using the code editor

In the following code, a node is configured to read data from a Logstore. 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.
    "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 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"
            }
        ]
    }
}

```

**Note:**

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

8.2.14 Configure Table Store Reader-Internal

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

Table Store 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. Table Store organizes data into instances and tables. Using data sharding and load balancing technologies, Table Store seamlessly expands the data scale.

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

The Table Store Internal model supports the multi-version mode and normal mode. Table Store Reader-Internal can export data in the two modes:

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

Table Store 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 {PrimaryKeyyy, 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 HBase Reader](#).

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


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


Data Integration data type	Table Store data type
LONG	INTEGER
DOUBLE	DOUBLE
STRING	STRING
BOOLEAN	BOOLEAN
BYTES	BINARY

Parameters

Parameter	Description	Required	Default value
mode	The mode in which Table Store Reader-Internal reads data. Valid values: normal and multiVersion.	Yes	None
endpoint	The endpoint of the Table Store server.	Yes	None
accessId	The AccessKey ID for accessing Table Store.	Yes	None
accessKey	The AccessKey secret for accessing Table Store.	Yes	None
instanceName	<p>The name of the Table Store instance. The instance is an entity for you to use and manage Table Store.</p> <p>After you activate the Table Store service, you must create an instance in the console before creating and managing tables. Instances are the basic unit for managing Table Store resources. All access control and resource measurement for applications are completed at the instance level.</p>	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 Table Store.	Yes	None
range	<p>The range of the data to export, in the format of [begin,end).</p> <ul style="list-style-type: none"> 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 equal to 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"}	<p>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.</p> <p>This parameter specifies the value range of the Table Store primary key and is used for data filtering. If you do not specify this parameter, the minimum value is used by default.</p> <p>The JSON format does not support binary data. If the value type in 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. A Java example is described as follows:</p> <ul style="list-style-type: none"> 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. <p>After you run the preceding code, the string "aGVsbG8=" is returned for the inputValue parameter.</p> <p>Finally, set this parameter to {"type":"binary","value" : "aGVsbG8="}.</p>	No	Data is read from the beginning of the table.

Parameter	Description	Required	Default value
range:{"end"}	<p>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.</p> <p>The JSON format does not support binary data. If the value type in 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. A Java example is described as follows:</p> <ul style="list-style-type: none"> 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. <p>After you run the preceding code, the string "aGVsbG8=" is returned for the inputValue parameter.</p> <p>Finally, set this parameter to {"type":"binary", "value":"aGVsbG8="}.</p>	No	Data is read until the end of the table.
range:{"split"}	<p>If an excessively large amount of data needs to be exported, you can specify this parameter to split one node to multiple concurrent threads.</p> <div>  Note: <ul style="list-style-type: none"> The value for the split parameter must be the shard key, which is the first column of the primary key, and the value type must be the same as that of the partition key. The specified value must fall within the value range of the begin and end parameters. The values for the split parameter 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. </div>	No	No sharding rule is specified by default.

Parameter	Description	Required	Default value
column	<p>The columns to be exported. Both regular and constant columns can be exported.</p> <p>Mode: The multi-version mode is supported.</p> <p>Normal column format: {"name": "{your column name}"}</p>	Yes	None
timeRange (only applicable to the multi-version mode)	<p>The time range of the requested data, in the format of [begin,end).</p> <div>  Note: The value of the begin parameter must be smaller than that of the end parameter. </div>	No	The data of all versions is read by default.
timeRange:{"begin"} (only applicable to the multi-version mode)	The start time for reading data. Valid values: 0 to LONG_MAX.	No	0
timeRange:{"end"} (only applicable to the multi-version mode)	The end time for reading data. Valid values: 0 to LONG_MAX.	No	LONG_MAX (9223372036 854775806L)
maxVersion (only applicable 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 by default.

Configure Table Store Reader-Internal by using the codeless UI

Currently, the codeless user interface (UI) is not supported for Table Store Reader-Internal.

Configure Table Store 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": {}
}

```

8.2.15 Configure RDBMS Reader

This topic describes the data types and parameters supported by relational database management system (RDBMS) Reader and how to configure it by using the code editor.

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. Currently, RDBMS Reader can read data from databases such as Dameng (DM), Db2, PPAS, and Sybase databases. If you want RDBMS Reader to read data from a common relational database, register the driver for the corresponding database type.

Specifically, RDBMS Reader connects to a remote RDBMS database through Java Database Connectivity (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 sends 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
jdbcUrl	<p>The JDBC URL for connecting to the RDBMS database. The format must be in accordance with official RDBMS 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.</p> <ul style="list-style-type: none"> 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 <p>You can enable RDBMS Reader to support a new database as follows:</p> <ul style="list-style-type: none"> Go to the directory of RDBMS Reader, <code>\${DATAX_HOME}/plugin/reader/rdbmswriter</code>. In the preceding directory, <code>\${DATAX_HOME}</code> indicates the main directory of Data Integration. Add the driver of your database to the drivers array in the plugin.json file in the RDBMS Reader directory. RDBMS Reader dynamically selects the appropriate database driver to connect to the database when nodes are run. <pre>{ "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 fewer problems you encounter.", "developer": "alibaba", "drivers": ["dm.jdbc.driver.DmDriver", "com.ibm.db2.jcc.DB2Driver", "com.sybase.jdbc3.jdbc.SybDriver", "com.edb.Driver"] },..</pre> <p>- Add the package of the driver to the libs directory in the RDBMS Reader directory.</p> <pre>... \$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</pre>	Yes	None
Issue: 20200628			271

Parameter	Description	Required	Default value
password	The password for connecting to the database.	Yes	None
table	The name of the table to be synchronized.	Yes	None
column	<p>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.</p> <ul style="list-style-type: none"> 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, such as ["id","1", "'bazhen.csy'", "null", "to_char(a + 1)", "2.3", "true"]. <ul style="list-style-type: none"> 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. The parameter cannot be left empty. 	Yes	None

Parameter	Description	Required	Default value
splitPk	<p>The field used for data sharding when RDBMS 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.</p> <ul style="list-style-type: none"> 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, RDBMS Reader returns an error. If you do not specify the splitPk parameter or leave it empty, RDBMS Reader synchronizes data through a single thread. 	No	N/A
where	<p>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 or <code>gmt_create>\$bizdate</code>.</p> <ul style="list-style-type: none"> 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	<p>The SELECT statement used for refined data filtering. If you specify this parameter, Data Integration directly filters data based on this parameter.</p> <p>For example, if you want to join multiple tables for data synchronization, set this parameter to <code>select a,b from table_a join table_b on table_a.id = table_b.id</code>. If you specify the querySql parameter, RDBMS Reader ignores the table, column, and where parameters that you have configured.</p>	No	None

Parameter	Description	Required	Default value
fetchSize	<p>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.</p> <div>  Note: A value greater than 2048 may lead to out of memory (OOM) during the data synchronization process. </div>	No	1024

Configure RDBMS Reader by using the codeless UI

Currently, the codeless user interface (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"
}

```

8.2.16 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.

Stream Reader data type	Category
STRING	String
LONG	Long integer
DATE	Date and time
BOOLEAN	Boolean
BYTES	Byte

Parameters

Parameter	Description	Required	Default value
column	<p>The data and type of columns in the source table. Multiple columns can be configured. You can set this parameter to generate random strings and specify the range. The example is as follows:</p> <pre>"column" : [{ "random": "8,15" }, { "random": "10,10" }]</pre> <p>The parameters are described as follows:</p> <ul style="list-style-type: none"> • "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.

```
{
  "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",
        "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 Stream Writer. For more information,
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"
        }
    ]
}
}
}

```

8.2.17 Configure HybridDB for MySQL Reader

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

HybridDB for 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 HybridDB for MySQL functions, such as now().

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

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

Data types

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

Category	HybridDB for MySQL data types
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:

- Except for the preceding data types, other types are not supported.
- HybridDB for MySQL Reader considers tinyint(1) as the INTEGER type.

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	<p>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.</p> <ul style="list-style-type: none"> 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 HybridDB for MySQL, for example, ["id", "table", "1", "mingya.wmy", "null", "to_char(a+1)", "2.3", "true"]. <ul style="list-style-type: none"> 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': 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	<p>The field used for data sharding when HybridDB for 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.</p> <ul style="list-style-type: none"> 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, HybridDB for 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	<p>The WHERE clause. For example, set this parameter to <code>gmt_create>\$bizdate</code>.</p> <ul style="list-style-type: none"> 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 this parameter to limit 10, which does not conform to the constraints of HybridDB for MySQL on the SQL WHERE clause. 	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 ["id","table","1","mingya.wmy","","null","to_char(a+1)","2.3","true"]. The priority of the querySql parameter is higher than those of the table , column , where , and splitPk parameters. If you specify the querySql parameter, HybridDB for MySQL Reader ignores the table, column, where, and splitPk parameters that you have configured. The data store specified by the datasource parameter parses information, including the username and password, from this parameter.	No	None
singleOrMulti (only applicable 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, HybridDB for MySQL Reader can only read data from the first shard. The singleOrMulti parameter is only used by the front end, but not by the back end.	Yes	multi

Configure HybridDB for MySQL Reader in the codeless UI

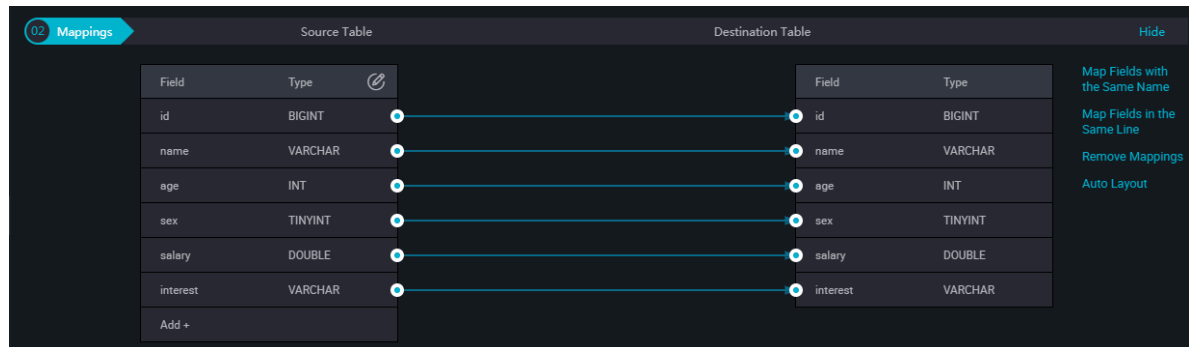
1. Configure the connections.

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	<p>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.</p> <div>  Note: The Shard Key parameter appears only when you configure the source connection for a sync node. </div>

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. The rules for adding fields are described as follows: <ul style="list-style-type: none"> 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. Channel

03 Channel

You can control the sync process by throttling the bandwidth or limiting the dirty data records allowed. [Learn more.](#)

* Expected Concurrency ?

* Bandwidth Throttling ☒ Disable ☐ Enable

Dirty Data Records Allowed dirty records, task ends. ?

Resource Group

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read 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 in which the synchronization task runs. By default, the task runs in the default resource group. If you need to run a large number of tasks and the resources are insufficient, you can purchase exclusive data integration resources or add a custom resource group. For more information, see #unique_15 and Add a custom resource group .

Configure HybridDB for 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",
  "steps": [
    {
      "parameter": {
        "datasource": "px_aliyun_hymysql", // The connection name.
        "column": [ // The columns to be synchronized.
          "id",
          "name",
          "sex",
          "salary",
          "age",
          "pt"
        ]
      }
    }
  ]
}
```



```

    },
    "where": "id=10001", // The WHERE clause.
    "splitPk": "id", // The shard key.
    "table": "person" // The name of the table to be synchronized.
  },
  {
    "name": "Reader",
    "category": "reader"
  },
  {
    "parameter": {}
  },
  {
    "version": "2.0", // The version number.
    "order": {
      "hops": [
        {
          "from": "Reader",
          "to": "Writer"
        }
      ]
    },
    "setting": {
      "errorLimit": { // The maximum number of dirty data records allowed.
        "record": ""
      },
      "speed": {
        "concurrent": 7, // The maximum number of concurrent threads.
        "throttle": true, // Specifies whether to enable bandwidth throttling.
        "mbps": 1, // The maximum transmission rate.
      }
    }
  }
}

```

8.2.18 Configure AnalyticDB for PostgreSQL Reader

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

AnalyticDB for PostgreSQL Reader allows you to read data from AnalyticDB for PostgreSQL

. AnalyticDB for PostgreSQL Reader connects to a remote AnalyticDB for PostgreSQL database and runs a SELECT statement to select and read data from the database.

Relational Database Service (RDS) supports the AnalyticDB for PostgreSQL storage engine.

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

- AnalyticDB for PostgreSQL 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 AnalyticDB for PostgreSQL database.

- If you specify the **querySql** parameter, AnalyticDB for PostgreSQL Reader directly sends the value of this parameter to the AnalyticDB for PostgreSQL database.

Data types

AnalyticDB for PostgreSQL Reader supports most AnalyticDB for PostgreSQL data types. Make sure that your data types are supported.

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


Category	AnalyticDB for PostgreSQL data type
Integer	BIGINT, BIGSERIAL, INTEGER, SMALLINT, and SERIAL
Floating point	DOUBLE, PRECISION, MONEY, NUMERIC, and REAL
String	VARCHAR, CHAR, TEXT, BIT, and INET
Date and time	DATE, TIME, and TIMESTAMP
Boolean	BOOLEAN
Binary	BYTEA

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	<p>The columns to be synchronized from the source table. The columns are described in a JSON array. The default value is <code>[*]</code>, which indicates all columns.</p> <ul style="list-style-type: none"> 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 AnalyticDB for PostgreSQL, for example, <code>["id", "table", "1", "mingya.wmy", "null", "to_char(a+1)", "2.3", "true"]</code>. <ul style="list-style-type: none"> 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': 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	<p>The field used for data sharding when AnalyticDB for 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 runs concurrent threads to synchronize data. This improves efficiency.</p> <ul style="list-style-type: none"> We recommend that you set this 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, AnalyticDB for 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	<p>The WHERE clause. AnalyticDB for PostgreSQLReader 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 <code>id>2 and sex=1</code>.</p> <ul style="list-style-type: none"> 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)	<p>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 <code>select a,b from table_a join table_b on table_a.id = table_b.id</code>.</p> <p>If you specify the querySql parameter, AnalyticDB for PostgreSQL Reader ignores the table, column, and where parameters that you have configured.</p>	No	None
fetchSize	<p>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.</p> <div>  Note: A value larger than 2048 may lead to the out of memory (OOM) error during the data synchronization process. </div>	No	512

Configure AnalyticDB for PostgreSQL Reader by using the codeless UI

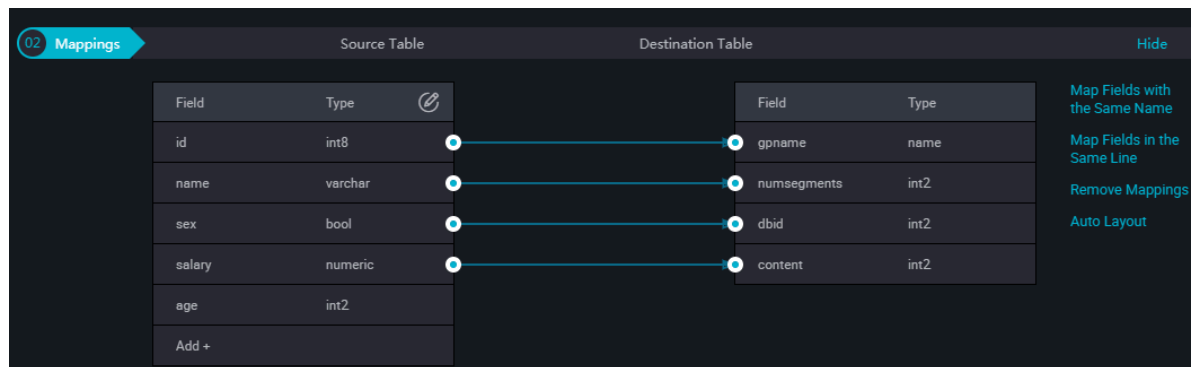
1. Configure the connections.

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	<p>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.</p> <div>  Note: The Shard Key parameter appears only when you configure the source connection for a sync node. </div>

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	<ul style="list-style-type: none"> 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. Channel

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read 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. If a large number of nodes including this sync node are deployed on the default resource group, the sync node may need to wait for resources. We recommend that you purchase an exclusive resource group for data integration or add a custom resource group. For more information, see #unique_15 and Add a custom resource group .

Configure AnalyticDB for PostgreSQLReader by using the code editor

```
{
  "type": "job",
  "steps": [
    {
      "parameter": {
        "datasource": "test_004", // The connection name.
        "column": [ // The columns to be synchronized.
          "id",
          "name",
          "sex",
          "salary",
          "age"
        ],
        "where": "id=1001", // The WHERE clause.
        "splitPk": "id", // The shard key.
        "table": "public.person" // The name of the table to be synchronized.
      },
      "name": "Reader",
```



```

    "category": "reader"
  },
  {
    "parameter": {},
    "name": "Writer",
    "category": "writer"
  }
],
"version": "2.0", // The version number.
"order": {
  "hops": [
    {
      "from": "Reader",
      "to": "Writer"
    }
  ]
},
"setting": {
  "errorLimit": { // The maximum number of dirty data records allowed.
    "record": ""
  },
  "speed": {
    "concurrent": 6, // The maximum number of concurrent threads.
    "throttle": false, // Specifies whether to enable bandwidth throttling.
  }
}
}
}

```

8.2.19 Configure POLARDB Reader

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

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

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

POLARDB 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 POLARDB functions, such as **now()**.

Data types

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

Category	POLARDB data type
Integer	INT, TINYINT, SMALLINT, MEDIUMINT, and BIGINT

Category	POLARDB data type
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:**


- Except for the preceding data types, other types are not supported.
- POLARDB Reader classifies tinyint(1) as the INTEGER type.

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	<p>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.</p> <ul style="list-style-type: none"> 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 POLARDB, for example, ["id", "table","1","'mingya.wmy'", "null", "to_char(a+1)","2.3", "true"]. <ul style="list-style-type: none"> 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': 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	<p>The field used for data sharding when POLARDB 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.</p> <ul style="list-style-type: none"> 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, POLARDB 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	<p>The WHERE clause. For example, set this parameter to gmt_create>\$bizdate.</p> <ul style="list-style-type: none"> 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 POLARDB on the SQL WHERE clause. 	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 <code>select a,b from table_a join table_b on table_a.id = table_b.id</code> . The priority of the querySql parameter is higher than those of the column, table, and where parameters. When you specify the querySql parameter, POLARDB Reader ignores the column , table , and where parameters that you have configured. The data store specified by the datasource parameter parses information, including the username and password, from this parameter.	No	None
singleOrMulti (only applicable 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: <code>"singleOrMulti": "multi"</code> . 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, POLARDB Reader can only read data from the first shard. <div>  Note: The singleOrMulti parameter is only used by the front end, but not by the back end. </div>	Yes	multi

Configure POLARDB Reader by using the codeless UI

1. Configure the connections.

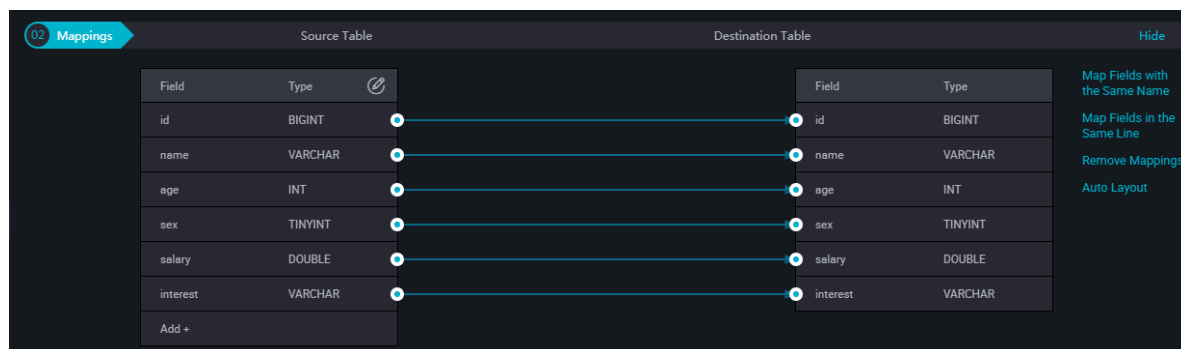
Configure the source and destination connections for the sync node.

The screenshot shows the 'Data Source' configuration window for a sync node. It has two main sections: 'Source' and 'Destination'.
Source Section:
 - * Data Source: POLARDB (dropdown)
 - * Table: polaradb_person (dropdown)
 - Filter: Enter a WHERE clause when you need to synchronize incremental data. Do not include the keyword WHERE. (text area)
 - Shard Key: id (text field)
 - Preview button
Destination Section:
 - * Data Source: POLARDB (dropdown)
 - * Table: polaradb_person (dropdown)
 - Statements Run Before Import: Enter SQL statements. These statements runs before the data is imported. (text area)
 - Statements Run After Import: Enter SQL statements. These statements runs after the data is imported. (text area)
 - * Primary Key: INSERT INTO (dropdown)
 - Violation (text field)

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	<p>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.</p> <div>  Note: The Shard Key parameter is displayed only when you configure the source connection for a sync node. </div>

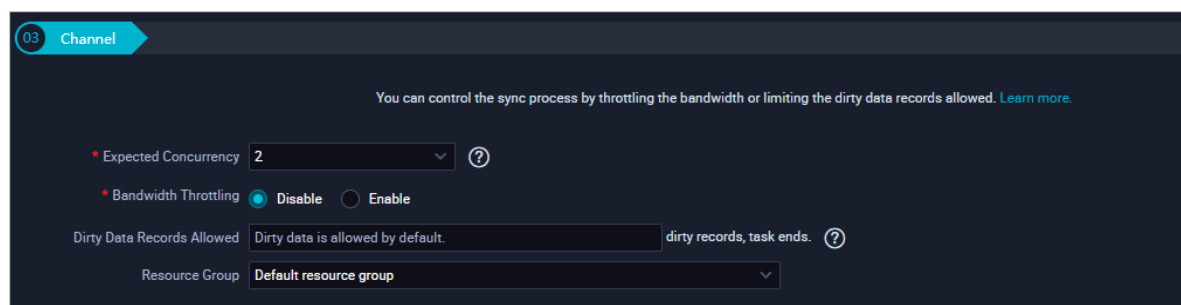
2. Configure field mapping, that is, the **column** parameter in the preceding parameter description.

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.

3. Configure channel control policies.



Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read 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. If a large number of nodes including this sync node are deployed on the default resource group, the sync node may need to wait for resources. We recommend that you purchase an exclusive resource group for data integration or add a custom resource group. For more information, see #unique_15 and Add a custom resource group .

Configure POLARBD 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",
  "steps": [
    {
      "parameter": {
        "datasource": "test_005", // The connection name.
        "column": [ // The columns to be synchronized.
          "id",
          "name",
          "age",
          "sex",
          "salary",
          "interest"
        ],
        "where": "id=1001", // The WHERE clause.
        "splitPk": "id", // The shard key.
        "table": "polardb_person" // The source table name.
      },
      "name": "Reader",
      "category": "reader"
    },
    {
      "parameter": {}
    }
  ],
  "version": "2.0", // The version number.
  "order": {
    "hops": [
      {
        "from": "Reader",
        "to": "Writer"
      }
    ]
  }
}
```



```

    }
  ]
},
"setting": {
  "errorLimit": { // The maximum number of dirty data records allowed.
    "record": ""
  },
  "speed": {
    "concurrent": 6, // The number of concurrent threads.
    "throttle": false, // Specifies whether to enable bandwidth throttling.
  }
}
}
}

```

8.2.20 Configure Elasticsearch Reader

This topic describes the working principles, features, and parameters of Elasticsearch Reader.

How it works

- Elasticsearch Reader reads data from Elasticsearch by slicing scroll queries. The slices are processed by multiple threads of a sync node.
- Data types are converted based on the mapping configuration of Elasticsearch.

For more information, see [documentation in the Elasticsearch official website](#).

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 columns to be synchronized.
          "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 Elasticsearch
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.
How it works	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.

Item	Description
Solution	<ul style="list-style-type: none"> - Elasticsearch Reader converts nested JSON data to single-dimensional data by using the following path formats: <ul style="list-style-type: none"> ■ 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: <p>Property[*]. Child property</p> - Elasticsearch Reader merges data in a string array to one property in the following format and removes duplicates: <p>Property[] where duplicates are removed</p> - Elasticsearch Reader merges multiple properties to one property in the following format: <p>Property 1,Property 2</p> - Elasticsearch Reader presents optional properties in the following format: <p>Property 1 Property 2</p>

Parameters

Parameter	Description	Required	Default value
endpoint	The endpoint of Elasticsearch.	Yes	None
username	The username for HTTP authentication.	No	NULL
password	The password for HTTP authentication.	No	NULL
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

Parameter	Description	Required	Default value
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 period of the client.	No	600,000
readTimeOut	The data reading timeout period of the client.	No	600,000
multiThread	Specifies whether to use multiple threads for an HTTP request.	No	true
column	The field types supported by Elasticsearch.	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 child properties.	No	false

8.2.21 Configure Kafka Reader

Kafka Reader allows you to read data from Kafka by using the Java SDK for Kafka.

Apache Kafka is a fast, scalable, and distributed message system with high throughput and fault-tolerance. It is used to publish and subscribe to messages. With the high-throughput, built-in partition, data replica, and fault-tolerance features, Kafka is suitable for processing a large amount of messages.

How it works

Kafka Reader reads data from Kafka by using the following version of the **Java SDK for Kafka**:

```
<dependency>
  <groupId>org.apache.kafka</groupId>
  <artifactId>kafka-clients</artifactId>
  <version>2.0.0</version>
</dependency>
```

Kafka Reader uses the following methods of the Java SDK for Kafka. For more information about the functions and limits of the SDK, see the Kafka official documentation.

- Use `KafkaConsumer` as the client that consumes messages.

```
org.apache.kafka.clients.consumer.KafkaConsumer<K,V>
```

- Query offsets based on the UNIX timestamp.

```
Map<TopicPartition,OffsetAndTimestamp> offsetsForTimes(Map<TopicPartition,Long>
timestampsToSearch)
```

- Seek to the start offset.

```
public void seekToBeginning(Collection<TopicPartition> partitions)
```

- Seek to the end offset.

```
public void seekToEnd(Collection<TopicPartition> partitions)
```

- Seek to the specified offset.

```
public void seek(TopicPartition partition,long offset)
```

- Poll for data from the server.

```
public ConsumerRecords<K,V> poll(final Duration timeout)
```





Note:

Kafka Reader automatically commits offsets when consuming data.

Parameters

Parameter	Description	Required
server	The broker server address of Kafka in the format of ip:port .	Yes
topic	The name of the topic from which data is read in Kafka. Kafka maintains feeds of messages in categories called topics.	Yes

Parameter	Description	Required
column	<p>The columns to be read by Kafka Reader. Constant, data, and property columns are supported.</p> <ul style="list-style-type: none"> Constant column: a column enclosed in single quotation marks (' '), such as ["abc", "123"]. Data column <ul style="list-style-type: none"> If your data is in JSON format, you can obtain JSON properties, such as ["event_id"]. If your data is in JSON format, you can obtain nested child JSON properties, such as ["tag.desc"]. Property column <ul style="list-style-type: none"> __key__: the key of the message. __value__: the complete content of the message. __partition__: the partition where the message resides. __headers__: the header of the message. __offset__: the offset of the message. __timestamp__: the timestamp of the message. <p>Example:</p> <pre>"column": ["__key__", "__value__", "__partition__", "__offset__", "__timestamp__", "123", "event_id", "tag.desc"]</pre>	Yes
keyType	The key type in Kafka. Valid values: BYTEARRAY, DOUBLE, FLOAT, INTEGER, LONG, and SHORT.	Yes
valueType	The value type in Kafka. Valid values: BYTEARRAY, DOUBLE, FLOAT, INTEGER, LONG, and SHORT.	Yes

Parameter	Description	Required
beginDateTime	The start timestamp for consuming data. This parameter defines the left boundary of the left-closed and right-open interval. The value is a time string in the yyyymmddhhmmss format. It can be used in combination with scheduling properties. For more information, see #unique_176 . Kafka 0.10.2 and later versions support this parameter.	You can set either beginDateTime or beginOffset, but not both.  Note: The beginDateTime and endDateTime parameters must be used in pairs.
endDateTime	The end timestamp for consuming data. This parameter defines the right boundary of the left-closed and right-open interval. The value is a time string in the yyyymmddhhmmss format. It can be used in combination with scheduling properties. For more information, see #unique_176 . Kafka 0.10.2 and later versions support this parameter.	You can set either endDateTime or endOffset, but not both.  Note: The beginDateTime and endDateTime parameters must be used in pairs.
beginOffset	The offset where data consumption starts, in one of the following formats: <ul style="list-style-type: none"> Numeric string: starts to consume data from the specified offset, such as 15553274. seekToBeginning: starts to consume data from the start offset. seekToLast: starts to consume data from the last offset. seekToEnd: starts to consume data from the end offset. The read data is null. 	You can set either beginOffset or beginDateTime, but not both.
endOffset	The offset where data consumption ends.	You can set either endOffset or endDateTime, but not both.

Parameter	Description	Required
skipExceedRecord	<p>Kafka uses public <code>ConsumerRecords<K, V> poll(final Duration timeout)</code> for data consumption. However, the data obtained in a poll may be beyond the boundary specified by the <code>endOffset</code> or <code>endTime</code> parameter. <code>skipExceedRecord</code> specifies whether to export such excess data to the destination. Kafka automatically commits offsets for data consumption. Therefore, we recommend that you set the <code>skipExceedRecord</code> parameter as follows:</p> <ul style="list-style-type: none"> Set <code>skipExceedRecord</code> to <code>false</code> for versions earlier than Kafka 0.10.2. Set <code>skipExceedRecord</code> to <code>true</code> for Kafka 0.10.2 and later versions. 	No. Default value: <code>false</code> .
partition	The partition of the topic in Kafka. Generally, Kafka Reader reads data in a specified offset interval from multiple partitions of a topic. If you want Kafka Reader to read data from a specified partition, you can set the <code>partition</code> parameter.	No. Default value: none.
kafkaConfig	The extended parameters specified when <code>KafkaConsumer</code> is created, such as <code>bootstrap.servers</code> , <code>auto.commit.interval.ms</code> , and <code>session.timeout.ms</code> . By setting parameters in <code>kafkaConfig</code> , you can control the data consumption behaviors of <code>KafkaConsumer</code> .	No

The following extended parameters can be configured in `kafkaConfig`:

- `fetch.min.bytes`: the minimum number of bytes that the consumer can obtain from the broker. The broker waits until the number of bytes reaches the specified value before returning data to the consumer.
- `fetch.max.wait.ms`: the maximum duration during which the consumer waits for data from the broker. Default value: 500 ms. The broker returns data to the consumer so long as either the minimum number of bytes or the maximum waiting duration reaches the values specified by the `fetch.min.bytes` and `fetch.max.wait.ms` parameters.
- `max.partition.fetch.bytes`: the maximum number of bytes in each partition returned by the broker to the consumer. Default value: 1 MB.

- `session.timeout.ms`: the timeout duration for disconnecting the consumer from the broker. After that, the consumer no longer receives data from the broker. Default value: 30s.
- `auto.offset.reset`: the handling method when no offset is available or the offset is invalid. This occurs if the consumer times out or the record with the specified offset has expired and been deleted. Default value: `latest`, which specifies that the consumer reads data from the latest record. You can change this value to `earliest`, which specifies that the consumer reads data from the start offset.
- `max.poll.records`: the number of messages that can be returned for a poll.
- `key.deserializer`: the method used to deserialize the message key, such as **`org.apache.kafka.common.serialization.StringDeserializer`**.
- `value.deserializer`: the method used to deserialize the message value, such as **`org.apache.kafka.common.serialization.StringDeserializer`**.
- `ssl.truststore.location`: the path of the Secure Sockets Layer (SSL) root certificate.
- `ssl.truststore.password`: the password of the truststore in the SSL root certificate. If Message Queue for Apache Kafka is used, set this parameter to `KafkaOnsClient`.
- `security.protocol`: the access protocol. Only Simple Authentication and Security Layer (SASL) SSL is supported.
- `sasl.mechanism`: the SASL authentication mode. If Message Queue for Apache Kafka is used, set this parameter to `PLAIN`.

Example:

```
{
  "group.id": "demo_test",
  "java.security.auth.login.config": "/home/admin/kafka_client_jaas.conf",
  "ssl.truststore.location": "/home/admin/kafka.client.truststore.jks",
  "ssl.truststore.password": "KafkaOnsClient",
  "security.protocol": "SASL_SSL",
  "sasl.mechanism": "PLAIN",
  "ssl.endpoint.identification.algorithm": ""
}
```

Configure Kafka Reader by using the code editor

In the following code, a node is configured to read data from a Kafka topic.

```
{
  "type": "job",
  "steps": [
    {
      "stepType": "kafka",
      "parameter": {
        "server": "host:9093",
        "column": [
```

```

        "_key_",
        "_value_",
        "_partition_",
        "_offset_",
        "_timestamp_",
        "123",
        "event_id",
        "tag.desc"
    ],
    "kafkaConfig": {
        "group.id": "demo_test"
    },
    "topic": "topicName",
    "keyType": "ByteArray",
    "valueType": "ByteArray",
    "beginDateTime": "20190416000000",
    "endDateTime": "20190416000006",
    "skipExceedRecord": "false"
},
"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"
    },
    "speed": {
        "throttle": false,
        "concurrent": 1,
        "dmu": 1
    }
}
}

```

8.2.22 Configure MaxCompute Reader

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

MaxCompute Reader allows you to read data from MaxCompute. For more information about MaxCompute, see [MaxCompute overview](#).

Based on the specified information such as the source project, table, partition, and field, MaxCompute Reader reads data from MaxCompute through Tunnel. For more information about common Tunnel commands, see [Tunnel commands](#).

MaxCompute Reader cannot read views. It can only read partitioned tables and non-partitioned tables. When enabling 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, and 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 identical to 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 case-insensitive.	Yes	None

Parameter	Description	Required	Default value
partition	<p>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.</p> <ul style="list-style-type: none">• If you want to read data from the partition with pt=1 and ds=shanghai, enter "partition": "pt=1/ds=shanghai".• If you want to read data from all the partitions with pt=1, enter "partition": "pt=1/ds=*".• If you want to read data from all the partitions in the test table, enter "partition": "pt=*/ds=*".	Required only for partitioned table	None

Parameter	Description	Required	Default value
column	<p>The columns in the source table that MaxCompute Reader reads. Assume that the fields of a table named test are id, name, and age.</p> <ul style="list-style-type: none"> To read the fields in turn, enter <code>"column":["id","name","age"]</code> or <code>"column":["*"]</code>. <div style="background-color: #f0f0f0; padding: 10px; margin: 10px 0;">  Note: We recommend that you do not set <code>"column":["*"]</code>. This is because data synchronization may fail if the source table changes in the column order, data type, or number of columns. </div> <ul style="list-style-type: none"> To read the name and id fields in turn, enter <code>"column":["name","id"]</code>. 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 <code>"column":["age","name","'1988-08-08 08:08:08','id']</code>, the data extracted contains an age column, a name column, a constant "1988-08-08 08:08:08", and an id column in turn. <p>The single quotation marks (' ') are used to identify constant columns. The constant column values exclude the single quotation marks (' ').</p> <div style="background-color: #f0f0f0; padding: 10px; margin: 10px 0;">  Note: <ul style="list-style-type: none"> 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. </div>	Yes	None

Configure MaxCompute 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.
Partition Information	The partitions to be synchronized.
Compression	Specifies whether to enable compression. Valid values: Enable and Disable .
Convert Empty Strings to Null	Specifies whether to convert empty strings to null.



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.

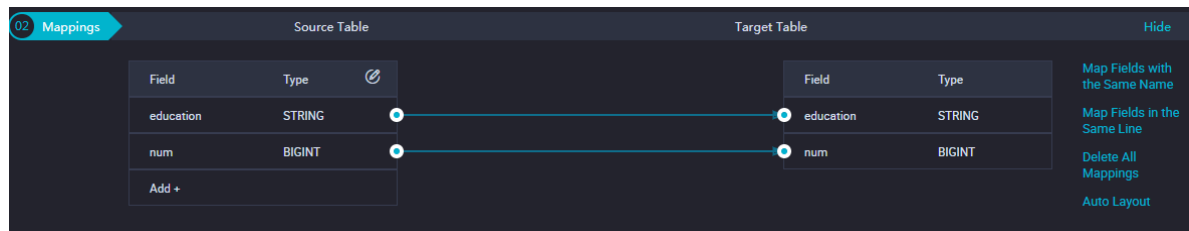
You can specify the partition key columns to be synchronized. Assume that the partition key column of a MaxCompute table is **pt=\${bdp.system.bizdate}**. You can configure the column to be synchronized to pt. Ignore it if the column is marked as unidentified.

- To synchronize all partitions, enter **pt=***.

- To synchronize some of the partitions, specify the corresponding dates.

2. Configure field mappings, 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	<ul style="list-style-type: none"> • 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.

03 Channel

You can control the sync process by throttling the bandwidth or limiting the dirty data records allowed. [Learn more.](#)

* Expected Maximum ?
Concurrency

* Bandwidth Throttling ☒ Disable ☐ Enable ?

Dirty Data Records Allowed ? The node automatically ends when the number of dirty data records reaches XX.

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read 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. If a large number of nodes including this sync node are deployed on the default resource group, the sync node may need to wait for resources. We recommend that you purchase an exclusive resource group for data integration or add a custom resource group. For more information, see #unique_15 and Add a custom resource group .

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",
  "version": "2.0",
  "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"
},
{ // The following template is used to configure Stream Writer. For more information,
  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"
        }
    ]
}
}

```

If you want to specify the [Tunnel endpoint](#) of MaxCompute, you can configure the connection in the code editor. To configure the connection, replace "datasource": "", in the preceding code with detailed parameters of the connection. Example:

```

"accessId":"*****",
"accessKey":"*****",
"endpoint":"http://service.eu-central-1.maxcompute.aliyun-inc.com/api",
"odpsServer":"http://service.eu-central-1.maxcompute.aliyun-inc.com/api",
"tunnelServer":"http://dt.eu-central-1.maxcompute.aliyun.com",

```

```
"project": "*****",
```

8.2.23 Configure InfluxDB Reader

InfluxDB is an open-source time series database developed by InfluxData. InfluxDB is written in Go and is designed for querying and storing time series data with high performance. InfluxDB Reader allows you to read data from InfluxDB.

Currently, you can only configure InfluxDB Reader by using the code editor. For more information, see [InfluxDB](#).

How it works

InfluxDB Reader connects to an InfluxDB instance through the Java client, sends an SQL statement to the instance, and obtains data points through the scan operations. A sync node is split to multiple export tasks based on the database, metric, and time range.

Limits



- 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 is converted to [3:00, 4:00).
- Currently, only InfluxDB 0.9 and later are supported.

Data types

Category	Data Integration data type	InfluxDB data type
String	String	String to which a data point in InfluxDB is serialized, including the timestamp, metric, tags, and value

Parameters

Parameter	Description	Required	Default value
endpoint	The HTTP endpoint for connecting to InfluxDB, in the format of http://IP:Port.	Yes	None
database	The name of the InfluxDB database.	Yes	None

Parameter	Description	Required	Default value
username	The username for connecting to the InfluxDB database.	Yes	None
password	The password for connecting to the InfluxDB database.	Yes	None
column	The metrics to be migrated.	Yes	None
beginDateTime	The start time of the time range of the data points to be migrated , in the format of yyyyMMddHHmmss. Specify the beginDateTime and endDateTime parameters to determine the time range of the data points to be migrated .	Yes	None <div>  Note: The start time and end time of the time range 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 is converted to [3:00, 4:00). </div>
endDateTime	The end time of the time range of the data points to be migrated , in the format of yyyyMMddHHmmss. Specify the beginDateTime and endDateTime parameters to determine the time range of the data points to be migrated .	Yes	None <div>  Note: The start time and end time of the time range 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 is converted to [3:00, 4:00). </div>

Configure InfluxDB Reader by using the codeless UI

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

Configure InfluxDB Reader by using the code editor

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

```
```json
{
 "order": {
 "hops": [
 {
 "from": "Reader",
 "to": "Writer"
 }
]
 },
 "setting": {
 "errorLimit": {
 "record": "0"
 },
 "speed": {
 "concurrent": 1,
 "throttle": true
 }
 },
 "steps": [
 {
 "category": "reader",
 "name": "Reader",
 "parameter": {
 "endpoint": "http://host:8086",
 "database": "",
 "username": "",
 "password": "",
 "column": [
 "xc"
]
 },
 "endDateTime": "20190515180000",
 "beginDateTime": "20190515170000"
 },
 {
 "category": "writer",
 "name": "Writer",
 "parameter": {},
 "stepType": ""
 }
],
 "type": "job",
 "version": "2.0"
}
```

...

## 8.2.24 Configure OpenTSDB Reader

OpenTSDB is a scalable and distributed time series database maintained by Yahoo.

OpenTSDB Reader allows you to read data from OpenTSDB.

Currently, you can only configure OpenTSDB Reader by using the code editor.

### How it works

OpenTSDB Reader connects to an OpenTSDB instance by sending an HTTP request, and obtains the connection information of the underlying HBase data store through the `/api/config` HTTP API operation. Then, OpenTSDB Reader connects to HBase through the AsyncHBase client and reads data points through scan operations. A sync node is split to multiple export tasks based on the database, metric, and time range. An export task exports the data of the specified metric in the specified hour.

### Limits



- 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 is converted to `[3:00, 4:00)`.
- Currently, only OpenTSDB 2.3.x is supported.
- You cannot directly query the data points by using the `/api/query` HTTP API operation. Before querying data points, connect to the underlying data store of OpenTSDB.

If you directly use the `/api/query` HTTP API operation to read a large amount of data points, the AsyncHBase client of OpenTSDB reports an exception of excessive callbacks. To avoid this issue, OpenTSDB Reader connects to the underlying HBase data store and reads data points through scan operations. To improve query efficiency, you can specify the metric and time range to allow OpenTSDB Reader to scan HBase tables in sequence.

### Data types

Category	Data Integration data type	OpenTSDB data type
String	String	String to which a data point in OpenTSDB is serialized , including the timestamp, metric, tags, and value

**Parameters**

Parameter	Description	Required	Default value
endpoint	The HTTP endpoint for connecting to OpenTSDB, in the format of http://IP:Port.	Yes	None
column	The metrics to be migrated.	Yes	None
beginDateTime	The start time of the time range of the data points to be migrated , in the format of yyyyMMddHHmmss. Specify the beginDateTime and endDateTime parameters to determine the time range of the data points to be migrated .	Yes	None   <b>Note:</b> The start time and end time of the time range 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 is converted to [3:00, 4:00).
endDateTime	The end time of the time range of the data points to be migrated , in the format of yyyyMMddHHmmss. Specify the beginDateTime and endDateTime parameters to determine the time range of the data points to be migrated .	Yes	None   <b>Note:</b> The start time and end time of the time range 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 is converted to [3:00, 4:00).

## Configure OpenTSDB Reader by using the codeless UI

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

## Configure OpenTSDB Reader by using the code editor

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

```
```json
{
  "order": {
    "hops": [
      {
        "from": "Reader",
        "to": "Writer"
      }
    ]
  },
  "setting": {
    "errorLimit": {
      "record": "0"
    },
    "speed": {
      "concurrent": 1,
      "throttle": true
    }
  },
  "steps": [
    {
      "category": "reader",
      "name": "Reader",
      "parameter": {
        "endpoint": "http://host:4242",
        "column": [
          "xc"
        ],
        "beginDateTime": "20190101000000",
        "endDateTime": "20190101030000"
      },
      "stepType": "opentsdb"
    },
    {
      "category": "writer",
      "name": "Writer",
      "parameter": {},
      "stepType": ""
    }
  ],
  "type": "job",
  "version": "2.0"
}
```


...

Performance test report

- Characteristics of test data

The data used in the performance test is described as follows:

- Metric: a metric, which is m.
- tag_k and tag_v: the key and value of the tag. The keys and values of the first four tags constitute a time series of 2,000,000 data points, that is, 10 (zones) × 20 (clusters) × 100 (groups) × 100 (apps). The ip tag corresponds to the index of the 2,000,000 data points, starting from 1.

tag_k	tag_v
zone	z1 to z10
cluster	c1 to c20
group	g1 to g100
app	a1 to a100
ip	ip1 to ip2,000,000

- value: a random value from 1 to 100.
- interval: a collection period of 10 seconds. The data collection lasts for 3 hours, and a total of 2,160,000,000 data points, that is $3 \times 60 \times 60 / 10 \times 2,000,000$, are collected.
- Performance test results

Number of channels	Data integration speed (records per second)	Data integration bandwidth (Mbit/s)
1	215,428	25.65
2	424,994	50.60
3	603,132	71.81

8.2.25 Configure Prometheus Reader

Prometheus is a time series database developed and maintained by SoundCloud. It is the open-source implementation of BorgMon, which is the monitoring system of Google. Prometheus Reader allows you to read data from Prometheus.

Currently, you can only configure Prometheus Reader by using the code editor.

How it works

Prometheus Reader connects to a Prometheus instance by sending an HTTP request, and obtains raw data points through the `/api/v1/query_range` HTTP API operation. A sync node is split to multiple export tasks based on the metric and time range.

Limits

- 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 is converted to `[3:00, 4:00)`.
- Currently, only Prometheus 2.9.x is supported.
- The default time range for reading data is 10s.



The `/api/v1/query_range` HTTP API operation only allows you to query a limited number of data points. If you specify a too large time range, the following error message is returned: `exceeded maximum resolution of 11,000 points per timeseries`. Therefore, Prometheus Reader obtains data at the time range of 10s by default. Even if the raw data points are stored by millisecond, up to 10,000 data points can be queried through the `/api/v1/query_range` HTTP API operation.

Data types

Category	Data Integration data type	Prometheus data type
String	String	String to which a data point in Prometheus is serialized , including the timestamp, metric, tags, and value

Parameters

Parameter	Description	Required	Default value
endpoint	The HTTP endpoint for connecting to Prometheus, in the format of <code>http://IP:Port</code> .	Yes	None
column	The metrics to be migrated.	Yes	None

Parameter	Description	Required	Default value
beginDateTime	The start time of the time range of the data points to be migrated , in the format of yyyyMMddHHmmss. Specify the beginDateTime and endDateTime parameters to determine the time range of the data points to be migrated .	Yes	None  Note: The start time and end time of the time range 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 is converted to [3:00, 4:00).
endDateTime	The end time of the time range of the data points to be migrated , in the format of yyyyMMddHHmmss. Specify the beginDateTime and endDateTime parameters to determine the time range of the data points to be migrated .	Yes	None  Note: The start time and end time of the time range 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 is converted to [3:00, 4:00).

Configure Prometheus Reader by using the codeless UI

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

Configure Prometheus Reader by using the code editor

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

```
```json
{
 "order": {
 "hops": [
 {
 "from": "Reader",
```

```

 "to": "Writer"
 }
]
 },
 "setting": {
 "errorLimit": {
 "record": "0"
 },
 "speed": {
 "concurrent": 1,
 "throttle": true
 }
 },
 "steps": [
 {
 "category": "reader",
 "name": "Reader",
 "parameter": {
 "endpoint": "http://localhost:9090",
 "column": [
 "up"
],
 "beginDateTime": "20190520150000",
 "endDateTime": "20190520160000"
 },
 "stepType": "prometheus"
 },
 {
 "category": "writer",
 "name": "Writer",
 "parameter": {},
 "stepType": ""
 }
],
 "type": "job",
 "version": "2.0"
}

```

### Performance test report

Number of channels	Data integration speed (records per second)	Data integration bandwidth (Mbit/s)
1	45,000	5.36
2	55,384	6.60
3	60,000	7.15

## 8.2.26 Configure PostgreSQL Reader

This topic describes the data types and parameters supported by PostgreSQL Reader and how to configure it by using the codeless user interface (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 then 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 sends 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 SELECT 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. Make sure 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
Floating point	double, precision, money, numeric, and real
String	varchar, char, text, bit, and inet
Date and time	date, time, and timestamp
Boolean	boolean
Binary	bytea




#### Note:

- Except for the preceding data types, other types 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
<b>datasource</b>	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None
<b>table</b>	The name of the table to be synchronized.	Yes	None
<b>column</b>	<p>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.</p> <ul style="list-style-type: none"> <li>Column pruning is supported. You can select and export specific columns.</li> <li>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.</li> <li>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"] . <ul style="list-style-type: none"> <li><b>id</b>: a column name.</li> <li><b>table</b>: the name of a column that contains reserved keywords.</li> <li><b>1</b>: an integer constant.</li> <li><b>'mingya.wmy'</b>: a string constant, which is enclosed in single quotation marks ( ' ' ).</li> <li><b>'null'</b>: a string.</li> <li><b>to_char(a+1)</b>: a function expression.</li> <li><b>2.3</b>: a floating-point constant.</li> <li><b>true</b>: a Boolean value.</li> </ul> </li> <li>The <b>column</b> parameter must explicitly specify a set of columns to be synchronized. The parameter cannot be left empty.</li> </ul>	Yes	None

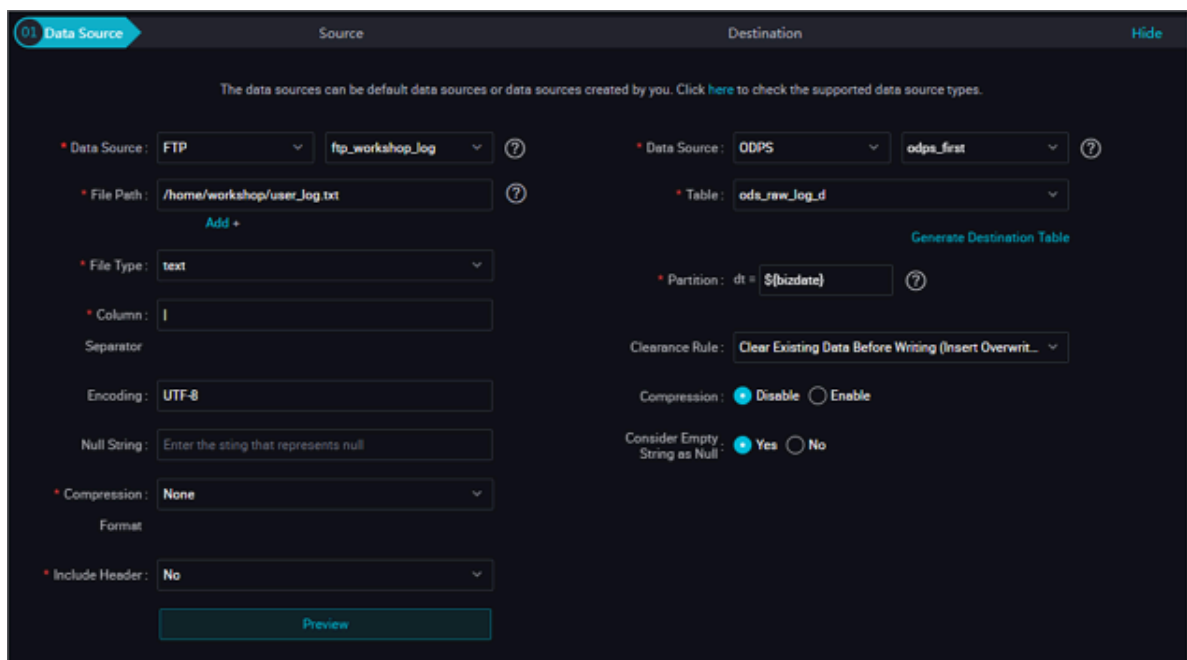
Parameter	Description	Required	Default value
<b>splitPk</b>	<p>The field used for data sharding when PostgreSQL Reader extracts data. If you specify the <b>splitPk</b> parameter, the table is sharded based on the shard key specified by this parameter. Data Integration then initiates concurrent sync threads, which improves efficiency.</p> <ul style="list-style-type: none"> <li>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.</li> <li>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, PostgreSQL Reader ignores the splitPk parameter and synchronizes data through a single thread.</li> <li>If you do not specify the splitPk parameter or leave it empty, Data Integration synchronizes data through a single thread.</li> </ul>	No	None
<b>where</b>	<p>The WHERE clause. PostgreSQL Reader generates a SELECT statement based on the <b>table</b>, <b>column</b>, and <b>where</b> parameters that you have configured, and uses the generated SELECT statement to select and read data. For example, set this parameter to <code>id&gt;2 and sex=1</code>.</p> <ul style="list-style-type: none"> <li>You can use the WHERE clause to synchronize incremental data.</li> <li>If you do not specify the where parameter or leave it empty, all data is synchronized.</li> </ul>	No	None
<b>querySql</b> (only available in the code editor)	<p>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 <code>select a,b from table_a join table_b on table_a.id = table_b.id</code>. If you specify the querySql parameter, PostgreSQL Reader ignores the table, column, where, and splitPk parameters that you have configured.</p>	No	None

Parameter	Description	Required	Default value
<b>fetchSize</b>	<p>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.</p> <div>  <b>Note:</b>            A value greater than 2048 may lead to out of memory (OOM) during the data synchronization process.         </div>	No	512

## Configure PostgreSQL Reader by using the codeless UI


### 1. Configure the connections.

Configure the source and destination connections for the sync node.



Parameter	Description
<b>Connection</b>	The <b>datasource</b> parameter in the preceding parameter description. Select a connection type, and enter the name of a connection that has been configured in DataWorks.
<b>Table</b>	The <b>table</b> parameter in the preceding parameter description.
<b>Filter</b>	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.



Parameter	Description
<b>Shard Key</b>	<p>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.</p> <p>If data sharding is performed based on the configured shard key, data can be read concurrently to improve data synchronization efficiency.</p> <div>  <b>Note:</b>  The Shard Key parameter appears only when you configure the source connection for a sync node. </div>

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.



Button or icon	Description
<b>Map Fields with the Same Name</b>	Click <b>Map Fields with the Same Name</b> to establish a mapping between fields with the same name. Note that the data types of the fields must match.
<b>Map Fields in the Same Line</b>	Click <b>Map Fields in the Same Line</b> to establish a mapping for fields in the same row. Note that the data types of the fields must match.
<b>Delete All Mappings</b>	Click <b>Delete All Mappings</b> to remove mappings that have been established.
<b>Auto Layout</b>	Click Auto Layout. The fields are automatically sorted based on specified rules.
<b>Change Fields</b>	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.

Button or icon	Description
<b>Add</b>	<ul style="list-style-type: none"> <li>Click Add to add a field. You can enter constants. Each constant must be enclosed in single quotation marks (' '), such as 'abc' and '123'.</li> <li>You can use scheduling parameters, such as \${bizdate}.</li> <li>You can enter functions supported by relational databases, such as now() and count(1).</li> <li>Fields that cannot be parsed are indicated by Unidentified.</li> </ul>

### 3. Configure channel control policies.

**03 Channel**

You can control the sync process by throttling the bandwidth or limiting the dirty data records allowed. [Learn more.](#)

\* Expected Maximum  [?](#)  
Concurrency

\* Bandwidth Throttling ☒ Disable ☐ Enable [?](#)

Dirty Data Records Allowed  [?](#) The node automatically ends when the number of dirty data records reaches XX.

Parameter	Description
<b>Expected Maximum Concurrency</b>	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.
<b>Bandwidth Throttling</b>	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.
<b>Dirty Data Records Allowed</b>	The maximum number of dirty data records allowed.
<b>Resource Group</b>	The resource group used for running the sync node. If a large number of nodes including this sync node are deployed on the default resource group, the sync node may need to wait for resources. We recommend that you purchase an exclusive resource group for data integration or add a custom resource group. For more information, see <a href="#">#unique_15</a> and <a href="#">Add a custom resource group</a> .

## 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 Integratio
n 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 Stream Writer. For more information,
 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"
 }
]
 }
}
```

## 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 sync 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 guaranteed when you enable PostgreSQL Reader to run concurrent threads on a single sync 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. 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

A PostgreSQL database supports only EUC\_CN and UTF-8 encoding formats for simplified Chinese characters. PostgreSQL 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 PostgreSQL database but data is written to the PostgreSQL database in a different encoding format, 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 the following ways:

- For data in batches, incremental add, update, and delete operations are distinguished by timestamps. The delete operations include logical delete operations. 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.

## 8.2.27 Configure OTSStream Reader

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

OTSStream Reader allows you to export the incremental data from Table Store. Incremental data can be considered as operations logs that include data and operation information.

Unlike plug-ins for exporting full data, OTSStream Reader only supports the multi-version mode. In addition, OTSStream Reader cannot export the data of specified columns. Before using OTSStream Reader, make sure that the Stream feature is enabled for the source table. You can enable this feature when creating the table or by using the UpdateTable operation in the SDK.

The method for enabling Stream is described as follows:

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

```
client.updateTable(updateTableRequest);
```

You can enable the Stream feature and set the expiration time by using the UpdateTable operation in the SDK. After this feature is enabled, you can export the incremental data of Table Store. After the Stream feature is enabled, the Table Store server saves your operations logs additionally. Each partition has a sequential operations log queue. Each operations log is removed by garbage collection after the specified expiration time.

The Table Store SDK provides several Stream API operations for reading these operations logs. OTSStream Reader obtains incremental data by using these API operations. By default, OTSStream Reader transforms incremental data to multiple 6-tuples (pk, colName, version, colValue, opType, and sequenceInfo) and imports them to MaxCompute.

### Export data by column

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

You can perform read/write operations by using the Table Store API. Table Store stores the incremental data by storing the records of recent write and modify operations on table data. Incremental data can be considered as a set of operation records.

Table Store supports the following three types of modify operations:

- **PutRow**: writes a row. If the row already exists, it is overwritten.
- **UpdateRow**: updates a row without modifying other data of the original row. You can add column values, overwrite column values if the corresponding version of a column already exists, delete all the versions of a column, or delete a version of a column.
- **DeleteRow**: deletes a row.

Table Store generates incremental data records based on each type of operations.

OTSStream Reader reads these records and exports the data in the format of Data Integration.

Table Store 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 Table Store. A row in Table Store may correspond to multiple exported rows. Each exported 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 to the Data Integration format, 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 corresponding column according to the primary key and the column name.
- DR (DELETE\_ROW): deletes a row. Delete all the data of the row according to the primary key.

In the following example, the table has two primary key columns: `pkName1` and `pkName2`.

<code>pkName1</code>	<code>pkName2</code>	<code>columnName</code>	<code>timestamp</code>	<code>columnValue</code>	<code>opType</code>
<code>pk1_V1</code>	<code>pk2_V1</code>	<code>col_a</code>	1441803688001	<code>col_val1</code>	U
<code>pk1_V1</code>	<code>pk2_V1</code>	<code>col_a</code>	1441803688002	<code>col_val2</code>	U
<code>pk1_V1</code>	<code>pk2_V1</code>	<code>col_b</code>	1441803688003	<code>col_val3</code>	U
<code>pk1_V2</code>	<code>pk2_V2</code>	<code>col_a</code>	1441803688000	N/A	DO
<code>pk1_V2</code>	<code>pk2_V2</code>	<code>col_b</code>	N/A	N/A	DA
<code>pk1_V3</code>	<code>pk2_V3</code>	N/A	N/A	N/A	DR
<code>pk1_V3</code>	<code>pk2_V3</code>	<code>col_a</code>	1441803688005	<code>col_val1</code>	U

In this example, seven rows are exported, corresponding to three rows in the Table Store 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 deleting all 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`.

## Export data by row

You can also export data by row. In this mode, OTSStream Reader exports operation records as rows. You must specify the export mode and the columns to be exported, as shown in the following example.

```
"parameter": {
 # Set other parameters, such as datasource and table, as usual.
 "mode": "single_version_and_update_only", # The export mode.
 "column": [# The columns to be exported from Table Store. You can specify the columns
 as needed.
 {
 "name": "uid" # The name of the column, which can be a primary key column or a
 property column.
 },
 {
 "name": "name" # The name of the column, which can be a primary key column
 or a property column.
 },
],
 "isExportSequenceInfo": false, # Specifies whether to export time-series information. If
 you set the mode parameter to single_version_and_update_only, this parameter can only
 be set to false.
}
```

The data exported by row is closer to the data in the original rows, which facilitates further data processing. Note the following points when you export data by row:

- The exported rows are extracted from operation records. Each exported row corresponds to a write or modify operation. If you only modify some of the columns for a row, the operation record only contains the modified rows, and other columns are empty.
- If you export data by row, the version number of each column, which is the timestamp of each column, cannot be exported. In addition, delete operations cannot be exported.

## Data types

Currently, OTSStream Reader supports all Table Store 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
<b>datasource</b>	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None
<b>table</b>	The name of the table from which incremental data is exported. You must enable the Stream feature for a table when creating the table, or by calling the UpdateTable operation after creating the table.	Yes	None

Parameter	Description	Required	Default value
<b>statusTable</b>	<p>The name of the table used by OTSStream Reader to store status records. These records help to filter out the data that is not covered by the target range and improve export efficiency. The table specified by the statusTable parameter is used to store status records. If the specified table does not exist, OTSStream Reader automatically creates the table. After a sync node is completed, you do not need to delete the table. The status records in the table can be used by the next sync node.</p> <ul style="list-style-type: none"> <li>You do not need to create the table in advance. You only need to provide the table name. If the specified table does not exist, OTSStream Reader attempts to create the table under your instance. If the specified table exists, OTSStream Reader checks whether the metadata of the table is as expected. If the metadata is not as expected, an exception is thrown.</li> <li>After a sync node is completed, you do not need to delete the table. The status records in the table can be used by the next sync node.</li> <li>Data in the table automatically expires based on the time-to-live (TTL). Therefore, the data volume is small.</li> <li>You can use the same table to store status records of multiple tables that are managed by the same instance. The status records are independent of each other.</li> </ul> <p>You must configure a name that is easily recognizable, such as <b>TableStoreStreamReaderStatusTable</b>. Note that the name cannot be the same as that for any business-related table.</p>	Yes	None

Parameter	Description	Required	Default value
<b>startTimes tampMillis</b>	<p>The start time (included) in milliseconds of the incremental data.</p> <ul style="list-style-type: none"> <li>OTSSStream Reader searches for the status records in the table specified by the statusTable parameter based on the time specified by the <b>startTimes tampMillis</b> parameter and reads data from this time point.</li> <li>If OTSSStream Reader cannot find status records of this time point, OTSSStream Reader reads incremental data retained by the system from the first entry, and skips the data that is written later than the time specified by the <b>startTimestampMillis</b> parameter.</li> </ul>	No	None
<b>endTime mpMillis</b>	<p>The end time (excluded) in milliseconds of the incremental data.</p> <ul style="list-style-type: none"> <li>OTSSStream Reader reads data from the time specified by the <b>startTimestampMillis</b> parameter and stops to read data that is written later than or equal to the time specified by the <b>endTime mpMillis</b> parameter.</li> <li>If OTSSStream Reader has read all the incremental data, OTSSStream Reader stops reading data even before the time specified by the <b>endTimeStampMillis</b> parameter.</li> </ul>	No	None
<b>date</b>	The date when data is exported. The format is yyyyMMdd, for example, 20151111. You must specify this parameter or the <b>startTimestampMillis</b> and <b>endTimeStampMillis</b> parameters. For example, Alibaba Cloud Data Process Center schedules nodes by day. Therefore, the date parameter is provided.	No	None
<b>isExportSe quenceInfo</b>	Specifies whether to export time-series information. Time-series information includes the time when data is written. The default value is false, indicating that time series information is not exported.	No	false
<b>maxRetries</b>	The maximum number of retries of each request for reading incremental data from Table Store. The default value is 30. Retries are performed at certain intervals. The total time of 30 retries is approximately 5 minutes. Generally, you can use the default value.	No	30

Parameter	Description	Required	Default value
<b>startTimeString</b>	The left boundary of the time range (left-closed and right-open) of incremental data, measured in milliseconds in the format of <code>yyyymmddhh24miss</code> .	No	None
<b>endTimeString</b>	The right boundary of the time range (left-closed and right-open) of incremental data, measured in milliseconds in the format of <code>yyyymmddhh24miss</code> .	No	None
<b>mode</b>	The export mode. If this parameter is set to <code>single_version_and_update_only</code> , data is exported by row. By default, this parameter is not specified, and data is not exported by column.	No	None

### Configure OTSStream Reader by using the codeless UI

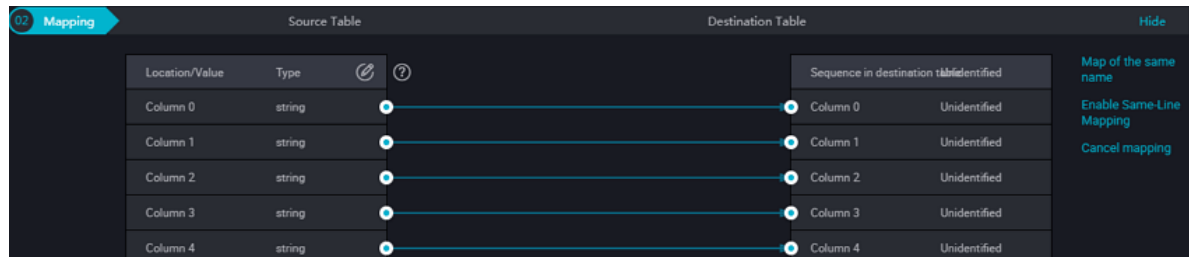
#### 1. Configure the connections.

Configure the source and destination connections for the sync node.

Parameter	Description
<b>Connection</b>	The <b>datasource</b> parameter in the preceding parameter description. Select a connection type, and enter the name of a connection that has been configured in DataWorks.
<b>Table</b>	The <b>table</b> parameter in the preceding parameter description.
<b>Start Timestamp</b>	The start time (included) in milliseconds of the incremental data.
<b>End Timestamp</b>	The end time (excluded) in milliseconds of the incremental data.
<b>State Table</b>	The name of the table for storing status records.
<b>Maximum Retries</b>	The <b>maxRetries</b> parameter in the preceding parameter description. Default value: 30.
<b>Export Time Information</b>	The <b>isExportSequenceInfo</b> parameter in the preceding parameter description. By default, Export Time Information is not selected.

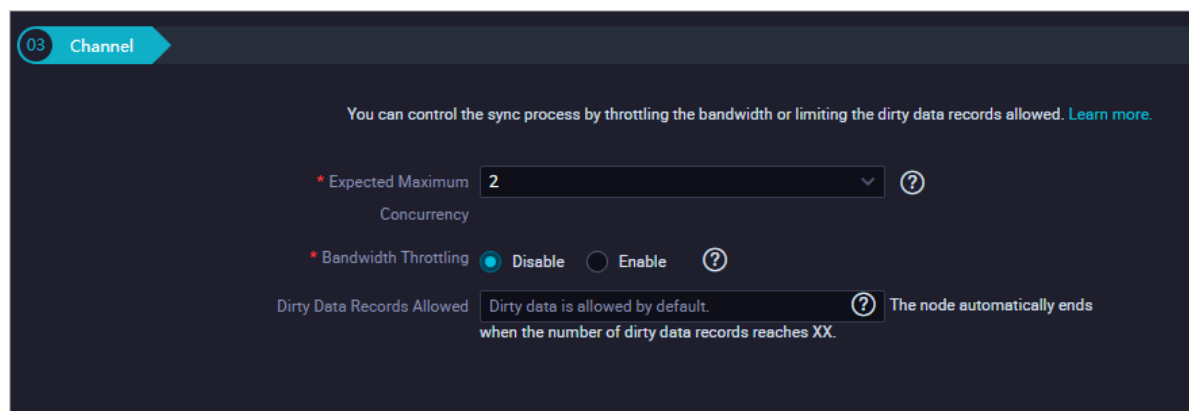
2. Configure field mapping, that is, the **column** parameter in the preceding parameter description.

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.



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

3. Configure channel control policies.



Parameter	Description
<b>Expected Maximum Concurrency</b>	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.

Parameter	Description
<b>Bandwidth Throttling</b>	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.
<b>Dirty Data Records Allowed</b>	The maximum number of dirty data records allowed.
<b>Resource Group</b>	The resource group used for running the sync node. If a large number of nodes including this sync node are deployed on the default resource group, the sync node may need to wait for resources. We recommend that you purchase an exclusive resource group for data integration or add a custom resource group. For more information, see <a href="#">#unique_15</a> and <a href="#">Add a custom resource group</a> .

### Configure OTSStream Reader by using the code editor

In the following code, a node is configured to read data from Table Store. 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 for
 storing status records.
 "maxRetries": 30, // The maximum number of retries on each request of reading
 incremental data from Table Store. It is set to 30 by default.
 "isExportSequenceInfo": false, // Specifies whether to export the time series
 information.
 "datasource": "${srcDatasource}", // The connection name.
 "startTimeString": "${startTime}", // The start time (included) of the incremental
 data.
 "table": "", // The name of the table to be synchronized.
 "endTimeString": "${endTime}" // The end time (excluded) of the incremental data
 },
 "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
 parameter to true.
 "concurrent": 1, // The maximum number of concurrent threads.
 }
},
"order": {
 "hops": [
 {
 "from": "Reader",
 "to": "Writer"
 }
]
}
}

```

## 8.2.28 Configure MetaQ Reader

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

Alibaba Cloud Message Queue is a professional message-oriented middleware developed by Alibaba Group. Based on technologies for building a highly available distributed cluster, Message Queue provides a complete set of cloud messaging services, including message subscription and publishing, message tracing, scheduled and delayed messages, resource statistics, and alerts. Message Queue provides asynchronous decoupling for distributed application systems and is suitable for Internet applications with large amounts of messages and high throughput. It is one of the core services used by Alibaba Group to support the Double 11 Shopping Festival.

MetaQ Reader reads real-time data from Message Queue through the Java SDK, converts the data to a format that is readable by Data Integration, and then sends the converted data to a writer.

### How it works

MetaQ Reader subscribes to the real-time data from Message Queue through the Java SDK of the following version:

```

<dependency>
 <groupId>com.taobao.metaq.final</groupId>
 <artifactId>metaq-client</artifactId>
 <version>4.0.1</version>
</dependency>
<dependency>
 <groupId>com.aliyun.openservices</groupId>
 <artifactId>ons-sdk</artifactId>
 <version>1.3.1</version>

```

&lt;/dependency&gt;

**Data types**

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

Data Integration data type	Message Queue data type
String	STRING

**Parameters**

Parameter	Description	Required
<b>accessId</b>	The AccessKey ID for accessing Message Queue.	Yes
<b>accessKey</b>	The AccessKey secret for accessing Message Queue.	Yes
<b>consumerId</b>	The consumer ID. A consumer, also known as a message subscriber, receives and consumes messages.  The consumer ID is the identifier of a type of consumers. The consumers with the same consumer ID generally receive and consume a type of messages, and use the same consumption logic.	Yes
<b>topicName</b>	The topic of the messages to be consumed. A topic is used to classify messages. It is the primary classifier.	Yes
<b>subExpression</b>	The subexpression used to filter messages of the topic.	Yes
<b>onsChannel</b>	The channel used for authentication when MetaQ Reader connects to Message Queue.	Yes



Parameter	Description	Required
<b>domainNam</b>	The endpoint for connecting to Message Queue.	Yes
<b>contentType</b>	The type of the message. Valid values: singlestringcolumn, text, and json.	Yes
<b>beginOffset</b>	The offset where the sync node starts to read data. Valid values: begin and lastRead.	Yes
<b>nullCurrentOffset</b>	The offset where the sync node starts to read data when the last offset is null. Valid values: begin and current.	Yes
<b>fieldDelimiter</b>	The column delimiter used to separate message strings, such as commas (,). Control characters are supported. Example: \u0001.	Yes
<b>column</b>	The columns to be synchronized.	Yes

### Configure MetaQ Reader by using the code editor

In the following code, a node is configured to read data from Message Queue.

```
{
 "job": {
 "content": [
 {
 "reader": {
 "name": "metaqreader",
 "parameter": {
 "accessId": "xxxxxxxxxxxx",
 "accessKey": "xxxxxxxxxxxxxxxx",
 "consumerId": "Test01",
 "topicName": "test",
 "subExpression": "*",
 "onsChannel": "ALIYUN",
 "domainName": "xxx.aliyun.com",
 "contentType": "singlestringcolumn",
 "beginOffset": "lastRead",
 "nullCurrentOffset": "begin",
 "fieldDelimiter": ",",
 "column": [
 "col0"
],
 "fieldDelimiter": ","
 }
 }
 }
]
 }
}
```

```
}
},
"writer": {
 "name": "streamwriter",
 "parameter": {
 "print": false
 }
}
}
]
}
}
```

## 8.2.29 Configure Hive Reader

This topic describes how Hive Reader works, the supported parameters, and how to configure it by using the code editor.

Hive is a Hadoop-based data warehouse tool used to process large amounts of structured logs. Hive maps structured data files to a table and allows you to run 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 a Hadoop Distributed File System (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 metadatabase, parses the configuration to obtain the file storage path, file format, and column delimiter of the HDFS file corresponding to the Hive table from which data is read, and reads data from the HDFS file.

The underlying logic of Hive Reader is the same as that of HDFS Reader. You can configure parameters of HDFS Reader in the parameters of Hive Reader. Data Integration transparently transmits the configured parameters to HDFS Reader.

**Parameters**

Parameter	Description	Required	Default value
<b>jdbcUrl</b>	<p>The Java Database Connectivity (JDBC) URL for connecting to the Hive metadatabase. Currently, Hive Reader can only access Hive metadatabases of the MySQL type.</p> <p>Make sure that the sync node is connected to the Hive metadatabase over a network and has access permissions on the metadatabase.</p>	Yes	None
<b>username</b>	The username for connecting to the Hive metadatabase.	Yes	None
<b>password</b>	The password for connecting to the Hive metadatabase.	Yes	None

Parameter	Description	Required	Default value
<b>column</b>	<p>The columns to read. Example: "column": ["id", "name"].</p> <ul style="list-style-type: none"> <li>Column pruning is supported. You can select and export specific columns.</li> <li>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.</li> <li>Constants are supported.</li> <li>The column parameter must explicitly specify a set of columns to be synchronized. The parameter cannot be left empty.</li> </ul>	Yes	None
<b>table</b>	<p>The name of the source Hive table.</p> <div>  <b>Note:</b> The name is case-sensitive.         </div>	Yes	None

Parameter	Description	Required	Default value
<b>partition</b>	<ul style="list-style-type: none"> <li>The partition in the source Hive table. This parameter is required for a partitioned Hive table. The sync node reads data from the partition specified by the <b>partition</b> parameter.</li> <li>This parameter is not required for a non-partitioned table.</li> </ul>	No	None

### Configure Hive Reader by using the codeless UI

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

### Configure Hive Reader by using the code editor

In the following code, a node is configured to read data from a Hive metadatabase.

```
{
 "order": {
 "hops": [
 {
 "from": "Reader",
 "to": "Writer"
 }
],
 "setting": {
 "errorLimit": {
 "record": "0"
 },
 "speed": {
 "concurrent": 1, // 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 parameter to
 true.
 }
 },
 "steps": [
 {
 "category": "reader",
 "name": "Reader",
 "parameter": {
```

```

"username": "", // The username for connecting to the Hive metadatabase.
"password": "", // The password for connecting to the Hive metadatabase.
"jdbcUrl": "jdbc:mysql://host:port/database", // The JDBC URL for connecting to the
Hive metadatabase.
"table": "", // The name of the source Hive table.
"partition": "", // The partition in the source Hive table. This parameter is required for a
partitioned Hive table.
"column": [
 "id",
 "name"
],
"stepType": "hive" // The reader type.
},
{
 "category": "writer",
 "name": "Writer",
 "parameter": {},
 "stepType": "stream"
}
],
"type": "job",
"version": "2.0", // The version number.
}

```

### 8.2.30 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.

Vertica Reader connects to a remote Vertica database through Java Database Connectivity (JDBC) and runs a SELECT statement to select and read data from the database.

#### How it works


Vertica Reader connects to a remote Vertica database through JDBC, generates a SELECT statement based on your configurations, and then sends the statement to the database. The Vertica database runs 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 through 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
<b>datasource</b>	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None
<b>jdbcUrl</b>	<p>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.</p> <p>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.</p> <div data-bbox="419 1272 1158 1431">  <b>Note:</b> The <b>jdbcUrl</b> parameter must be included in the connection parameter. </div> <p>The value of the <b>jdbcUrl</b> parameter must be in compliance with the standard format supported by Vertica. You can also specify the information of the attachment facility. Example: <code>jdbc:vertica://1*.0.0.1:3306/database</code>.</p>	No	None
<b>username</b>	The username for connecting to the Vertica database.	No	None
<b>table</b>	The name of the table to be synchronized.	Yes	None
<b>password</b>	The password for connecting to the Vertica database.	No	None

Parameter	Description	Required	Default value
<b>table</b>	<p>The name of the table to be synchronized. Vertica Reader can read data from multiple tables. The tables are described in a JSON array.</p> <p>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.</p> <div>  <b>Note:</b>            The <b>table</b> parameter must be included in the connection parameter.         </div>	Yes	None
<b>column</b>	<p>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.</p> <ul style="list-style-type: none"> <li>Column pruning is supported. You can select and export specific columns.</li> <li>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.</li> <li>Constants are supported.</li> <li>The <b>column</b> parameter must explicitly specify a set of columns to be synchronized. The parameter cannot be left empty.</li> </ul>	Yes	None



Parameter	Description	Required	Default value
<b>splitPk</b>	<p>The field used for data sharding when Vertica Reader extracts data. If you specify the <b>splitPk</b> 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.</p> <ul style="list-style-type: none"> <li>We recommend that you set the <b>splitPk</b> 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.</li> <li>Currently, the <b>splitPk</b> 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.</li> <li>If you leave the <b>splitPk</b> parameter empty, Vertica Reader extracts the data in the source table through a single thread.</li> </ul>	No	None
<b>where</b>	<p>The WHERE clause. Vertica Reader generates a SELECT statement based on the <b>column</b>, <b>table</b>, and <b>where</b> parameters that you have configured, and uses the generated SELECT statement to select and read data.</p> <p>For example, you can specify the <b>where</b> parameter during testing. To synchronize data generated on the current day, set the <b>where</b> parameter to <code>gmt_create &gt; \$bizdate</code>.</p> <ul style="list-style-type: none"> <li>You can use the <b>WHERE</b> clause to synchronize incremental data.</li> <li>If you do not specify the <b>where</b> parameter or leave it empty, all data is synchronized.</li> </ul>	No	None
<b>querySql</b>	<p>The <b>SELECT</b> statement used for refined data filtering. If you specify this parameter, Data Integration directly filters data based on this parameter.</p> <p>If you specify the <b>querySql</b> parameter, Vertica Reader ignores the <b>table</b>, <b>column</b>, and <b>where</b> parameters that you have configured.</p>	No	None

Parameter	Description	Required	Default value
<b>fetchSize</b>	<p>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.</p> <div>  <b>Note:</b>            A value greater than 2048 may lead to out of memory (OOM) during the data synchronization process.         </div>	No	1024

### Configure Vertica Reader by using the codeless UI

Currently, the codeless user interface (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
 parameter to true.
 "concurrent": 1,// The maximum number of concurrent threads.
 }
}
}

```

### 8.2.31 Configure SAP HANA Reader

SAP HANA is an in-memory computing platform that can be deployed on premises or on the cloud. It features high-performance data queries. You can directly query and analyze a large amount of real-time data without performing operations such as modeling and aggregation. This topic describes the parameters supported by SAP HANA Reader and how to configure it by using the code editor.

#### Parameters

Parameter	Description
<b>username</b>	The username for accessing SAP HANA.
<b>password</b>	The password for accessing SAP HANA.
<b>column</b>	The columns to be synchronized. Set the value to an asterisk (*) if all the columns in the source table are to be synchronized. .
<b>table</b>	The name of the table to be synchronized.
<b>jdbcUrl</b>	The JDBC URL for connecting to SAP HANA. Example: <b>jdbc:sap://127.0.0.1:30215? currentschema=TEST</b> .

Parameter	Description
<b>splitPk</b>	<p>The field used for data sharding when SAP HANA 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.</p> <p>Currently, the splitPk parameter supports data sharding only for integers. If no integer fields exist, do not specify this parameter.</p>

### Configure SAP HANA Reader by using the codeless UI

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

### Configure SAP HANA Reader by using the code editor

In the following code, a node is configured to synchronize data from SAP HANA to MaxCompute.

```
{
 "type": "job",
 "steps": [
 {
 "stepType": "saphana",
 "parameter": {
 "UserName": "Username",
 "Password": "Password",
 "column": [
 "Column 1",
 "Column 2",
 "Column 3"
],
 "connection": [
 {
 "table": [
 "The name of the table to be synchronized."
],
 "jdbcUrl": [
 "jdbc:sap://127.0.0.1:30215? currentschema=TEST"
]
 }
],
 "splitPk": "Column 1" // The field used for data sharding. If you specify the
splitPk parameter, the table is sharded based on the shard key specified by this
parameter.
 },
 "name": "Reader",
 "category": "reader"
 },
 {
 "stepType": "odps",
 "parameter": {
 "partition": "",
 "truncate": true,

```

```

 "datasource": "example", // The connection name.
 "column": [
 "*"
],
 "table": ""
 },
 "name": "Writer",
 "category": "writer"
}
],
"version": "2.0",
"order": {
 "hops": [
 {
 "from": "Reader",
 "to": "Writer"
 }
]
},
"setting": {
 "errorLimit": {
 "record": "" // The maximum number of dirty data records allowed.
 },
 "speed": {
 "concurrent": 2, // 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
 parameter to true.
 }
}
}

```

### 8.2.32 Gbase8a Reader

This topic describes the data types and parameters supported by Gbase8a Reader and how to configure it by using the code editor.

Gbase8a is a new type of column-oriented analytical database. Gbase8a Reader allows you to read data from Gbase8a databases.



#### Notice:

Currently, Gbase8a Reader only supports [Use exclusive resource groups for data integration](#) and does not support the default resource group or [custom resource groups](#).

Gbase8a Reader connects to a remote Gbase8a database through Java Database Connectivity (JDBC), generates a SELECT statement based on your configurations, and then sends the statement to the database. The Gbase8a database executes the statement and returns the result. Then, Gbase8a Reader assembles the returned data to abstract datasets in custom data types supported by Data Integration, and passes the datasets to a writer.


- Gbase8a Reader generates the SELECT statement based on the **table**, **column**, and **where** parameters that you set, and sends the generated SELECT statement to the Gbase8a database.
- If you specify the **querySql** parameter, Gbase8a Reader directly sends the value of this parameter to the Gbase8a database.


Gbase8a Reader accesses a Gbase8a database through the MySQL database driver. You need to confirm the compatibility between the driver version and your Gbase8a database. Gbase8a Reader uses the following version of the MySQL database driver:

```
<dependency>
 <groupId>mysql</groupId>
 <artifactId>mysql-connector-java</artifactId>
 <version>5.1.22</version>
</dependency>
```

### Parameters


Parameter	Description	Required	Default value
<b>datasource</b>	The name of the connection. If the edition of the DataWorks service that you activated supports Gbase8a connections, you can add a Gbase8a connection and reference the connection in this parameter.  You can connect to the Gbase8a database based on the settings of the <b>jdbcUrl</b> or <b>username</b> parameter.	No	None

Parameter	Description	Required	Default value
<b>jdbcUrl</b>	<p>The JDBC URL for connecting to the Gbase8a database. You can specify multiple JDBC URLs in a JSON array for a database.</p> <p>If you specify multiple JDBC URLs, Gbase8a Reader verifies the connectivity of the URLs in sequence to find a valid URL.</p> <p>If no URL is valid, Gbase8a Reader returns an error.</p> <div>  <b>Note:</b>            The <b>jdbcUrl</b> parameter must be included in the <b>connection</b> parameter.         </div> <p>The value of the <b>jdbcUrl</b> parameter must be in compliance with the standard format supported by Gbase8a. You can also specify the information of the attachment facility.            Example: jdbc:mysql://127.0.0.1:3306/database. You must specify the jdbcUrl parameter or the <b>datasource</b> parameter.</p>	No	None
<b>username</b>	The username for connecting to the Gbase8a database.	No	None
<b>password</b>	The password for connecting to the Gbase8a database.	No	None

Parameter	Description	Required	Default value
<b>table</b>	<p>The name of the source table from which Gbase8a Reader reads data. Gbase8a Reader can read data from multiple tables. The tables are described in a JSON array.</p> <p>If you specify multiple tables, make sure that the tables have the same schema. Gbase8a Reader does not check whether the tables have the same schema.</p> <div>  <b>Note:</b>            The <b>table</b> parameter must be included in the <b>connection</b> parameter.         </div>	Yes	None
<b>column</b>	<p>The columns that Gbase8a Reader reads from the source table. The columns are described in a JSON array. The default value is [ * ], which indicates all columns in the source table.</p> <ul style="list-style-type: none"> <li>Column pruning is supported. You can specify specific columns for Gbase8a Reader to export.</li> <li>The column order can be changed. You can configure Gbase8a Reader to export columns in an order different from that specified in the schema of the table.</li> <li>Constants are supported. Example: '123'.</li> <li>Functions are supported. Example: date('now').</li> <li>The <b>column</b> parameter must explicitly specify a set of columns to read. The parameter cannot be left empty.</li> </ul>	Yes	None



Parameter	Description	Required	Default value
<b>splitPk</b>	<p>The field used for data sharding when Gbase8a Reader reads data. If you specify the <b>splitPk</b> 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.</p> <ul style="list-style-type: none"> <li>We recommend that you set the <b>splitPk</b> 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.</li> <li>Currently, the <b>splitPk</b> parameter supports data sharding only for integers but not for other data types such as strings, floating points, and dates. If you specify this parameter to a column of an unsupported type, Gbase8a Reader ignores the <b>splitPk</b> parameter and reads data through a single thread.</li> <li>If you leave the <b>splitPk</b> parameter empty, Gbase8a Reader reads data in the source table through a single thread.</li> </ul>	No	None
<b>where</b>	<p>The WHERE clause. Gbase8a Reader generates a SELECT statement based on the <b>column</b>, <b>table</b>, and <b>where</b> parameters that you set, and uses the generated SELECT statement to select and read data.</p> <p>For example, you can set the <b>where</b> parameter to <b>limit 10</b> during testing. To read data generated on the current day, set the <b>where</b> parameter to <b>gmt_create &gt; \$bizdate</b>.</p> <ul style="list-style-type: none"> <li>You can use the <b>WHERE</b> clause to read incremental data.</li> <li>If you do not specify the <b>where</b> parameter or leave it empty, all data is read.</li> </ul>	No	None

Parameter	Description	Required	Default value
<b>querySql</b>	<p>The <b>SELECT</b> statement used for refined data filtering. If you specify this parameter, Data Integration directly filters data based on this parameter.</p> <p>If you specify the <b>querySql</b> parameter, Gbase8a Reader ignores the <b>table</b>, <b>column</b>, <b>where</b>, and <b>splitPk</b> parameters that you set.</p>	No	None
<b>fetchSize</b>	<p>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.</p> <div>  <b>Note:</b>            A value greater than 2048 may lead to out of memory (OOM) during the data reading process.         </div>	No	1,024

### Configure Gbase8a Reader by using the codeless UI

Currently, the codeless user interface (UI) is not supported for Gbase8a Reader.

### Configure Gbase8a Reader by using the code editor

The following example provides sample code for configuring Gbase8a Reader. For more information, see [Create a sync node by using the code editor](#).

```
{
 "type": "job",
 "steps": [
 {
 "stepType": "gbase8a", // The reader name.
 "parameter": {
 "datasource": "", // The connection name.
 "username": "",
 "password": "",
 "where": "",
 "column": [// The columns to be read.
 "id",
 "name"
],
 "splitPk": "id",
 "connection": [
 {
 "table": [// The name of the table whose data is to be read.
 "table"
]
 }
]
 }
 }
]
}
```

```

 "jdbcUrl": [
 "jdbc:mysql://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. The value
 false indicates that the bandwidth is not throttled. The value 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.
 }
}
}
}

```

### 8.2.33 Configure Datahub Reader

Datahub is a real-time data distribution platform designed to process streaming data. You can publish and subscribe 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 Reader reads data from Datahub through the Java SDK of the following version:


```


<dependency>
 <groupId>com.aliyun.datahub</groupId>
 <artifactId>aliyun-sdk-datahub</artifactId>
 <version>2.9.1</version>

```

&lt;/dependency&gt;

**Parameters**

Parameter	Description	Required
<b>endpoint</b>	The endpoint of Datahub.	Yes
<b>accessId</b>	The AccessKey ID for accessing Datahub.	Yes
<b>accessKey</b>	The AccessKey secret for accessing Datahub.	Yes
<b>project</b>	The name of the source Datahub project. A project is the resource management unit in Datahub for resource isolation and control.	Yes
<b>topic</b>	The source topic of Datahub.	Yes
<b>batchSize</b>	The number of data records to read at a time. Default value: 1024.	No
<b>beginDateTime</b>	<p>The start time of data consumption. This parameter defines the left boundary of an interval (left-closed and right-open) in the format of <b>yyyyMMddHHmmss</b>. The parameter can work with the scheduling time parameter in DataWorks.</p> <div>  <b>Note:</b>  Specify the <b>beginDateTime</b> and <b>endDateTime</b> parameters to determine the time range for consuming data. </div>	Yes

Parameter	Description	Required
<b>endTime</b>	<p>The end time of data consumption. This parameter defines the right boundary of an interval (left-closed and right-open) in the format of <b>yyyyMMddHHmmss</b>. The parameter can work with the scheduling time parameter in DataWorks.</p> <div>  <b>Note:</b>  Specify the <b>beginDateTime</b> and <b>endTime</b> parameters to determine the time range for consuming data. </div>	Yes

### Configure Datahub Reader by using the codeless UI

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

### Configure Datahub Reader by using the code editor

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

```
{
 "job": {
 "content": [
 {
 "reader": {
 "name": "loghubreader",
 "parameter": {
 "endpoint": "xxx" // The endpoint of Datahub.
 "accessId": "xxx", // The AccessKey ID for accessing Datahub.
 "accessKey": "xxx", // The AccessKey secret for accessing Datahub.
 "project": "xxx", // The name of the source Datahub project.
 "topic": "xxx" // The source Datahub topic.
 "batchSize": 1000, // The number of data records to read at a time.
 "beginDateTime": "20180910111214", // The start time of data consumption.
 "endTime": "20180910111614", // The end time of data consumption.
 "column": [
 "col0",
 "col1",
 "col2",
 "col3",
 "col4"
]
 }
 },
 "writer": {
```

```
 "name": "streamwriter",
 "parameter": {
 "print": false
 }
 }
}
]
```

### 8.2.34 Configure ApsaraDB for OceanBase Reader

ApsaraDB for OceanBase is a financial-grade distributed relational database developed by Alibaba Cloud and Ant Financial. This topic describes how ApsaraDB for OceanBase Reader works, the supported parameters, and how to configure it by using the code editor.

**Notice:**

Currently, ApsaraDB for OceanBase Reader only supports exclusive resource groups and does not support the default resource group or custom resource groups. For more information, see [Use exclusive resource groups for data integration](#) and [Add a custom resource group](#).

#### Background

ApsaraDB for OceanBase implements automated and non-disruptive disaster recovery across cities with the Five Data Centers in Three Regions solution. It provides high availability for financial services through conventional hardware. With the online scaling capability, ApsaraDB for OceanBase is a Chinese database that has undergone strict verification in terms of functionality, stability, scalability, and performance.

ApsaraDB for OceanBase Reader allows you to read data from ApsaraDB for OceanBase in Oracle or MySQL mode.

ApsaraDB for OceanBase Reader connects to a remote ApsaraDB for OceanBase database through Java Database Connectivity (JDBC), and runs a SELECT statement to select and read data from the ApsaraDB for OceanBase database.

ApsaraDB for OceanBase Reader connects to a remote ApsaraDB for OceanBase database through the Java client, generates a SELECT statement based on your configurations, and then sends the statement to the database. The ApsaraDB for OceanBase database runs the statement and returns the result. Then, ApsaraDB for OceanBase Reader assembles the returned data to abstract datasets in custom data types supported by Data Integration, and sends the datasets to a writer.

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

**Note:**


ApsaraDB for OceanBase supports the Oracle and MySQL modes. The WHERE clause and columns that contain functions must be in compliance with the SQL syntax supported by Oracle or MySQL. Otherwise, the SQL statement may fail to be run.

ApsaraDB for OceanBase Reader accesses an ApsaraDB for OceanBase database through the OceanBase driver. Confirm the compatibility between the driver version and your ApsaraDB for OceanBase database. ApsaraDB for OceanBase Reader uses the following version of the OceanBase database driver:


```
<dependency>
 <groupId>com.alipay.OceanBase</groupId>
 <artifactId>OceanBase-connector-java</artifactId>
 <version>3.1.0</version>
</dependency>
```

**Parameters**


Parameter	Description	Required	Default value
<b>datasource</b>	The connection name. It must be identical to the name of the ApsaraDB for OceanBase connection that you added in DataWorks.  You can connect to the ApsaraDB for OceanBase database based on the settings of the <b>jdbcUrl</b> or <b>username</b> parameter.	No	None

Parameter	Description	Required	Default value
<b>jdbcUrl</b>	<p>The JDBC URL for connecting to the ApsaraDB for OceanBase database. You can specify multiple JDBC URLs for a database. The JDBC URLs are described in a JSON array.</p> <p>If you specify multiple JDBC URLs, ApsaraDB for OceanBase Reader verifies the connectivity of the URLs in sequence to find a valid URL.</p> <p>If no URL is valid, ApsaraDB for OceanBase Reader returns an error.</p> <div>  <b>Note:</b>            The <b>jdbcUrl</b> parameter must be included in the <b>connection</b> parameter.         </div> <p>The value of the <b>jdbcUrl</b> parameter must be in compliance with the standard format supported by ApsaraDB for OceanBase. You can also specify the information of the attachment facility. Example: <code>jdbc:OceanBase://127.0.0.1:3306/database</code>. One of the <b>jdbcUrl</b> parameter and the <b>datasource</b> parameter must be specified.</p>	No	None
<b>username</b>	The username for connecting to the ApsaraDB for OceanBase database.	No	None
<b>password</b>	The password for connecting to the ApsaraDB for OceanBase database.	No	None



Parameter	Description	Required	Default value
<b>table</b>	<p>The name of the table to be synchronized. ApsaraDB for OceanBase Reader can read data from multiple tables. The tables are described in a JSON array.</p> <p>If you specify multiple tables, make sure that the tables have the same schema. ApsaraDB for OceanBase Reader does not check whether the tables have the same schema.</p> <div>  <b>Note:</b>            The <b>table</b> parameter must be included in the <b>connection</b> parameter.         </div>	Yes	None
<b>column</b>	<p>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.</p> <ul style="list-style-type: none"> <li>Column pruning is supported. You can select and export specific columns.</li> <li>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.</li> <li>Constants are supported. Example: '123'.</li> <li>Functions are supported. Example: date('now').</li> <li>The <b>column</b> parameter must explicitly specify a set of columns to be synchronized. The parameter cannot be left empty.</li> </ul>	Yes	None

Parameter	Description	Required	Default value
<b>splitPk</b>	<p>The column used for data sharding when ApsaraDB for OceanBase Reader extracts data. If you specify the <b>splitPk</b> 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.</p> <ul style="list-style-type: none"> <li>We recommend that you set the <b>splitPk</b> 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.</li> <li>Currently, the <b>splitPk</b> 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 value of an unsupported type, ApsaraDB for OceanBase Reader returns an error.</li> <li>If you leave the <b>splitPk</b> parameter empty, ApsaraDB for OceanBase Reader extracts the data in the source table through a single thread.</li> </ul>	No	None
<b>where</b>	<p>The WHERE clause. ApsaraDB for OceanBase Reader generates a SELECT statement based on the <b>table</b>, <b>column</b>, and <b>where</b> parameters that you have configured, and uses the generated SELECT statement to select and read data.</p> <p>For example, you can set the <b>where</b> parameter to <b>limit 10</b> during testing. To synchronize data generated on the current day, set the <b>where</b> parameter to <b>gmt_create &gt; \$bizdate</b>.</p> <ul style="list-style-type: none"> <li>You can use the <b>WHERE</b> clause to synchronize incremental data.</li> <li>If you do not specify the <b>where</b> parameter or leave it empty, all data is synchronized.</li> </ul>	No	None

Parameter	Description	Required	Default value
<b>querySql</b>	<p>The <b>SELECT</b> statement used for refined data filtering. If you specify this parameter, Data Integration directly filters data based on this parameter.</p> <p>If you specify the <b>querySql</b> parameter, ApsaraDB for OceanBase Reader ignores the <b>table</b>, <b>column</b>, <b>where</b>, and <b>splitPk</b> parameters that you have configured.</p>	No	None
<b>fetchSize</b>	<p>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.</p> <div>  <b>Note:</b>  A value greater than 2048 may lead to out of memory (OOM) during the data synchronization process. </div>	No	1024

### Configure ApsaraDB for OceanBase Reader by using the codeless UI

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

### Configure ApsaraDB for OceanBase Reader by using the code editor

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

```
{
 "type": "job",
 "steps": [
 {
 "stepType": "apsaradb_for_OceanBase", // The reader type.
 "parameter": {
 "datasource": "", // The connection name.
 "where": "",
 "column": [// The columns to be synchronized.
 "id",
 "name"
],
 "splitPk": ""
 },
 "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
 parameter to true.
 "concurrent": 1,// The maximum number of concurrent threads.
 }
 }
}

```

### 8.2.35 Hologres Reader

Hologres Reader allows you to read data from Hologres. You can use Hologres Reader to read data from the tables of Hologres data stores and write the data to other types of data stores.



#### Notice:

Currently, you can only use exclusive resource groups for Data Integration for Hologres Reader. The default resource group and custom resource groups are not supported. For more information, see [Use exclusive resource groups for data integration](#) and [Add a custom resource group](#).

Hologres Reader reads data from Hologres tables by using PostgreSQL statements. The number of concurrent threads is determined based on the number of shards in the source Hologres table. Each shard corresponds to one SELECT statement.


- In a CREATE TABLE statement that is used to create a table in Hologres, the `CALL set_table_property('table_name', 'shard_count', 'xx')` command sets the number of table shards.

By default, the `shard_count` field is set to the default number of table shards of the database created in Hologres. The amount of computing resources for the relevant Hologres instance determines the default number of table shards.

- A SELECT statement uses the shard specifies by the built-in field `hg_shard_id` of the source Hologres table to query data.

## Parameters

Parameter	Description	Required	Default value
<b>endpoint</b>	<p>The <b>endpoint</b> used to connect to the source Hologres instance, in the format of <code>instance-id-region-endpoint.hologres.aliyuncs.com:port</code>. You can view the endpoints of a Hologres instance on the configuration page of the instance in the Hologres console.</p> <p>The <b>endpoint</b> of a Hologres instance varies with the network types, including the classic network, Internet, and Virtual Private Cloud (VPC). Select an appropriate <b>endpoint</b> based on the network where the resource group for Data Integration and the Hologres instance reside. Otherwise, the connection may fail or the performance may be poor. The formats of these three types of endpoints are as follows:</p> <ul style="list-style-type: none"> <li>Classic network endpoint: <code>instance-id-region-endpoint-internal.hologres.aliyuncs.com:port</code></li> <li>Public endpoint: <code>instance-id-region-endpoint.hologres.aliyuncs.com:port</code></li> <li>VPC endpoint: <code>instance-id-region-endpoint-vpc.hologres.aliyuncs.com:port</code></li> </ul> <p>We recommend that you deploy the resource group for Data Integration and the Hologres instance in the same zone of the same region to guarantee network connection and optimal performance.</p>	Yes	None
<b>accessId</b>	The <b>AccessKey ID</b> of the account used to access Hologres.	Yes	None
<b>accessKey</b>	The <b>AccessKey secret</b> of the account used to access Hologres. Specify an AccessKey secret of an account that is authorized to read data from the source table.	Yes	None

Parameter	Description	Required	Default value
<b>database</b>	The name of the source database in the Hologres instance.	Yes	None
<b>table</b>	The name of the source Hologres table. If the table is a partitioned table, specify the name of the parent partitioned table.	Yes	None
<b>column</b>	The columns in the source Hologres table from which data is read. The primary key columns of the source table must be included. Set the value to an asterisk (*) if data is read from all the columns in the source table. That is, set the column parameter as follows: "column": ["*"].	Yes	None
<b>partition</b>	<p>The partition key column and the corresponding value of the source Hologres table, in the format of column=value. This parameter is valid for partitioned tables.</p> <div>  <b>Notice:</b> <ul style="list-style-type: none"> <li>Currently, Hologres only supports list partitioning and you can only specify a single column as the partition key column. The data type of the partition key column must be INT4 or TEXT.</li> <li>The parameter value must match the partition expression in the data definition language (DDL) statements used to create the source table.</li> <li>The corresponding child partitioned tables must have been created with data.</li> </ul> </div>	No	Null, indicating a non-partitioned table.
<b>fetchSize</b>	The number of data records to be read from the source Hologres table at a time by using the SELECT statement.	No	1,000

## Configure Hologres Reader by using the codeless UI

### 1. Configure the connections.

Configure the source and destination connections for the sync node.

Parameter	Description
<b>Connection</b>	The name of the connection.
<b>Table</b>	The <b>table</b> parameter in the preceding parameter description.
<b>Filter</b>	The filter condition for the data to be synchronized. Currently, the <b>WHERE</b> clause is not supported. The SQL syntax is determined by the selected connection.

### 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.

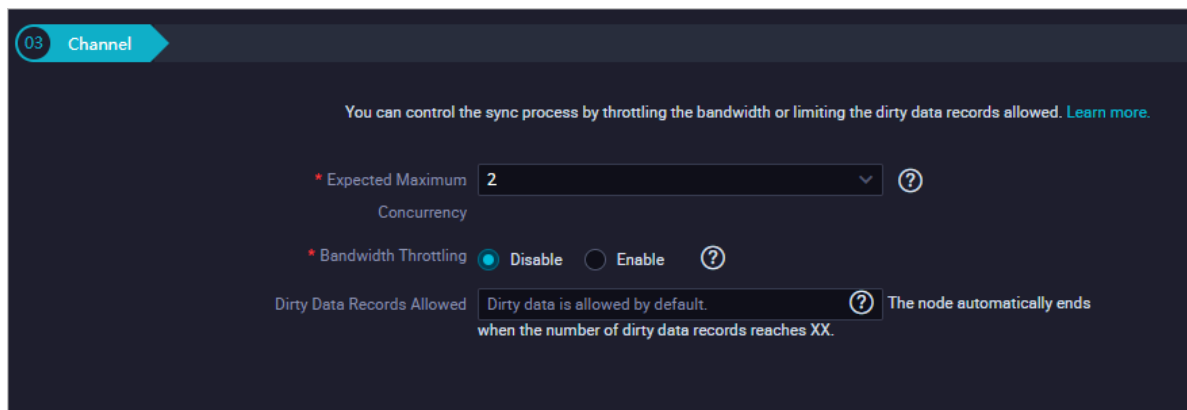
02 Mappings			Source Table	Target Table	Hide
Field	Type		Field	Type	
bizdate	DATE	•	age	BIGINT	•
region	VARCHAR	•	job	STRING	•
pv	BIGINT	•	marital	STRING	•
uv	BIGINT	•	education	STRING	•
browse_size	BIGINT	•	default	STRING	•
Add +			housing	STRING	
					Map Fields with the Same Name Map Fields in the Same Line Delete All Mappings Auto Layout

GUI Element	Description
<b>Map Fields with the Same Name</b>	Click <b>Map Fields with the Same Name</b> to establish a mapping between fields with the same name. Note that the data types of the fields must match.
<b>Map Fields in the Same Line</b>	Click <b>Map Fields in the Same Line</b> to establish a mapping for fields in the same row. Note that the data types of the fields must match.
<b>Delete All Mappings</b>	Click <b>Delete All Mappings</b> to remove mappings that have been established.
<b>Auto Layout</b>	Click Auto Layout. The fields are automatically sorted based on specified rules.



GUI Element	Description
<b>Change Fields</b>	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.
<b>Add</b>	<ul style="list-style-type: none"> <li>Click Add to add a field. You can enter constants. Each constant must be enclosed in single quotation marks (' '), such as <b>'abc'</b> and <b>'123'</b>.</li> <li>You can use scheduling parameters, such as <b>\${bizdate}</b>.</li> <li>You can enter functions supported by relational databases, such as <b>now()</b> and <b>count(1)</b>.</li> <li>Fields that cannot be parsed are indicated by Unidentified.</li> </ul>

### 3. Configure channel control policies.



Parameter	Description
<b>Expected Maximum Concurrency</b>	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).
<b>Bandwidth Throttling</b>	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.
<b>Dirty Data Records Allowed</b>	The maximum number of dirty data records allowed.

## Configure Hologres Reader by using the code editor

- Use a non-partitioned table as the source table
- In the following code, a node is configured to read data from a non-partitioned Hologres table.

```
{
 "type": "job",
 "version": "2.0", // The version number.
 "steps": [
 {
 "stepType": "holo", // The reader type.
 "parameter": {
 "endpoint": "instance-id-region-endpoint.hologres.aliyuncs.com:port",
 "accessId": "*****", // The AccessKey ID of the account used to
 access Hologres.
 "accessKey": "*****", // The AccessKey secret of the account
 used to access Hologres.
 "database": "postgres",
 "table": "holo_reader_****",
 "column": [// The columns to be synchronized.
 "tag",
 "id",
 "title"
]
 },
 "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 parameter to true.
 "concurrent": 1, // The maximum number of concurrent threads.
 }
 },
 "order": {
 "hops": [
 {
 "from": "Reader",
 "to": "Writer"
 }
]
 }
}
```

```
}
```

- The following section provides the sample DDL statements used to create a non-partitioned Hologres table.

```
begin;
drop table if exists holo_reader_basic_src;
create table holo_reader_basic_src(
 tag text not null,
 id int not null,
 title text not null,
 body text,
 primary key (tag, id));
call set_table_property('holo_reader_basic_src', 'orientation', 'column');
call set_table_property('holo_reader_basic_src', 'shard_count', '3');
commit;
```

- Use a partitioned table as the source table
  - In the following code, a node is configured to read data from a child partitioned table in Hologres in the SDK (Fast Write) mode.

**Note:**

Exercise caution when you set the **partition** parameter.

```
{
 "type": "job",
 "version": "2.0", // The version number.
 "steps": [
 {
 "stepType": "holo", // The reader type.
 "parameter": {
 "endpoint": "instance-id-region-endpoint.hologres.aliyuncs.com:port",
 "accessId": "*****", // The AccessKey ID of the account used to
 access Hologres.
 "accessKey": "*****", // The AccessKey secret of the account
 used to access Hologres.
 "database": "postgres",
 "table": "holo_reader_basic_****",
 "partition": "tag=foo",
 "column": [
 ""
],
 "fetchSize": "100"
 },
 "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 parameter to true.
 "concurrent":1,// The maximum number of concurrent threads.
 }
 },
 "order":{
 "hops":[
 {
 "from":"Reader",
 "to":"Writer"
 }
]
 }
}

```

- The following section provides the sample DDL statements used to create a partitioned Hologres table.

```

begin;
drop table if exists holo_reader_basic_part_src;
create table holo_reader_basic_part_src(
 tag text not null,
 id int not null,
 title text not null,
 body text,
 primary key (tag, id))
partition by list(tag);
call set_table_property('holo_reader_basic_part_src', 'orientation', 'column');
call set_table_property('holo_reader_basic_part_src', 'shard_count', '3');
commit;

create table holo_reader_basic_part_src_1583161774228 partition of holo_reader_basic_part_src for values in ('foo');

Make sure that the child partitioned table has been created with proper data.
postgres=# \d+ holo_reader_basic_part_src
 Table "public.holo_reader_basic_part_src"
 Column | Type | Collation | Nullable | Default | Storage | Stats target | Description
-----+-----+-----+-----+-----+-----+-----+-----
tag | text | | not null | | extended | |
id | integer| | not null | | plain | |
title | text | | not null | | extended | |
body | text | | | | extended | |
Partition key: LIST (tag)
Indexes:
 "holo_reader_basic_part_src_pkey" PRIMARY KEY, btree (tag, id)

```

```
Partitions: holo_reader_basic_part_src_1583161774228 FOR VALUES IN ('foo')
```

## 8.2.36 TSDB Reader

TSDB Reader allows you to read data from Time Series Database (TSDB). This topic describes the data types and parameters supported by TSDB Reader and how to configure it by using the code editor.



### Notice:

Currently, you can only use exclusive resource groups for Data Integration for TSDB Reader. The default resource group and custom resource groups are not supported. For more information, see [Use exclusive resource groups for data integration](#) and [Add a custom resource group](#).

### Background

TSDB is a high-performance, cost-effective, stable, and reliable online time series database service. It features high read/write performance and high compression ratio for data storage. It also enables the interpolation and aggregation of time series data. TSDB is widely applied in Internet of Things (IoT) and Internet. It can be used to monitor devices and business and send alerts in real time. For more information, see [What is TSDB](#).

TSDB Reader connects to a TSDB instance through an HTTP request and obtains data points through the `/api/query` or `/api/mquery` HTTP API operation. For more information, see [HTTP API overview](#). A sync node is split to multiple export tasks based on the time series and time range.

### Limits

- If a metric reports a data volume larger than the specified threshold for an hour, you need to change the value of the `-j` parameter to adjust the memory size of the Java virtual machine (JVM).

If the synchronization speed of the downstream writer is lower than that of TSDB Reader, the task backlog may occur. You need to change the memory size of the JVM to an

appropriate value. For example, run the following command to extract data from an Alibaba Cloud TSDB database to a local data store with the JVM memory size specified:

```
python datax/bin/datax.py tsdb2stream.json -j "-Xms4096m -Xmx4096m"
```

- 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 is converted to **[3:00, 4:00)**.

## Data types



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

Data Integration data type	TSDB data type
STRING	String to which a data point in TSDB is serialized, including the timestamp, metric, tags, fields, and value

## Parameters

Parameter	Description	Required	Default value
<b>name</b>	The name of the reader.	Yes	tsdbreader
<b>sinkDbType</b>	<p>The type of the destination database.</p> <p>Valid values: TSDB and RDB.</p> <ul style="list-style-type: none"> <li>A value of TSDB indicates that the destination data store is an Alibaba Cloud TSDB, OpenTSDB, InfluxDB, Prometheus, or Timescale database.</li> <li>A value of RDB indicates that the destination data store is a relational database such as an AnalyticDB for MySQL, Oracle, MySQL, PostgreSQL, or Distributed Relational Database Service (DRDS) database.</li> </ul>	No	TSDB
<b>endpoint</b>	The HTTP endpoint for connecting to the source TSDB database. Specify the endpoint in the http://IP address:Port format.	Yes	None

Parameter	Description	Required	Default value
<b>column</b>	<p>The columns to be synchronized. The value of this parameter varies with the type of the destination database.</p> <ul style="list-style-type: none"> <li>• TSDB: The metrics to be synchronized.</li> <li>• RDB: The tags to be synchronized and the <b>__metric__</b>, <b>__ts__</b>, and <b>__value__</b> fields.</li> </ul> <p>The <b>__metric__</b> field is added to synchronize metrics and the <b>__ts__</b> field is added to synchronize <b>timestamps</b>. If you want to synchronize single-value data, the <b>__value__</b> field is added to synchronize values. To synchronize multi-value data, specify the <b>field</b> parameter.</p>	Yes	None
<b>metric</b>	The metrics to be synchronized. The parameter is valid only when the destination data store is a relational database.	No	None
<b>field</b>	The fields to be synchronized. This parameter is valid only for multi-value data synchronization.	No	None
<b>tag</b>	The tags to be synchronized, in the format of key-value pairs. These tags are used to filter time series.	No	None
<b>splitIntervalMs</b>	The interval of collecting data to be synchronized. An export task is split to multiple subtasks to collect data based on the interval. Unit: milliseconds.	Yes	None

Parameter	Description	Required	Default value
<b>beginDateTime</b>	<p>The start time of the time range of the data points to be synchronized, in the format of yyyy-MM-dd HH:mm:ss. Specify the <b>beginDateTime</b> and <b>endDateTime</b> parameters to determine the time range of the data points to be synchronized.</p> <div>  <b>Note:</b>            The start time and end time of the time range are automatically converted to on-the-hour time. For example, if you set the time range to <b>[3:35, 4:55)</b> on April 18, 2019, the time range is converted to <b>[3:00, 4:00)</b>.         </div>	Yes	None
<b>endDateTime</b>	<p>The end time of the time range of the data points to be synchronized, in the format of yyyy-MM-dd HH:mm:ss. Specify the <b>beginDateTime</b> and <b>endDateTime</b> parameters to determine the time range of the data points to be synchronized.</p> <div>  <b>Note:</b>            The start time and end time of the time range are automatically converted to on-the-hour time. For example, if you set the time range to <b>[3:35, 4:55)</b> on April 18, 2019, the time range is converted to <b>[3:00, 4:00)</b>.         </div>	Yes	None

### Configure TSDB Reader by using the codeless UI

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

### Configure TSDB Reader by using the code editor

For more information about how to use the code editor, see [Create a sync node by using the code editor](#).



- In the following code, a node is configured to read data from an Alibaba Cloud TSDB database and write time series data to a local data store.

Sample time series data:

```
{"metric":"m","tags":{"app":"a19","cluster":"c5","group":"g10","ip":"i999","zone":"z1"},"timestamp":1546272263,"value":1}
```

Sample code:

```
{
 "type":"job",
 "version":"2.0",// The version number.
 "steps":[
 {
 "stepType":"tsdb",
 "parameter":{
 "sinkDbType": "TSDB", // The type of the destination database.
 "endpoint": "http://localhost:8242",
 "column": [
 "m"
],
 "splitIntervalMs": 60000,
 "beginDateTime": "2019-01-01 00:00:00",
 "endDateTime": "2019-01-01 01:00:00"
 },
 "name":"Reader",
 "category":"reader"
 },
 {
 "stepType":"stream",
 "parameter":{
 "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.
 },
 "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"
 }
]
 }
}
```

```
}
```

- In the following code, a node is configured to read data from an Alibaba Cloud TSDB database and write relational data to a local data store.

Sample relational data:

```
m 1546272125 a1 c1 g2 i3021 z4 1.0
```

Sample code:

```
{
 "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": 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"
 }
]
 }
 }
}

```

- In the following code, a node is configured to read data from an Alibaba Cloud TSDB database and write single-value 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":{
 "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"
 }
]
 }
}

```

- In the following code, a node is configured to read data from an Alibaba Cloud TSDB database and write multi-value 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",
 "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": 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"
 }
]
}
}
}

```

- In the following code, a node is configured to read data from an Alibaba Cloud TSDB database and write single-value data to an AnalyticDB for MySQL database. In addition, tags are specified to filter time series.

```

{
 "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"
],
 "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": 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"
 }
]
}
}

```

```
}
```

- In the following code, a node is configured to read data from an Alibaba Cloud TSDB database and write multi-value data to an AnalyticDB for MySQL database. In addition, tags are specified to filter time series.

```
{
 "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",
 "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",
 "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": 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"
 }
]
}
}
}

```

- In the following code, a node is configured to read data from an Alibaba Cloud TSDB database and write single-value 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",
 "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": 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"
 }
]
 }
}

```

- In the following code, a node is configured to read data from an Alibaba Cloud TSDB database and write multi-value 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",
 "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": 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"
 }
]
}
}
```

## 8.3 Writer configuration

### 8.3.1 Configure Datahub Writer

This topic describes the 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 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 Object Storage Service (OSS).

**Note:**


Strings can only be UTF-8 encoded. The size of each string must not exceed 1 MB.

## Channel types

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** and **file**. The following configuration sets the channel type to file:

```
"agent.sinks.dataXSinkWrapper.channel": "file"
```

## Parameters

Parameter	Description	Required	Default value
<b>accessId</b>	The AccessKey ID for accessing Datahub.	Yes	None
<b>accessKey</b>	The AccessKey secret for accessing Datahub.	Yes	None
<b>endpoint</b>	The endpoint of Datahub.	Yes	None
<b>maxRetryCount</b>	The maximum number of retries if the sync node fails.	No	None
<b>mode</b>	The mode for writing strings.	Yes	None
<b>parseContent</b>	The data to be parsed.	Yes	None
<b>project</b>	<p>The organizational unit in Datahub. Each project contains one or more topics.</p> <div>  <b>Note:</b>            Datahub projects are independent from MaxCompute projects. Projects created in MaxCompute cannot be used in Datahub.         </div>	Yes	None
<b>topic</b>	The minimum unit for data subscription and publication. You can use topics to distinguish different types of streaming data.	Yes	None
<b>maxCommitSize</b>	The amount of data 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
<b>batchSize</b>	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	1024

Parameter	Description	Required	Default value
<b>maxCommitInterval</b>	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	30000
<b>parseMode</b>	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 user interface (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": [
 // The following template is used to configure Stream Reader. For more information
 , see the corresponding topic.
 {
 "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 that Datahub Writer buffers
 before sending it to the destination.
 },
 "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,// 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.
}
},
"order": {
 "hops": [
 {
 "from": "Reader",
 "to": "Writer"
 }
]
}
}
```

### 8.3.2 Configure Db2 Writer

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

Db2 Writer allows you to write data to tables stored in Db2 databases. Specifically, Db2 Writer connects to a remote Db2 database through Java Database Connectivity (JDBC), and runs an INSERT INTO statement to write data to the Db2 database. Internally, data is submitted to Db2 database in batches.

Db2 Writer is designed for extract-transform-load (ETL) developers to import data from data warehouses to Db2 databases. Db2 Writer can also be used as a data migration tool by users such as database administrators (DBAs).

Db2 Writer obtains data from a Data Integration reader, and writes the data to the destination database by running 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, Db2 Writer makes batch updates with the PreparedStatement method and sets the rewriteBatchedStatements parameter to true. In this way, Db2 Writer buffers data, and submits a write request when the amount of data in the buffer reaches a specific threshold.

**Note:**

A sync node that uses Db2 Writer must have at least the permission to run 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.

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

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

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
<b>jdbcUrl</b>	The JDBC URL for connecting to the Db2 database. In accordance with official Db2 specifications, the URL must be in the <b>jdbc:db2://ip:port/database</b> format. You can also specify the information of the attachment facility.	Yes	None
<b>username</b>	The username for connecting to the Db2 database.	Yes	None
<b>password</b>	The password for connecting to the Db2 database.	Yes	None
<b>table</b>	The name of the destination table.	Yes	None
<b>column</b>	The columns in the destination table to which data is written. Separate the columns with a comma (,). Example: <b>"column": ["id", "name", "age"]</b> . 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: <b>"column": ["*"]</b> .	Yes	None
<b>preSql</b>	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.	No	None

Parameter	Description	Required	Default value
<b>postSql</b>	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.	No	None
<b>batchSize</b>	The number of data records to write at a time. 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 lead to the out of memory (OOM) error during the data synchronization process.	No	1024

### Configure Db2 Writer by using the codeless UI

Currently, the codeless UI is not supported for Db2 Writer.

### Configure Db2 Writer by using the code editor

In the following code, a node 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 Stream Reader. For more information
 , see the corresponding topic.
 {
 "stepType": "stream",
 "parameter": {},
 "name": "Reader",
 "category": "reader"
 },
 {
 "stepType": "db2", // The writer type.
 "parameter": {
 "postSql": [], // The SQL statement to run after the sync node is run.
 "password": "", // The password for connecting to the Db2 database.
 "jdbcUrl": "jdbc:db2://ip:port/database", // The JDBC URL for connecting to the
 Db2 database.
 "column": [
 "id"
],
 "batchSize": 1024, // The number of data records to write at a time.
 "table": "", // The name of the destination table.
 "username": "", // The username for connecting to the Db2 database.
 "preSql": [] // The SQL statement to run 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 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"
 }
]
}
}

```

### 8.3.3 Configure DRDS Writer

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

DRDS Writer allows you to write data to tables stored in Distributed Relational Database Service (DRDS) databases. Specifically, DRDS Writer connects to the proxy of a remote DRDS database through Java Database Connectivity (JDBC), and runs a `REPLACE INTO` statement to write data to the DRDS database.



#### Note:

- To run 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 configuring DRDS Writer. For more information, see [Configure a DRDS connection](#).

DRDS Writer is designed for extract-transform-load (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 database administrators (DBAs).

DRDS Writer obtains data from a Data Integration reader, and writes the data to the destination database by running 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 run the 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.

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
<b>datasource</b>	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None
<b>table</b>	The name of the destination table.	Yes	None
<b>writeMode</b>	The write mode. Valid values: insert ignore and replace. <ul style="list-style-type: none"> <li>insert ignore: If a primary key conflict or unique index conflict occurs, data cannot be written.</li> <li>replace: If a primary key conflict or unique index conflict occurs, original rows are deleted and new rows are inserted. That is, all fields of original rows are replaced.</li> </ul>	No	insert ignore
<b>column</b>	The columns in the destination table to which data is written. Separate the columns with a comma (.). Example: " <b>column</b> ": ["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: " <b>column</b> ": ["*"].	Yes	None

Parameter	Description	Required	Default value
<b>preSql</b>	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 UI, and multiple SQL statements in the code editor.	No	None
<b>postSql</b>	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.	No	None
<b>batchSize</b>	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 out of memory (OOM) error during the data synchronization process.	No	1024

## Configure DRDS Writer by using the codeless UI

### 1. Configure the connections.

Configure the source and destination connections for the sync node.

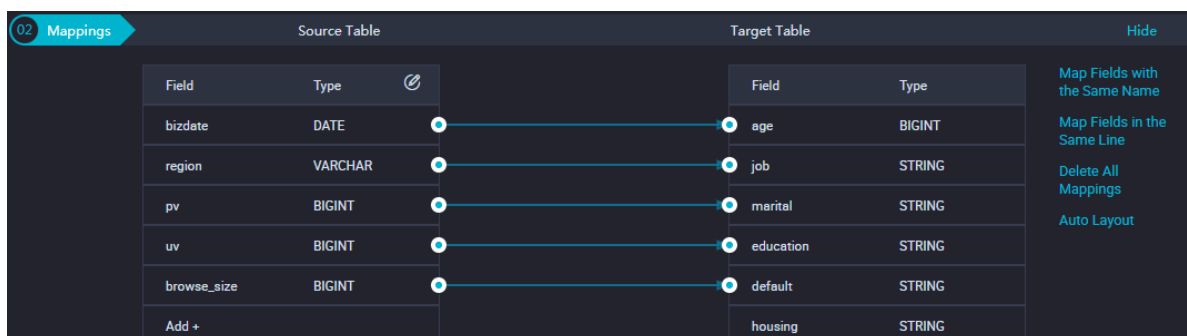
The screenshot shows the 'Data Source' configuration window in DataWorks. It is divided into 'Source' and 'Destination' tabs. Under 'Source', the 'Data Source' is set to 'MySQL' and the 'Table' is 'bird\_rds'. There is a 'Data Filtering' field with 'id=1' and a 'Sharding Key' field. Under 'Destination', the 'Data Source' is set to 'DRDS' and the 'Table' is 'px\_31'. There are two 'Statements Run' fields: 'Before Import' and 'After Import', both containing 'select \* from px\_31'. There are also help icons (?) next to several fields.

Parameter	Description
<b>Connection</b>	The <b>datasource</b> parameter in the preceding parameter description. Select a connection type, and enter the name of a connection that has been configured in DataWorks.
<b>Table</b>	The <b>table</b> parameter in the preceding parameter description.

Parameter	Description
<b>Statement Run Before Writing</b>	The <b>preSql</b> parameter in the preceding parameter description. Enter an SQL statement to run before the sync node is run.
<b>Statement Run After Writing</b>	The <b>postSql</b> parameter in the preceding parameter description. Enter an SQL statement to run after the sync node is run.
<b>Solution to Primary Key Violation</b>	The <b>writeMode</b> parameter in the preceding parameter description. Select the expected write mode.

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 a field.



Button or icon	Description
<b>Map Fields with the Same Name</b>	Click <b>Map Fields with the Same Name</b> to establish a mapping between fields with the same name. Note that the data types of the fields must match.
<b>Map Fields in the Same Line</b>	Click <b>Map Fields in the Same Line</b> to establish a mapping for fields in the same row. Note that the data types of the fields must match.
<b>Delete All Mappings</b>	Click <b>Delete All Mappings</b> to remove mappings that have been established.
<b>Auto Layout</b>	Click Auto Layout. The fields are automatically sorted based on specified rules.
<b>Change Fields</b>	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.

Button or icon	Description
<b>Add</b>	<ul style="list-style-type: none"> <li>Click Add to add a field. You can enter constants. Each constant must be enclosed in single quotation marks (' '), such as <b>'abc'</b> and <b>'123'</b>.</li> <li>You can use scheduling parameters, such as <b>\${bizdate}</b>.</li> <li>You can enter functions supported by relational databases, such as <b>now()</b> and <b>count(1)</b>.</li> <li>Fields that cannot be parsed are indicated by Unidentified.</li> </ul>

### 3. Configure channel control policies.

Parameter	Description
<b>Expected Maximum Concurrency</b>	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.
<b>Bandwidth Throttling</b>	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.
<b>Dirty Data Records Allowed</b>	The maximum number of dirty data records allowed.
<b>Resource Group</b>	The resource group used for running the sync node. If a large number of nodes including this sync node are deployed on the default resource group, the sync node may need to wait for resources. We recommend that you purchase an exclusive resource group for data integration or add a custom resource group. For more information, see <a href="#">#unique_15</a> and <a href="#">Add a custom resource group</a> .

## 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": [
 { // The following template is used to configure Stream Reader. For more information
 , see the corresponding topic.
 "stepType": "stream",
 "parameter": {},
 "name": "Reader",
 "category": "reader"
 },
 {
 "stepType": "drds", // The writer type.
 "parameter": {
 "postSql": [], // The SQL statement to run 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 run 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 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"
 }
]
 }
}
```

```
}
```

### 8.3.4 Configure FTP Writer

This topic describes the parameters supported by FTP Writer and how to configure it by using the codeless user interface (UI) and code editor.

FTP Writer allows you to write one or more comma-separated values (CSV) files to a remote FTP server. Specifically, FTP Writer converts the data obtained from a Data Integration reader to CSV files and writes these files to a remote FTP server by using FTP-related network protocols.

**Note:**

You must configure a connection before configuring FTP Writer. For more information, see [Configure an FTP connection](#).

FTP Writer can write files that store logical two-dimensional tables, such as CSV files that store text data, to an FTP server.

FTP Writer allows you to convert data obtained from a Data Integration reader to files and write the files to an FTP server. The files on the FTP server store unstructured data only. Currently, FTP Writer supports the following features:

- Writes only files that store text data. The text data must be logical two-dimensional tables. FTP Writer cannot write files that store Binary Large Object (BLOB) data, such as video data.
- Writes CSV-like and text files with custom delimiters.
- Writes compressed files to an FTP server.
- Uses concurrent threads to write files. Each thread writes a file.

Currently, FTP Reader does not support the following features:

- Uses concurrent threads to write a single file.
- Distinguishes between data types. FTP does not distinguish between data types. Therefore, FTP Writer writes all data as strings to files on an FTP server.

## Parameters

Parameter	Description	Required	Default value
<b>datasource</b>	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None
<b>timeout</b>	The timeout period to connect to the FTP server. Unit: milliseconds.	No	60000
<b>path</b>	The directory on the FTP server to which the files are written. FTP Writer writes multiple files to the directory concurrently based on the concurrency setting.	Yes	None
<b>fileName</b>	The name prefix of the files to be written to the FTP server. A random suffix is appended to the specified prefix to form the actual file name used by each thread.	Yes	None
<b>writeMode</b>	The mode in which FTP Writer writes the files. Valid values: <ul style="list-style-type: none"> <li>truncate: deletes all existing files with the specified file name prefix in the destination directory before writing files to the directory.</li> <li>append: writes the files based on the specified file name prefix and guarantees that the actual file names do not conflict with those of existing files.</li> <li>nonConflict: returns an error if a file with the specified file name prefix exists in the destination directory.</li> </ul>	Yes	None
<b>fieldDelimiter</b>	The column delimiter used in the files to be written to the FTP server. The delimiter must be a single character.	Yes	None
<b>skipHeader</b>	Specifies whether to skip the header (if exists) of a CSV-like file. The <b>skipHeader</b> parameter is not supported for compressed files.	No	false
<b>compress</b>	The compression format of the files to be written to the FTP server. The GZIP and BZIP2 compression format are supported.	No	None
<b>encoding</b>	The encoding format of the files to be written to the FTP server.	No	utf-8

Parameter	Description	Required	Default value
<b>nullFormat</b>	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
<b>dateFormat</b>	The format in which the data of the Date type is serialized in a file, for example, " <b>dateFormat</b> ":"yyyy-MM-dd".	No	None
<b>fileFormat</b>	The format in which the files are written to the FTP server. 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
<b>header</b>	The table header if the files are written as text files, for example, ['id', 'name', 'age'].	No	None
<b>markDoneFile</b>	The name of the file the existence of which indicates that the sync node succeeds. Data Integration checks whether the file exists after data synchronization. Set this parameter to the absolute path of the file.	No	None



## Configure FTP Writer by using the codeless UI

### 1. Configure the connections.

Configure the source and destination connections for the sync node.

The data sources can be default data sources or data sources created by you. Click [here](#) to check the supported data source types.

**Source**

- Data Source: **FTP** (dropdown) **ftp\_workshop\_log** (dropdown) ?
- File Path: **/home/workshop/user\_log.txt** ?
- File Type: **text** (dropdown)
- Column: **I** (text input)
- Separator: (text input)
- Encoding: **UTF-8** (dropdown)
- Null String: Enter the string that represents null (text input)
- Compression: **None** (dropdown)
- Format: (text input)
- Include Header: **No** (dropdown)

**Destination**

- Data Source: **ODPS** (dropdown) **odps\_first** (dropdown) ?
- Table: **ods\_raw\_log\_d** (dropdown)
- Partition: **dt = \${bizdate}** ?
- Clearance Rule: **Clear Existing Data Before Writing (Insert Overwrite...)** (dropdown)
- Compression: **Disable** (radio button) **Enable** (radio button)
- Consider Empty String as Null: **Yes** (radio button) **No** (radio button)

[Generate Destination Table](#)

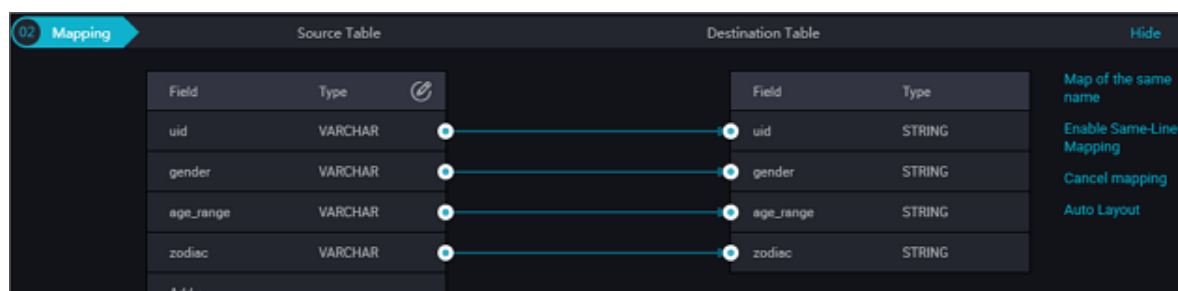
[Preview](#)

Parameter	Description
<b>Connection</b>	The <b>datasource</b> parameter in the preceding parameter description. Select a connection type, and enter the name of a connection that has been configured in DataWorks.
<b>File Path</b>	The <b>path</b> parameter in the preceding parameter description.
<b>File Type</b>	The format of the files to be written to the FTP server. The default format is CSV.
<b>Field Delimiter</b>	The <b>fieldDelimiter</b> parameter in the preceding parameter description. The default delimiter is comma (,).
<b>Encoding</b>	The <b>encoding</b> parameter in the preceding parameter description. The default encoding format is UTF-8.
<b>Null String</b>	The <b>nullFormat</b> parameter in the preceding parameter description, which defines a string that represents the null value.
<b>Time Format</b>	The <b>dateFormat</b> parameter in the preceding parameter description. Specify the format in which the data of the Date type is serialized in a file.

Parameter	Description
<b>Solution to Duplicate Prefixes</b>	The <b>writeMode</b> parameter in the preceding parameter description.

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 a field.



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

### 3. Configure channel control policies.

**03 Channel**

You can control the sync process by throttling the bandwidth or limiting the dirty data records allowed. [Learn more.](#)

\* Expected Maximum  ?  
Concurrency

\* Bandwidth Throttling ☒ Disable ☐ Enable ?

Dirty Data Records Allowed  ? The node automatically ends when the number of dirty data records reaches XX.

Parameter	Description
<b>Expected Maximum Concurrency</b>	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.
<b>Bandwidth Throttling</b>	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.
<b>Dirty Data Records Allowed</b>	The maximum number of dirty data records allowed.
<b>Resource Group</b>	The resource group used for running the sync node. If a large number of nodes including this sync node are deployed on the default resource group, the sync node may need to wait for resources. We recommend that you purchase an exclusive resource group for data integration or add a custom resource group. For more information, see <a href="#">#unique_15</a> and <a href="#">Add a custom resource group</a> .

### Configure FTP Writer by using the code editor

In the following code, a node is configured to write files to an FTP server.

```
{
 "type": "job",
 "version": "2.0", // The version number.
 "steps": [
 { // The following template is used to configure Stream Reader. For more information
 , see the corresponding topic.
 "stepType": "stream",
 "parameter": {},
 "name": "Reader",
 "category": "reader"
 },
],
}
```

```

"stepType": "ftp", // The writer type.
"parameter": {
 "path": "", // The directory on the FTP server to which the files are written.
 "fileName": "", // The name prefix of the files to be written to the FTP server.
 "nullFormat": "null", // The string that represents null.
 "dateFormat": "yyyy-MM-dd HH:mm:ss", // The format of the time.
 "datasource": "", // The connection name.
 "writeMode": "", // The write mode.
 "fieldDelimiter": ";", // The column delimiter.
 "encoding": "", // The encoding format.
 "fileFormat": "" // The format in which FTP Writer writes the files.
},
"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"
 }
]
}
}

```

### 8.3.5 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, HBase 1.1.x, and HBase 2.x are supported.
- If you use HBase 0.94.x, set the `hbaseVersion` parameter to **094x** for HBase Writer.

```

"writer": {
 "hbaseVersion": "094x"
}

```

```
}
```

- If you use HBase 1.1.x or HBase 2.x, set the hbaseVersion parameter to **11x** for HBase 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.
- Except for the data types listed in the following table, other types are not supported.

Category	HBase data type
Integer	INT, LONG, and SHORT
Floating point	FLOAT and DOUBLE
Boolean	BOOLEAN
String	STRING

## Parameters

Parameter	Description	Required	Default value
<b>haveKerberos</b>	<p>Specifies whether Kerberos authentication is required. A value of true indicates that Kerberos authentication is required.</p> <div>  <b>Note:</b> <ul style="list-style-type: none"> <li>If the value is true, the following five Kerberos-related parameters must be specified: <ul style="list-style-type: none"> <li><b>kerberosKeytabFilePath</b></li> <li><b>kerberosPrincipal</b></li> <li><b>hbaseMasterKerberosPrincipal</b></li> <li><b>hbaseRegionserverKerberosPrincipal</b></li> <li><b>hbaseRpcProtection</b></li> </ul> </li> <li>If the value is false, Kerberos authentication is not required and you do not need to specify the preceding parameters.</li> </ul> </div>	No	false
<b>hbaseConfig</b>	<p>The properties of the HBase cluster, in JSON format. The <b>hbase.zookeeper.quorum</b> 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.</p> <div>  <b>Note:</b> <p>You must use the internal endpoint to access ApsaraDB for HBase.</p> </div>	Yes	None
<b>mode</b>	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
<b>table</b>	The name of the HBase table to which data is written. The name is case-sensitive.	Yes	None
<b>encoding</b>	The encoding format in which a string is converted through <b>byte[]</b> . Currently, UTF-8 and GBK are supported.	No	utf-8

Parameter	Description	Required	Default value
<b>column</b>	The HBase columns to which data is written. <ul style="list-style-type: none"><li>• <b>index</b>: the ID of the column in the source table, starting from 0.</li><li>• <b>name</b>: the name of the column in the HBase table, in the columnFamily:column format.</li><li>• <b>type</b>: the type of the data written, which is used by the byte[] constructor.</li></ul>	Yes	None
<b>maxVersion</b>	The number of versions written by HBase Writer 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

Parameter	Description	Required	Default value
<b>range</b>	<p>The rowkey range that HBase Writer writes.</p> <ul style="list-style-type: none"> <li><b>startRowkey</b>: the start rowkey.</li> <li><b>endRowkey</b>: the end rowkey.</li> <li><b>isBinaryRowkey</b>: the operation called by <b>byte[]</b> to convert the specified start and end rowkeys. Default value: false. If the value is true, <b>Bytes.toBytesBinary(rowkey)</b> is called. If the value is false, <b>Bytes.toBytes(rowkey)</b> is called.</li> </ul> <pre>"range": {   "startRowkey": "aaa",   "endRowkey": "ccc",   "isBinaryRowkey": false }</pre> <pre>"column": [   {     "index": 1,     "name": "cf1:q1",     "type": "string"   },   {     "index": 2,     "name": "cf1:q2",     "type": "string"   } ]</pre>	No	None
<b>rowkeyColumn</b>	<p>The rowkey of each HBase cell.</p> <ul style="list-style-type: none"> <li><b>index</b>: the ID of the column in the source table, starting from 0. If the column is a constant, set the value to -1.</li> <li><b>type</b>: the type of the data written, which is used by the <b>byte[]</b> constructor.</li> <li><b>value</b>: 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.</li> </ul> <p>Example:</p> <pre>"rowkeyColumn": [   {     "index": 0,     "type": "string"   },   {     "index": -1,     "type": "string",     "value": "_"   } ]</pre>	Yes	None



Parameter	Description	Required	Default value
<b>walFlag</b>	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
<b>writeBufferSize</b>	The write buffer size, in bytes, of the HBase client. If you specify this parameter, you must also specify the autoflush parameter.  autoflush: <ul style="list-style-type: none"> <li>If the value is true, the HBase client sends a PUT request each time it receives an edit.</li> <li>If the value is false, the HBase client sends a PUT request only when its write buffer is full.</li> </ul>	No	8M

### 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 store.
 "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
column.
 "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 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"
 }
]
}
}

```

```
}
```

### 8.3.6 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.

HBase11xsql Writer allows you to write data in batches to HBase tables created through Phoenix. Phoenix can encode the primary key to rowkey. If you directly use the HBase API to write data to an HBase table created through 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.

Specifically, HBase11xsql Writer connects to a remote HBase data store through Java Database Connectivity (JDBC), and runs an UPSERT statement to write data to the HBase data store.

#### Features

HBase11xsql Writer supports writing data of an indexed table and synchronously updating all indexed tables.

#### Limits

The limits of the HBase11xsql Writer are as follows:



- HBase11xsql Writer can write data only to HBase 1.x.
- HBase11xsql Writer only supports tables created through Phoenix. Native HBase tables are not supported.
- HBase11xsql Writer cannot write data with timestamps.

#### How it works

HBase11xsql Writer connects to an HBase data store through Phoenix, which is a JDBC driver, and runs an UPSERT statement to write data in batches to the destination table. With Phoenix, you can 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

Parameter	Description	Required	Default value
table	The name of the destination table. The name is case-sensitive. Generally, the name of a table created through Phoenix consists of uppercase letters.	Yes	None
column	<p>The name of the column. The name is case-sensitive. Generally, the name of each column in a table created through Phoenix consists of uppercase letters.</p> <div>  <b>Note:</b> <ul style="list-style-type: none"> <li>HBase11xsql Writer writes data strictly in accordance with the order of the columns obtained from the reader.</li> <li>You do not need to specify the data type for each column. HBase11xsql Writer automatically obtains the metadata of columns from Phoenix.</li> </ul> </div>	Yes	None
hbaseConfig	<p>The properties of the HBase cluster. The hbase.zookeeper.quorum parameter is required. It specifies the ZooKeeper ensemble servers.</p> <div>  <b>Note:</b> <ul style="list-style-type: none"> <li>Separate the IP addresses with commas (,). Example: ip1, ip2, ip3.</li> <li>The zookeeper.znode.parent parameter is optional. Default value: /hbase.</li> </ul> </div>	Yes	None
batchSize	The number of data records to write at a time.	No	256
nullMode	<p>The method of processing null values. Valid values:</p> <ul style="list-style-type: none"> <li>skip: HBase11xsql Writer does not write null values to the HBase data store.</li> <li>empty: HBase11xsql Writer writes 0 or an empty string to the HBase data store instead of null values. For a column of the numeric type, HBase11xsql Writer writes 0. For a column of the VARCHAR type, HBase11xsql Writer writes an empty string.</li> </ul>	No	skip

### Configure HBase11xsql Writer by using the code editor

Example:

```
{
 "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 destination 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"
 }
}

```

### Column order

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.

## FAQ

- Q: What is the proper number of concurrent threads? Can I increase the number of concurrent threads to speed up the synchronization?

A: The recommended number of concurrent threads is 5 to 10. Increasing the number of concurrent threads does not help speed up the synchronization. In the data import process, the default size of a Java virtual machine (JVM) heap is 2 GB. Concurrent synchronization requires multiple threads. However, too many threads sometimes cannot speed up the synchronization and may even deteriorate the performance because of frequent garbage collection (GC).

- 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.

### 8.3.7 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, Optimized Row Columnar (ORC), or Parquet files to the specified directory in Hadoop Distributed File System (HDFS). In addition, you can associate the fields in the files with those in Hive tables. You must configure a connection before configuring HDFS Writer. For more information, see [Configure an HDFS connection](#).

**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.

## Limits

- Currently, 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 columns in a file to HDFS.
- Currently, Hive data types such as DECIMAL, BINARY, ARRAYS, MAPS, STRUCTS, and UNION are not supported.
- 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.
- Currently, 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.

### 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. After all the files are written, moves the files in the temporary directory to the specified directory in HDFS. HDFS Writer guarantees that the file names do not conflict with existing files in HDFS when moving 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 and files that are written to the directory.



#### Note:

To synchronize data, use an administrator account with the read and write permissions.

### 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


Category	Hive data type	
Boolean	BOOLEAN	
Date and time	DATE and TIMESTAMP	

### Parameters


Parameter	Description	Required	Default value
<b>defaultFS</b>	The address of the HDFS Namenode, such as <code>hdfs://127.0.0.1:9000</code> . The default resource group does not support configuring advanced Hadoop parameters related to the high availability feature. In this case, you can add a custom resource group. For more information, see <a href="#">Add a custom resource group</a> .	Yes	None
<b>fileType</b>	The format of the files to be written to HDFS. Valid values: <ul style="list-style-type: none"> <li>text: the text file format.</li> <li>orc: the ORC file format.</li> <li>parquet: the common Parquet file format.</li> </ul>	Yes	None
<b>path</b>	The directory in HDFS to which the files are written. HDFS Writer writes multiple files to the directory concurrently 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 <code>/user/hive/warehouse/</code> . The storage path of the hello table created in the test database is <code>/user/hive/warehouse/test.db/hello</code> .	Yes	None




Parameter	Description	Required	Default value
<b>fileName</b>	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

Parameter	Description	Required	Default value
<b>column</b>	<p>The columns to be written to HDFS. You cannot write only some of the columns in a file to HDFS.</p> <p>To associate the fields in a file with those in a Hive table, specify the name and type parameters for each field.</p> <p>You can also specify the column parameter in the following way:</p> <pre>"column": [   {     "name": "userName",     "type": "string"   },   {     "name": "age",     "type": "long"   } ]</pre>	Yes (Not required if the filetype parameter is set to parquet.)	None
<b>writeMode</b>	<p>The mode in which HDFS Writer writes the files. Valid values:</p> <ul style="list-style-type: none"> <li>• <b>append</b>: writes the files based on the specified file name prefix and guarantees that the actual file names do not conflict with those of existing files.</li> <li>• <b>nonConflict</b>: returns an error if a file with the specified file name prefix exists in the destination directory.</li> <li>• <b>truncate</b>: deletes all existing files with the specified file name prefix in the destination directory before writing files to the directory. For example, if you set the filename parameter to abc, all files whose names start with abc are deleted.</li> </ul> <div>  <b>Note:</b>  Parquet files do not support the append mode. They support only the nonConflict mode. </div>	Yes	None

Parameter	Description	Required	Default value
<b>fieldDelimiter</b>	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
<b>compress</b>	<p>The compression format of the files to be written to HDFS. By default, this parameter is left empty, that is, files are not compressed.</p> <p>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.</p>	No	None
<b>encoding</b>	The encoding format of the files to be written to HDFS.	No	None

Parameter	Description	Required	Default value
<b>parquetSchema</b>	<p>The schema of the files to be written to HDFS. This parameter is required only when the fileType parameter is set to parquet. Format:</p> <pre>message messageType {   required, dataType, columnName;   ..... ; }</pre> <p>The parameters are described as follows:</p> <ul style="list-style-type: none"> <li>messageTypeName: the name of the MessageType object.</li> <li>required: specifies whether the field is required or optional. We recommend that you set the parameter to optional for all fields.</li> <li>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.</li> </ul> <div>  <b>Note:</b>  Each line, including the last one, must end with a semicolon (;). </div> <p>Example:</p> <pre>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; }</pre>	No	None
<b>hadoopConfig</b>	<p>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. In this case, you can add a custom resource group. For more information, see <a href="#">Add a custom resource</a></p>	No	None

Parameter	Description	Required	Default value
<b>kerberosKeytabFilePath</b>	The absolute path of the keytab file for Kerberos authentication.	Required if the <b>haveKerberos</b> parameter is set to true	None
<b>kerberosPrincipal</b>	<p>The Kerberos principal to which Kerberos can assign tickets. Example: <b>****/hadoopclient@**. ***</b>. Required if the <b>haveKerberos</b> parameter is set to true</p> <div>  <b>Note:</b>            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:           <pre>"haveKerberos":true, "kerberosKeytabFilePath":"/opt/datax/ **/keytab", "kerberosPrincipal":"**/hadoopclient @**. ***"</pre> </div>	No	None

### Configure HDFS Writer by using the codeless UI

Currently, the codeless user interface (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 reader 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
 parameter to true.
 }
},
"order": {
 "hops": [
 {
 "from": "Reader",
 "to": "Writer"
 }
]
}
}

```

### 8.3.8 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.

The similarity and difference between ApsaraDB for Memcache databases and user-created Memcached databases are as follows:

- Same as user-created 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 on the cloud. These services are billed in pay-as-you-go mode.

Memcache Writer writes data to ApsaraDB for Memcache databases based on the Memcached protocol.

Currently, Memcache Writer writes data only in text format. The method of converting data types varies with the format of writing data.


- text: Memcache Writer uses the specified column delimiter to serialize source data to a string.
- binary: Currently, this format is not supported.

#### Parameters

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

Parameter	Description	Required	Default value
<b>writeMode</b>	<p>The write mode. Valid values:</p> <ul style="list-style-type: none"><li>• set: stores the source data.</li><li>• add: stores the source data only when its key does not exist in the destination ApsaraDB for Memcache database. Currently, this mode is not supported.</li><li>• replace: uses the source data to replace the data record with the same key in the destination ApsaraDB for Memcache database. Currently, this mode is not supported.</li><li>• 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. Currently, this mode is not supported.</li><li>• 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. Currently, this mode is not supported.</li></ul>	Yes	None



Parameter	Description	Required	Default value
<b>writeFormat</b>	<p>The format in which Memcache Writer writes the source data. Currently, only the text format is supported.</p> <p>text: serialize the source data to the text format.</p> <p>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.</p> <p>For example, the source data is as follows:</p> <pre>  ID   NAME   COUNT     --- :----- :-----    23   "CDP"   100  </pre> <p>If you set the column delimiter to a backslash and a caret (\^), data is written to ApsaraDB for Memcache in the following format:</p> <pre>  KEY (OCS)   VALUE(OCS)    :----- :-----    23        CDP\^100  </pre>	No	None
<b>expireTime</b>	<p>The expiration time of the source data to be cached in ApsaraDB for Memcache. Currently, ApsaraDB for Memcache supports the following two types of expiration time:</p> <ul style="list-style-type: none"> <li>• unixtime: the UNIX timestamp, indicating that data is invalid at a certain time point in the future. The UNIX timestamp represents the number of seconds that have elapsed since 00:00:00 on January 1, 1970.</li> <li>• seconds: the relative time in seconds starting from the current time point. It specifies the time range during which data is valid.</li> </ul> <div>  <b>Note:</b>            If the specified expiration time is larger than 30 days, the server identifies the time as the UNIX timestamp.         </div>	No	0, indicating that the data never expires

Parameter	Description	Required	Default value
<b>batchSize</b>	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 out of memory (OOM) error during the data synchronization process.	No	1024

### Configure Memcache Writer by using the codeless UI

Currently, the codeless user interface (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": [
 { // The following template is used to configure Stream Reader. For more information
 , see the corresponding topic.
 "stepType": "stream",
 "parameter": {},
 "name": "Reader",
 "category": "reader"
 },
 {
 "stepType": "ocs", // The writer type.
 "parameter": {
 "writeFormat": "text", // The format in which Memcache Writer writes the source
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 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"
 }
]
 }
 }
 }
```

### 8.3.9 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. With the powerful functionalities of indexes in MongoDB, MongoDB Writer can efficiently write data to MongoDB databases. If you want to update data, specify the primary key.



#### Note:

- You must configure a connection before configuring MongoDB Writer. For more information, see [Configure a MongoDB connection](#).
- 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 adding 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	Boolean
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
<b>datasource</b>	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None
<b>collection Name</b>	The name of the MongoDB collection.	Yes	None
<b>column</b>	The columns in MongoDB. <ul style="list-style-type: none"> <li>name: the name of the column.</li> <li>type: the data type of the column.</li> <li>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.</li> </ul>	Yes	None
<b>writeMode</b>	Specifies whether to overwrite data. <ul style="list-style-type: none"> <li>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.</li> <li>replaceKey: the primary key for each record. Data is overwritten based on this primary key. The primary key must be unique.</li> </ul>	No	None

Parameter	Description	Required	Default value
<b>preSql</b>	<p>The action to perform before the sync node is run. For example, you can clear outdated data before data synchronization. If the <b>preSql</b> parameter is left empty, no action is performed before data synchronization. Make sure that the value of the <b>preSql</b> parameter complies with the JSON syntax.</p> <p>Before running the sync node, Data Integration performs the action specified by the <b>preSql</b> parameter. After the action is completed, Data Integration starts to read or write data. The action does not affect the data that is read or written. By specifying the <b>preSql</b> parameter, you can guarantee the idempotence of the read or write operation. For example, you can specify the <b>preSql</b> parameter to clear outdated data before data synchronization based on business requirements. If the sync node fails, you only need to rerun the sync node.</p> <p>The format requirements for the <b>preSql</b> parameter are as follows:</p> <ul style="list-style-type: none"> <li>Configure the type field to specify the action type. Valid values: drop and remove. Example: <code>"preSql":{"type":"remove"}</code>. <ul style="list-style-type: none"> <li>drop: deletes the collection specified by the <code>collectionName</code> parameter and the data in the collection.</li> <li>remove: deletes data based on conditions.</li> <li>json: the conditions for deleting data. Example: <code>"preSql":{"type":"remove", "json":{"'operationTime': {'\$gte': ISODate('\${last_day}T00:00:00.424+0800')}}}"</code>. In the preceding JSON string, <code>\${last_day}</code> is a scheduling parameter of DataWorks. The format is <code>[\$yyyy-mm-dd]</code>. You can use comparison operators (such as <code>\$gt</code>, <code>\$lt</code>, <code>\$gte</code>, and <code>\$lte</code>), logical operators (such as <code>\$and</code> and <code>\$or</code>), and functions (such as <code>max</code>, <code>min</code>, <code>sum</code>, <code>avg</code>, and <code>ISODate</code>) supported by MongoDB as needed. For more</li> </ul> </li> </ul>	No	None

## Configure MongoDB Writer by using the codeless UI

Currently, the codeless user interface (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 written. If
the replacekey parameter is set to _id, set the type parameter to ObjectId. If you 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 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.
 "mbps": "1" // The maximum transmission rate.
 }
},
"order": {
 "hops": [
 {
 "from": "Reader",
 "to": "Writer"
 }
]
}
}

```

### 8.3.10 Configure MySQL Writer

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

MySQL Writer allows you to write data to tables stored in MySQL databases. Specifically, MySQL Writer connects to a remote MySQL database through Java Database Connectivity (JDBC), and runs 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 configuring MySQL Writer. For more information, see [Configure a MySQL connection](#).

MySQL Writer can be used as a data migration tool by users such as database administrators (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 run 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	BOOLEAN
Binary	TINYBLOB, MEDIUMBLOB, BLOB, LONGBLOB, and VARBINARY

## Parameters

Parameter	Description	Required	Default value
<b>datasource</b>	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None
<b>table</b>	The name of the destination table.	Yes	None
<b>writeMode</b>	<p>The write mode. Valid values: insert into, on duplicate key update, and replace into.</p> <ul style="list-style-type: none"> <li>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.</li> <li>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.</li> <li>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.</li> </ul>	No	insert into



Parameter	Description	Required	Default value
<b>column</b>	The columns in the destination table to which data is written. Separate the columns with commas (.). 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
<b>preSql</b>	<p>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 UI, and multiple SQL statements in the code editor.</p> <div>  <b>Note:</b>            If you specify multiple SQL statements in the code editor, the system does not guarantee that they are run in the same transaction.         </div>	No	None
<b>postSql</b>	<p>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.</p> <div>  <b>Note:</b>            If you specify multiple SQL statements in the code editor, the system does not guarantee that they are run in the same transaction.         </div>	No	None
<b>batchSize</b>	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 out of memory (OOM) error during the data synchronization process.	No	1024

## Configure MySQL Writer by using the codeless UI

### 1. Configure the connections.

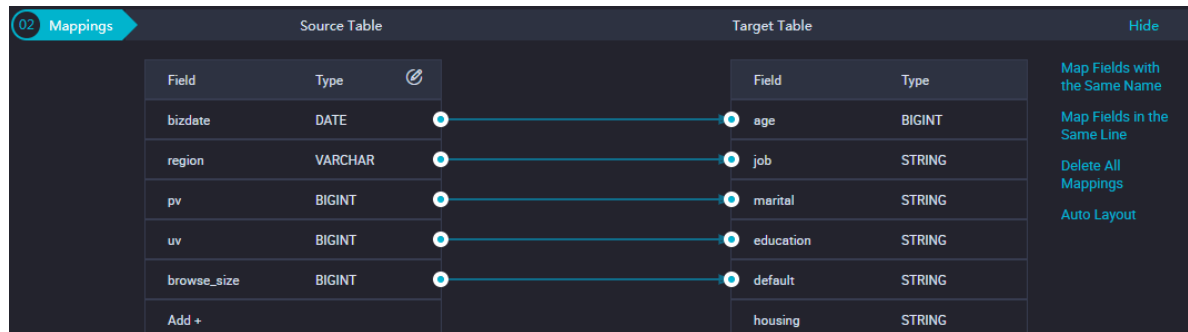
Configure the source and destination connections for the sync node.

The screenshot shows the '01 Data Source' configuration window. It has two main sections: 'Source' and 'Destination'.  
 In the 'Source' section:  
 - 'Data Source' is set to 'ODPS' with a dropdown arrow.  
 - 'Table' is set to 'odps\_first' with a dropdown arrow.  
 - 'Data Filtering' has a text area with a filter icon.  
 - 'Sharding Key' has a text area with a sharding icon.  
 In the 'Destination' section:  
 - 'Data Source' is set to 'MySQL' with a dropdown arrow.  
 - 'Table' is set to 'person' with a dropdown arrow.  
 - 'Statements Run : Before Import' has a text area with the placeholder 'Enter SQL statements to be run before data import'.  
 - 'Statements Run : After Import' has a text area with the placeholder 'Enter SQL statements to be run after data import'.  
 A 'Preview' button is at the bottom center.

Parameter	Description
<b>Connection</b>	The <b>datasource</b> parameter in the preceding parameter description. Select a connection type, and enter the name of a connection that has been configured in DataWorks.
<b>Table</b>	The <b>table</b> parameter in the preceding parameter description.
<b>Statement Run Before Writing</b>	The <b>preSql</b> parameter in the preceding parameter description. Enter an SQL statement to run before the sync node is run.
<b>Statement Run After Writing</b>	The <b>postSql</b> parameter in the preceding parameter description. Enter an SQL statement to run after the sync node is run.
<b>Solution to Primary Key Violation</b>	The <b>writeMode</b> parameter in the preceding parameter description. Select the expected write mode.

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.



Button or icon	Description
<b>Map Fields with the Same Name</b>	Click <b>Map Fields with the Same Name</b> to establish a mapping between fields with the same name. Note that the data types of the fields must match.
<b>Map Fields in the Same Line</b>	Click <b>Map Fields in the Same Line</b> to establish a mapping for fields in the same row. Note that the data types of the fields must match.
<b>Delete All Mappings</b>	Click <b>Delete All Mappings</b> to remove mappings that have been established.
<b>Auto Layout</b>	Click Auto Layout. The fields are automatically sorted based on specified rules.
<b>Change Fields</b>	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.
<b>Add</b>	<ul style="list-style-type: none"> <li>Click Add to add a field. You can enter constants. Each constant must be enclosed in single quotation marks (' '), such as <b>'abc'</b> and <b>'123'</b>.</li> <li>You can use scheduling parameters, such as <b>\${bizdate}</b>.</li> <li>You can enter functions supported by relational databases, such as <b>now()</b> and <b>count(1)</b>.</li> <li>Fields that cannot be parsed are indicated by Unidentified.</li> </ul>

### 3. Configure channel control policies.

**03 Channel**

You can control the sync process by throttling the bandwidth or limiting the dirty data records allowed. [Learn more.](#)

\* Expected Maximum  ?  
Concurrency

\* Bandwidth Throttling ☒ Disable ☐ Enable ?

Dirty Data Records Allowed  ? The node automatically ends when the number of dirty data records reaches XX.

Parameter	Description
<b>Expected Maximum Concurrency</b>	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.
<b>Bandwidth Throttling</b>	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.
<b>Dirty Data Records Allowed</b>	The maximum number of dirty data records allowed.
<b>Resource Group</b>	The resource group used for running the sync node. If a large number of nodes including this sync node are deployed on the default resource group, the sync node may need to wait for resources. We recommend that you purchase an exclusive resource group for data integration or add a custom resource group. For more information, see <a href="#">#unique_15</a> and <a href="#">Add a custom resource group</a> .

### 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": [// The following template is used to configure Stream Reader. For more
 information, see the corresponding topic.
 {
 "stepType": "stream",
 "parameter": {},
 "name": "Reader",
 "category": "reader"
 }
]
}
```

```

 },
 {
 "stepType": "mysql", // The writer type.
 "parameter": {
 "postSql": [], // The SQL statement to run 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 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"
 }
]
 }
}

```

### 8.3.11 Configure Oracle Writer

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

Oracle Writer allows you to write data to tables stored in primary Oracle databases. Specifically, Oracle Writer connects to a remote Oracle database through Java Database Connectivity (JDBC), and runs an `INSERT INTO` statement to write data to the Oracle database.



**Note:**

You must configure a connection before configuring Oracle Writer. For more information, see [Configure an Oracle connection](#).

Oracle Writer is designed for extract-transform-load (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 database administrators (DBAs).

Oracle Writer obtains data from a Data Integration reader, connects to a remote Oracle database through JDBC, and then runs 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
<b>datasource</b>	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None
<b>table</b>	The name of the destination table.	Yes	None

Parameter	Description	Required	Default value
<b>writeMode</b>	<p>The write mode. Valid values: insert into, on duplicate key update, and replace into.</p> <ul style="list-style-type: none"> <li>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.</li> <li>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.</li> <li>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.</li> </ul>	No	insert into
<b>column</b>	<p>The columns in the destination table to which data is written. Separate the columns with commas (.). 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":["*"].</p>	Yes	None
<b>preSql</b>	<p>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 UI, and multiple SQL statements in the code editor.</p>	No	None
<b>postSql</b>	<p>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.</p>	No	None
<b>batchSize</b>	<p>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 out of memory (OOM) error during the data synchronization process.</p>	No	1024

## Configure Oracle Writer by using the codeless UI

### 1. Configure the connections.

Configure the source and destination connections for the sync node.

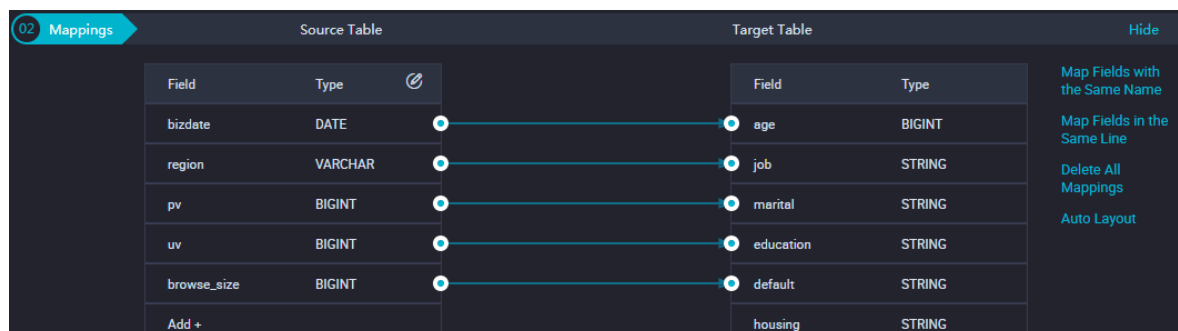
The screenshot shows the '01 Data Source' configuration window in DataWorks. It has two main tabs: 'Source' and 'Destination'.  
 In the 'Source' tab:  
 - 'Data Source' is set to 'ODPS' with a dropdown menu.  
 - 'Table' is set to 'tbl\_odps' with a dropdown menu.  
 - 'Partition' is empty with a dropdown menu.  
 - 'Compression' has radio buttons for 'Disable' (selected) and 'Enable'.  
 - 'Consider Empty String as Null' has radio buttons for 'Yes' (selected) and 'No'.  
 - A 'Preview' button is at the bottom.  
 In the 'Destination' tab:  
 - 'Data Source' is set to 'Oracle' with a dropdown menu.  
 - 'Table' is set to 'PENGXI\_PERSON' with a dropdown menu.  
 - 'Statements Run : Before Import' contains the text 'select \* from PENGXI\_PERSON'.  
 - 'Statements Run : After Import' contains the text 'select \* from PENGXI\_PERSON'.  
 A 'Hide' button is in the top right corner.

Parameter	Description
<b>Connection</b>	The <b>datasource</b> parameter in the preceding parameter description. Select a connection type, and enter the name of a connection that has been configured in DataWorks.
<b>Table</b>	The <b>table</b> parameter in the preceding parameter description.
<b>Statement Run Before Writing</b>	The <b>preSql</b> parameter in the preceding parameter description. Enter an SQL statement to run before the sync node is run.
<b>Statement Run After Writing</b>	The <b>postSql</b> parameter in the preceding parameter description. Enter an SQL statement to run after the sync node is run.
<b>Solution to Primary Key Violation</b>	The <b>writeMode</b> parameter in the preceding parameter description. Select the expected write mode.



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.



Button or icon	Description
<b>Map Fields with the Same Name</b>	Click <b>Map Fields with the Same Name</b> to establish a mapping between fields with the same name. Note that the data types of the fields must match.
<b>Map Fields in the Same Line</b>	Click <b>Map Fields in the Same Line</b> to establish a mapping for fields in the same row. Note that the data types of the fields must match.
<b>Delete All Mappings</b>	Click <b>Delete All Mappings</b> to remove mappings that have been established.
<b>Auto Layout</b>	Click Auto Layout. The fields are automatically sorted based on specified rules.
<b>Change Fields</b>	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.
<b>Add</b>	<ul style="list-style-type: none"> <li>Click Add to add a field. You can enter constants. Each constant must be enclosed in single quotation marks (' '), such as <b>'abc'</b> and <b>'123'</b>.</li> <li>You can use scheduling parameters, such as <b>\${bizdate}</b>.</li> <li>You can enter functions supported by relational databases, such as <b>now()</b> and <b>count(1)</b>.</li> <li>Fields that cannot be parsed are indicated by Unidentified.</li> </ul>

### 3. Configure channel control policies.

**03 Channel**

You can control the sync process by throttling the bandwidth or limiting the dirty data records allowed. [Learn more.](#)

\* Expected Maximum  ?  
Concurrency

\* Bandwidth Throttling ☒ Disable ☐ Enable ?

Dirty Data Records Allowed  ? The node automatically ends when the number of dirty data records reaches XX.

Parameter	Description
<b>Expected Maximum Concurrency</b>	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.
<b>Bandwidth Throttling</b>	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.
<b>Dirty Data Records Allowed</b>	The maximum number of dirty data records allowed.
<b>Resource Group</b>	The resource group used for running the sync node. If a large number of nodes including this sync node are deployed on the default resource group, the sync node may need to wait for resources. We recommend that you purchase an exclusive resource group for data integration or add a custom resource group. For more information, see <a href="#">#unique_15</a> and <a href="#">Add a custom resource group</a> .

### 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": [
 { // The following template is used to configure Stream Reader. For more information
 , see the corresponding topic.
 "stepType": "stream",
 "parameter": {},
 "name": "Reader",
 "category": "reader"
 },
],
}
```

```

"stepType":"oracle",// The writer type.
"parameter":{
 "postSql":[],// The SQL statement to run 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 run 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 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"
 }
]
}
}

```

### 8.3.12 Configure OSS Writer

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

OSS Writer allows you to write one or more CSV-like files to Object Storage Service (OSS). The number of files written to OSS depends on the number of concurrent threads and the total number of files to be synchronized.



#### Note:

You must configure a connection before configuring OSS Writer. For more information, see [Configure an OSS connection](#).

OSS Writer can write files that store logical two-dimensional tables, such as CSV files that store text data, to OSS. For more information about OSS, see [OSS overview](#).

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. Currently, 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 object when the number of rows in the current file exceeds a specific value.

Currently, 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.

## Parameters

Parameter	Description	Required	Default value
<b>datasource</b>	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None
<b>object</b>	<p>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 as follows:</p> <ul style="list-style-type: none"> <li>• "object": "datax": The names of the files start with datax, which is followed by a random string as the suffix.</li> <li>• "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.</li> </ul> <p>If you do not want to add a random universally unique identifier (UUID) as the suffix, we recommend that you set the <b>writeSingleObject</b> parameter to true.</p>	Yes	None

Parameter	Description	Required	Default value
<b>writeMode</b>	<p>The mode in which OSS Writer writes the files. Valid values:</p> <ul style="list-style-type: none"> <li>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 <code>abc</code>, all objects whose names start with <code>abc</code> are deleted.</li> <li>append: writes all files and guarantees 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 <code>DI</code>, the actual names of the files written to OSS are in the following format: <code>DI_****_****_****</code>.</li> <li>nonConflict: returns an error message if an object with the specified object name exists. For example, if you set the object parameter to <code>abc</code> and the object named <code>abc123</code> exists, an error message is returned.</li> </ul>	Yes	None
<b>fileFormat</b>	<p>The format in which the files are written to OSS. Valid values: <code>csv</code> and <code>text</code>.</p> <ul style="list-style-type: none"> <li>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 (<code>"</code>).</li> <li>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.</li> </ul>	No	text
<b>fieldDelimiter</b>	The column delimiter used in the files to be written to OSS.	No	,
<b>encoding</b>	The encoding format of the files to be written to OSS.	No	UTF-8
<b>nullFormat</b>	The string that represents null. No standard strings can represent null in text files. Therefore, Data Integration provides the <code>nullFormat</code> parameter to define which string represents a null pointer. For example, if you specify <code>nullFormat="null"</code> , Data Integration considers null as a null pointer.	No	None

Parameter	Description	Required	Default value
<b>header</b> (advanced parameter, which is not supported on the codeless UI)	The table header in the files to be written to OSS, for example, ['id', 'name', 'age'].	No	None
<b>maxFileSize</b> (advanced parameter, which is not supported on the codeless UI)	<p>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.</p> <p>If file rotation occurs, suffixes, such as <b>_1, _2, and _3</b>, are appended to the new file names that consist of file name prefixes and random UUIDs.</p>	No	<b>100000</b>
<b>suffix</b> (advanced parameter, which is not supported on the codeless UI)	The file name extension of the files to be written to OSS. For example, if you set the <b>suffix</b> parameter to .csv, the final name of a file written to OSS is in the following format: <b>fileName****.csv</b> .	No	None

## Configure OSS Writer by using the codeless UI

### 1. Configure the connections.

Configure the source and destination connections for the sync node.

The screenshot shows the '01 Data Source' configuration window. It has two main sections: 'Source' and 'Destination'. The 'Source' section is currently selected and contains the following fields: 'Data Source' (set to OSS), 'Object Prefix' (with an 'Add +' button), 'File Type' (set to csv), 'Column Separator' (set to comma), 'Encoding' (set to UTF-8), 'Null String' (with a placeholder 'Enter the sting that represents null'), 'Compression' (set to None), and 'Include Header' (set to No). The 'Destination' section contains: 'Data Source' (set to OSS), 'Object Prefix', 'File Type' (set to csv), 'Column Separator', 'Encoding' (set to UTF-8), 'Null String' (with a placeholder 'Enter the sting that represents null'), 'Time Format' (with a placeholder 'Enter the time format'), and 'Solution to Duplicate' (set to Replace the Original File). A 'Preview' button is located at the bottom of the Source section.

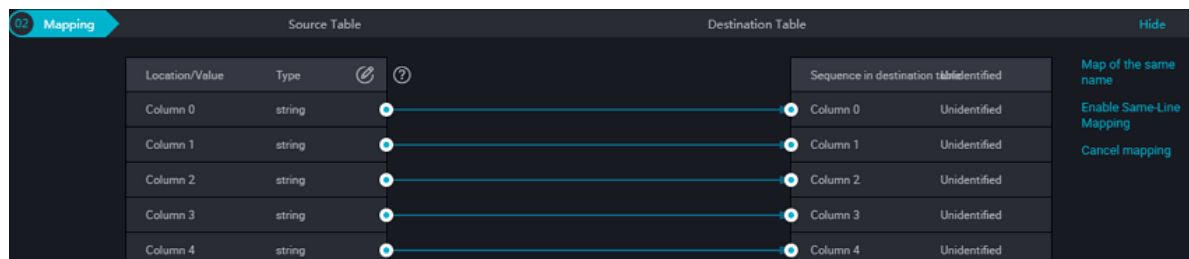
Parameter	Description
<b>Connection</b>	The <b>datasource</b> parameter in the preceding parameter description. Select a connection type, and enter the name of a connection that has been configured in DataWorks.
<b>Object Name Prefix</b>	The <b>object</b> parameter in the preceding parameter description. Enter the path of the directory for storing the files. Do not include the name of the OSS bucket in the path.
<b>File Type</b>	The fileFormat parameter in the preceding parameter description. Valid values: csv and text.

Parameter	Description	
Field Delimiter	The <b>fieldDelimiter</b> parameter in the preceding parameter description. The default delimiter is comma (,).	
Encoding	The <b>encoding</b> parameter in the preceding parameter description. The default encoding format is UTF-8.	
Null String	The <b>nullFormat</b> 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.	
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	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.	



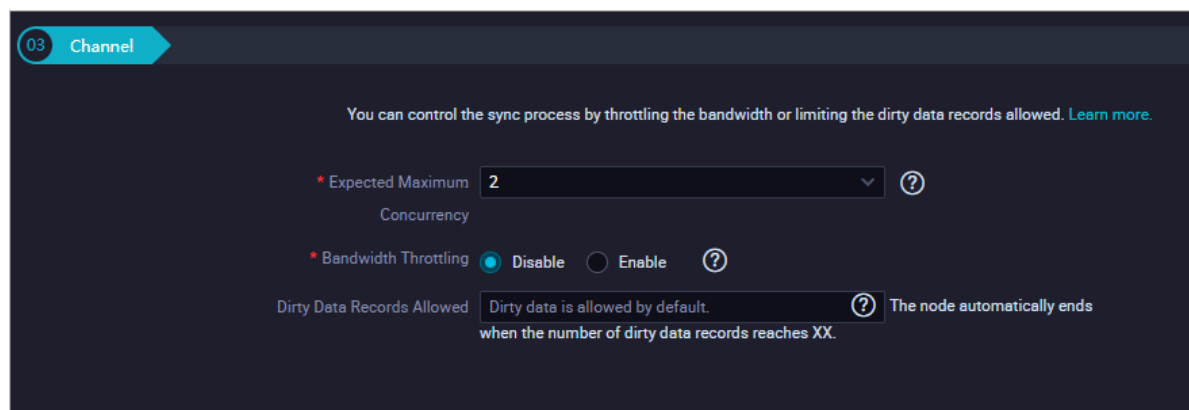
2. Configure field mapping, that is, the **column** parameter in the preceding parameter description.

Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right.



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

3. Configure channel control policies.



Parameter	Description
<b>Expected Maximum Concurrency</b>	The maximum number of concurrent threads to read and write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.

Parameter	Description
<b>Bandwidth Throttling</b>	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.
<b>Dirty Data Records Allowed</b>	The maximum number of dirty data records allowed.
<b>Resource Group</b>	The servers on which nodes are run. If an excessively large number of nodes are run on the default resource group, some nodes may be delayed due to insufficient resources. In this case, we recommend that you purchase an exclusive resource group for data integration or add a custom resource group. For more information, see <a href="#">#unique_15</a> and <a href="#">Add a custom resource group</a> .

### 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 serialized 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 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"
 }
]
}
}
}

```

### Write ORC or Parquet files to OSS

Currently, OSS Writer uses HDFS Writer to write ORC or Parquet files to OSS. In addition to the original parameters, OSS Writer provides extended parameters **Path** and **FileFormat**. For more information about the extended parameters, see [Configure HDFS Writer](#).

- In the following code, a node is configured to write ORC files to OSS.

```

{
 "stepType": "oss",
 "parameter": {
 "datasource": "",
 "fileFormat": "orc",
 "path": "/tests/case61",
 "fileName": "orc",
 "writeMode": "append",
 "column": [
 {
 "name": "col1",
 "type": "BIGINT"
 },
 {
 "name": "col2",
 "type": "DOUBLE"
 },
 {
 "name": "col3",
 "type": "STRING"
 }
],
 "writeMode": "append",
 "fieldDelimiter": "\t",
 "compress": "NONE",
 "encoding": "UTF-8"
 }
}

```

- In the following code, a node is configured to write Parquet files to OSS.

```

{
 "stepType": "oss",
 "parameter": {
 "datasource": "",

```

```

 "fileFormat": "parquet",
 "path": "/tests/case61",
 "fileName": "test",
 "writeMode": "append",
 "fieldDelimiter": "\t",
 "compress": "SNAPPY",
 "encoding": "UTF-8",
 "parquetSchema": "message test { required int64 int64_col;\n required binary
str_col (UTF8);\nrequired group params (MAP) {\nrepeated group key_value {\n
required binary key (UTF8);\nrequired binary value (UTF8);\n}\n}\nrequired group
params_arr (LIST) {\n repeated group list {\n required binary element (UTF8);\n
}\n}\nrequired group params_struct {\n required int64 id;\n required binary name (
UTF8);\n }\nrequired group params_arr_complex (LIST) {\n repeated group list {\n
required group element {\n required int64 id;\n required binary name (UTF8);\n}\n
}\n}\nrequired group params_complex (MAP) {\nrepeated group key_value {\nrequired
binary key (UTF8);\nrequired group value {\n required int64 id;\n required binary
name (UTF8);\n }\n}\n}\nrequired group params_struct_complex {\n required int64
id;\n required group detail {\n required int64 id;\n required binary name (UTF8);\n
}\n}\n}",
 "dataxParquetMode": "fields"
 }
}

```

### 8.3.13 Configure PostgreSQL Writer

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

PostgreSQL Writer allows you to write data to a PostgreSQL database. Specifically, PostgreSQL Writer connects to a remote PostgreSQL database through Java Database Connectivity (JDBC), and runs an SQL statement to write data to the PostgreSQL database.



#### Note:

You must configure a connection before configuring PostgreSQL Writer. For more information, see [Configure a PostgreSQL connection](#).

- 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	BOOLEAN
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
<b>datasource</b>	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None
<b>table</b>	The name of the destination table.	Yes	None
<b>writeMode</b>	<p>The write mode. Valid values: insert and copy.</p> <ul style="list-style-type: none"> <li>• insert: runs 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.</li> <li>• 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.</li> </ul>	No	insert

Parameter	Description	Required	Default value
<b>column</b>	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
<b>preSql</b>	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 UI, and multiple SQL statements in the code editor.	No	None
<b>postSql</b>	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.	No	None
<b>batchSize</b>	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 out of memory (OOM) error during the data synchronization process.	No	1024

Parameter	Description	Required	Default value
<b>pgType</b>	<p>The PostgreSQL configuration for converting data types. Valid values: bigint[], double[], text[], jsonb, and json. Example:</p> <pre> {   "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"           }         }       }     }   ] } </pre>	No	None

### Configure PostgreSQL Writer by using the codeless UI

#### 1. Configure the connections.

Configure the source and destination connections for the sync node.

The data sources can be default data sources or data sources created by you. Click [here](#) to check the supported data source types.

\* Data Source: SQL Server SQL Server ?

\* Table: PENGXI\_PERSON ?

Data Filtering: 1 ?

Sharding Key:  ?

Preview

\* Data Source: PostgreSQL PostgreSQL ?

\* Table: PENGXI\_PERSON ?

Statements Run: select \* from PENGXI\_PERSON ?

Before Import

Statements Run: select \* from PENGXI\_PERSON ?

After Import

Parameter	Description
<b>Connection</b>	The <b>datasource</b> parameter in the preceding parameter description. Select a connection type, and enter the name of a connection that has been configured in DataWorks.
<b>Table</b>	The <b>table</b> parameter in the preceding parameter description.
<b>Statement Run Before Writing</b>	The <b>preSql</b> parameter in the preceding parameter description. Enter an SQL statement to run before the sync node is run.
<b>Statement Run After Writing</b>	The <b>postSql</b> parameter in the preceding parameter description. Enter an SQL statement to run after the sync node is run.
<b>Write Method</b>	The <b>writeMode</b> parameter in the preceding parameter description. You can select insert or copy.



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.



Parameter	Description
<b>Map Fields with the Same Name</b>	Click <b>Map Fields with the Same Name</b> to establish a mapping between fields with the same name. Note that the data types of the fields must match.
<b>Map Fields in the Same Line</b>	Click <b>Map Fields in the Same Line</b> to establish a mapping for fields in the same row. Note that the data types of the fields must match.
<b>Delete All Mappings</b>	Click <b>Delete All Mappings</b> to remove mappings that have been established.
<b>Auto Layout</b>	Click Auto Layout. The fields are automatically sorted based on specified rules.

3. Configure channel control policies.

03 Channel

You can control the sync process by throttling the bandwidth or limiting the dirty data records allowed. [Learn more.](#)

Expected Maximum

2

?

Concurrency

Bandwidth Throttling

☒ Disable
 ☐ Enable

?

Dirty Data Records Allowed

Dirty data is allowed by default.

?

The node automatically ends when the number of dirty data records reaches XX.

Parameter	Description
<b>Expected Maximum Concurrency</b>	The maximum number of concurrent threads to read and write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.

Parameter	Description
<b>Bandwidth Throttling</b>	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.
<b>Dirty Data Records Allowed</b>	The maximum number of dirty data records allowed.
<b>Resource Group</b>	The servers on which nodes are run. If an excessively large number of nodes are run on the default resource group, some nodes may be delayed due to insufficient resources. In this case, we recommend that you purchase an exclusive resource group for data integration or add a custom resource group. For more information, see <a href="#">#unique_15</a> and <a href="#">Add a custom resource group</a> .

### 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 run after the sync node is run.
 "datasource": "", // The connection name.
 "col1",
 "col2"
 },
 "table": "", // The name of the destination table.
 "preSql": [] // The SQL statement to run 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 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"
 }
]
}
}
```

### 8.3.14 Configure Redis Writer

Redis Writer is a writer plug-in developed based on the Data Integration framework. It can be used to import data from data stores such as data warehouses to Redis databases.

REmote DIctionary Server (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. For more information about Redis, see [redis.io](https://redis.io).

Redis Writer interacts with a Redis server through 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 configuring Redis Writer. For more information, see [Configure a Redis connection](#).
- 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
<b>datasource</b>	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None

Parameter	Description	Required	Default value
<b>keyIndexes</b>	<p>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 <b>keyIndexes</b> parameter to [0,1].</p> <div>  <b>Note:</b>            After you specify the <b>keyIndexes</b> 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.         </div>	Yes	None
<b>keyFieldDelimiter</b>	The delimiter used to separate keys when data is written to Redis. Example: <b>key=key1\u0001id</b> . If multiple keys need to be concatenated, this parameter is required. If only one key exists, this parameter is not required.	No	\u0001
<b>batchSize</b>	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 out of memory (OOM) error during the data synchronization process.	No	1000
<b>expireTime</b>	<p>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.</p> <ul style="list-style-type: none"> <li>seconds: the relative time in seconds starting from the current time point. It specifies the time range during which data is valid.</li> <li>unixtime: the UNIX timestamp, indicating that data is invalid at a certain time point in the future. The UNIX timestamp represents the number of seconds that have elapsed since 00:00:00 on January 1, 1970.</li> </ul> <div>  <b>Note:</b>            If the specified expiration time is larger than 30 days, the server identifies the time as the UNIX timestamp.         </div>	No	0, indicating that the values never expire
<b>timeout</b>	The timeout period to connect to Redis when data is written to Redis. Unit: milliseconds.	No	30000

Parameter	Description	Required	Default value
<b>dateFormat</b>	The format in which the data of the Date type is written to Redis. Set the value to <b>yyyy-MM-dd HH:mm:ss</b> .	No	None
<b>writeMode</b>	<p>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 with the specified data type of the values.</p> <div> <b>Note:</b> When configuring 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.</div>	No	string

The following table lists the data types supported by Redis Writer.

Type	Parameter	Description	Required
String  <pre>"writeMode":{   "type": "string",   "mode": "set",   "valueFieldDelimiter": "\u0001" }</pre>	<b>type</b>	The data type of the values is String.	Yes
	<b>mode</b>	The mode in which data of the String type is written to Redis.	Yes. Valid value: set (overwrites the existing data).
	<b>valueFieldDelimiter</b>	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: <b>value1\u0001value2\u0001value3.</b>	No. Default value: \u0001.
List  <pre>"writeMode":{   "type": "list",   "mode": "lpush  rpush",   "valueFieldDelimiter": "\u0001" }</pre>	<b>type</b>	The data type of the values is List.	Yes
	<b>mode</b>	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).
	<b>valueFieldDelimiter</b>	The delimiter used to separate values if the data is of the String type. Example: <b>value1\u0001value2\u0001value3.</b>	No. Default value: \u0001.
Set  <pre>"writeMode":{   "type": "set",   "mode": "sadd",   "valueFieldDelimiter": "\u0001" }</pre>	<b>type</b>	The data type of the values is Set.	Yes
	<b>mode</b>	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

## 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 data is
 // written to Redis.
 "dateFormat": "yyyy-MM-dd HH:mm:ss", // The format in which the data of the
 // Date 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 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"
 }
]
 }
}
}Writer"
```

```
}
 }
}
```

### 8.3.15 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 user interface (UI) and code editor.

SQL Server Writer allows you to write data to tables stored in primary SQL Server databases. Specifically, SQL Server Writer connects to a remote SQL Server database through Java Database Connectivity (JDBC), and runs an `INSERT INTO` statement to write data to the SQL Server database. Internally, data is submitted to the database in batches.

**Note:**

You must configure a connection before configuring SQL Server Writer. For more information, see [Configure an SQL Server connection](#).

SQL Server Writer is designed for extract-transform-load (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 database administrators (DBAs).

SQL Server Writer obtains data from a Data Integration reader, and writes the data to the destination database by running 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 the `rewriteBatchedStatements` parameter 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 run 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
<b>datasource</b>	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None
<b>table</b>	The name of the destination table.	Yes	None
<b>column</b>	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
<b>preSql</b>	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 UI, and multiple SQL statements in the code editor.	No	None

Parameter	Description	Required	Default value
<b>postSql</b>	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.	No	None
<b>writeMode</b>	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
<b>batchSize</b>	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 out of memory (OOM) error during the data synchronization process.	No	1024

## Configure SQL Server Writer by using the codeless UI

### 1. Configure the connections.

Configure the source and destination connections for the sync node.

The screenshot shows the configuration interface for a SQL Server Writer. It has two main sections: 'Source' and 'Destination'.  
 In the 'Source' section:  
 - 'Data Source' is set to 'SQL Server'.  
 - 'Table' is set to 'public.Person'.  
 - 'Data Filtering' has a placeholder text in Chinese: '请参考相应SQL语法编写where过滤语句（不要填写where关键字）。该过滤语句将用作增量同步'.  
 - 'Sharding Key' is set to 'col'.  
 - 'Statements Run' is set to 'select \* from PERSON PERSON'.  
 In the 'Destination' section:  
 - 'Data Source' is set to 'SQL Server'.  
 - 'Table' is set to 'dbo.writer'.  
 - 'Statements Run' is set to 'select \* from PERSON PERSON'.  
 A 'Preview' button is located at the bottom of the 'Source' section.

Parameter	Description
<b>Connection</b>	The <b>datasource</b> parameter in the preceding parameter description. Select a connection type, and enter the name of a connection that has been configured in DataWorks.

Parameter	Description
<b>Table</b>	The <b>table</b> parameter in the preceding parameter description.
<b>Statement Run Before Writing</b>	The <b>preSql</b> parameter in the preceding parameter description. Enter an SQL statement to run before the sync node is run.
<b>Statement Run After Writing</b>	The <b>postSql</b> parameter in the preceding parameter description. Enter an SQL statement to run after the sync node is run.
<b>Solution to Primary Key Violation</b>	The <b>writeMode</b> parameter in the preceding parameter description. Select the expected write mode.

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
<b>Map Fields with the Same Name</b>	Click <b>Map Fields with the Same Name</b> to establish a mapping between fields with the same name. Note that the data types of the fields must match.
<b>Map Fields in the Same Line</b>	Click <b>Map Fields in the Same Line</b> to establish a mapping for fields in the same row. Note that the data types of the fields must match.
<b>Delete All Mappings</b>	Click <b>Delete All Mappings</b> to remove mappings that have been established.
<b>Auto Layout</b>	Click Auto Layout. The fields are automatically sorted based on specified rules.
<b>Change Fields</b>	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.

Parameter	Description
<b>Add</b>	<ul style="list-style-type: none"> <li>Click Add to add a field. You can enter constants. Each constant must be enclosed in single quotation marks (' '), such as <b>'abc'</b> and <b>'123'</b>.</li> <li>You can use scheduling parameters, such as <b>\${bizdate}</b>.</li> <li>You can enter functions supported by relational databases, such as <b>now()</b> and <b>count(1)</b>.</li> <li>Fields that cannot be parsed are indicated by Unidentified.</li> </ul>

### 3. Configure channel control policies.

**03 Channel**

You can control the sync process by throttling the bandwidth or limiting the dirty data records allowed. [Learn more.](#)

\* Expected Maximum  ?  
Concurrency

\* Bandwidth Throttling ☒ Disable ☐ Enable ?

Dirty Data Records Allowed  ? The node automatically ends when the number of dirty data records reaches XX.

Parameter	Description
<b>Expected Maximum Concurrency</b>	The maximum number of concurrent threads to read and write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
<b>Bandwidth Throttling</b>	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.
<b>Dirty Data Records Allowed</b>	The maximum number of dirty data records allowed.
<b>Resource Group</b>	The servers on which nodes are run. If an excessively large number of nodes are run on the default resource group, some nodes may be delayed due to insufficient resources. In this case, we recommend that you purchase an exclusive resource group for data integration or add a custom resource group. For more information, see <a href="#">#unique_15</a> and <a href="#">Add a custom resource group</a> .

## 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 run 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 run 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 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"
 }
]
 }
}
```

}

### 8.3.16 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
<b>endpoint</b>	The endpoint for accessing Elasticsearch, in the format of http://xxx.com:9999.	No	None
<b>accessId</b>	<p>The AccessKey ID for accessing Elasticsearch, which is used for authorization when a connection with Elasticsearch is established.</p> <div>  <b>Note:</b>            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.         </div>	No	None
<b>accessKey</b>	The AccessKey secret for accessing Elasticsearch.	No	None
<b>index</b>	The index name in Elasticsearch.	No	None
<b>indexType</b>	The type name in the index of Elasticsearch.	No	Elasticsearch

Parameter	Description	Required	Default value
<b>cleanup</b>	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
<b>batchSize</b>	The number of data records to write at a time.	No	1000
<b>trySize</b>	The number of retries after a failure.	No	30
<b>timeout</b>	The connection timeout of the client.	No	600000
<b>discovery</b>	Specifies whether to enable Node Discovery . When Node Discovery is enabled, the server list in the client is polled and regularly updated.	No	false
<b>compression</b>	Specifies whether to enable compression for an HTTP request.	No	true
<b>multiThread</b>	Specifies whether to use multiple threads for an HTTP request.	No	true
<b>ignoreWriteError</b>	Specifies whether to ignore write errors and proceed with writing without retries.	No	false
<b>ignoreParseError</b>	Specifies whether to ignore format parsing errors and proceed with writing.	No	true
<b>alias</b>	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

Parameter	Description	Required	Default value
<b>aliasMode</b>	<p>The mode in which an alias is added after the data is imported. Valid values: append and exclusive.</p> <ul style="list-style-type: none"> <li>append: adds an alias for the current index. One alias maps multiple indexes.</li> <li>exclusive: deletes the existing alias of the current index and adds a new alias. One alias maps one index.</li> </ul> <p>Elasticsearch Writer can convert aliases to actual index names. Using aliases helps you migrate data from one index to another , search data across multiple indexes in a unified manner, and create a view on a subset of data in an index.</p>	No	append
<b>splitter</b>	<p>The delimiter (-,-) for splitting the source data if you are inserting an array to Elasticsearch.</p> <p>For 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.</p>	No	-,-
<b>settings</b>	The settings of an index. The settings must be in accordance with Elasticsearch official specifications.	No	None



Parameter	Description	Required	Default value
<b>column</b>	<p>The fields of the document. The parameters for each field include basic parameters such as <b>name</b> and <b>type</b> and advanced parameters such as <b>analyzer</b>, <b>format</b>, and <b>array</b>.</p> <p>The field types supported by Elasticsearch are as follows:</p> <ul style="list-style-type: none"> <li>- id // The id type corresponds to the <code>_id</code> type in Elasticsearch, and can be considered as the unique primary key. Data with the same ID will be overwritten and not indexed.</li> <li>- string</li> <li>- text</li> <li>- keyword</li> <li>- long</li> <li>- integer</li> <li>- short</li> <li>- byte</li> <li>- double</li> <li>- float</li> <li>- date</li> <li>- boolean</li> <li>- binary</li> <li>- integer_range</li> <li>- float_range</li> <li>- long_range</li> <li>- double_range</li> <li>- date_range</li> <li>- geo_point</li> <li>- geo_shape</li> <li>- ip</li> <li>- token_count</li> <li>- array</li> <li>- object</li> <li>- nested</li> </ul> <ul style="list-style-type: none"> <li>• When the field type is Text, you can specify the <b>analyzer</b>, <b>norms</b>, and <b>index_options</b> parameters. Example: <pre> {   "name": "col_text",   "type": "text",   "analyzer": "ik_max_word" } </pre> </li> <li>• When the field type is date, you can specify the <b>format</b> and <b>timezone</b> parameters, indicating the date serialization format and the time zone, respectively. Alternatively, you can specify the <b>origin</b> parameter instead of timezone. Example: <pre> {   "name": "col_date",   "type": "date",   "format": "yyyy-MM-dd HH:mm:ss",   "origin": true } </pre> </li> </ul>	Yes	None
Issue: 20200628	<pre> {   "name": "col_date",   "type": "date",   "format": "yyyy-MM-dd HH:mm:ss",   "origin": true } </pre>		485

Parameter	Description	Required	Default value
<b>actionType</b>	<p>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.</p> <ul style="list-style-type: none"> <li>index: Data Integration uses <b>Index.Builder</b> 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. <ul style="list-style-type: none"> <li>If the ID is not specified, Elasticsearch generates a unique ID by default. In this case, the document is directly inserted to Elasticsearch.</li> <li>If the ID is specified, Elasticsearch replaces the existing document with the document to be inserted.</li> </ul> </li> </ul> <div style="background-color: #f0f0f0; padding: 10px; margin: 10px 0;">  <b>Note:</b> In this case, you cannot modify specific fields in the document. </div> <ul style="list-style-type: none"> <li>update: Data Integration uses <b>Update.Builder</b> 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.</li> </ul>	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. To write data to the Elasticsearch database deployed in a VPC, use [#unique\\_15/unique\\_15\\_Connect\\_42\\_section\\_hcx\\_l05\\_vcg](#) or [custom resource groups](#).

### 8.3.17 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
<b>endpoint</b>	The endpoint for accessing Log Service.	Yes	None
<b>accessKeyId</b>	The AccessKey ID for accessing Log Service.	Yes	None
<b>accessKeySecret</b>	The AccessKey secret for accessing Log Service.	Yes	None
<b>project</b>	The name of the destination Log Service project.	Yes	None
<b>logstore</b>	The name of the destination Logstore.	Yes	None
<b>topic</b>	The name of the destination topic.	No	Empty string
<b>batchSize</b>	<p>The number of data records to write to LogHub at a time. Default value: 1024.</p> <div>  <b>Note:</b>            The size of the data to write to LogHub at a time cannot exceed 5 MB. Change the value of this parameter based on the size of a single data record.         </div>	No	1024  By default, LogHub Writer writes 1,024 records to LogHub at a time. You can change the value as required.
<b>column</b>	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
 parameter to true.
 }
 },
 "order": {
 "hops": [
 {
 "from": "Reader",
 "to": "Writer"
 }
]
 }
}
```

```
}
```

### 8.3.18 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.

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.

#### How it works

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.
- Currently, a sync node that is run on the default resource group may fail to connect to Open Search that is deployed in a Virtual Private Cloud (VPC).

#### Features

##### Column order

The columns in Open Search are unordered. Therefore, 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 are set to the default value or null.

For example, an Open Search table contains column a, column b, and column c, and you only need to write data to column b and column c. You can set the column parameter to `["c","b"]`. In this case, the first and second columns of the source data obtained from a reader are imported to column c and column b in Open Search respectively. Column a in the Open Search table is set to the default value or null.

- Column configuration error handling

To avoid losing the data of redundant columns and guarantee high data reliability, Open Search Writer returns an error message if more columns are written than expected. For



example, if an Open Search table contains column a, column b, and column 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.

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
<b>accessId</b>	The AccessKey ID for accessing Alibaba Cloud.	Yes	None
<b>accessKey</b>	The AccessKey secret for accessing Alibaba Cloud.	Yes	None
<b>host</b>	The endpoint for accessing Open Search. You can view the endpoint in the Alibaba Cloud console.	Yes	None
<b>indexName</b>	The name of the Open Search project.	Yes	None
<b>table</b>	The table to which data is written. You can specify only one table because Data Integration does not support importing data to multiple tables at a time.	Yes	None

Parameter	Description	Required	Default value
<b>column</b>	<p>The columns in the destination table to which data is written. 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": ["*"]. Separate the columns with a comma (,) if data is written to some of the columns in the destination table. Example: "column": ["id","name"].</p> <p>Open Search Writer supports filtering columns and changing the order of columns. For example, an Open Search 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 as follows: "column": ["c","b"]. During data synchronization, column a is automatically set to null.</p>	Yes	None
<b>batchSize</b>	<p>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 applied for the account used to access Open Search.</p> <p>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.</p>	Required only for writing data to a partitioned table	300

Parameter	Description	Required	Default value
<b>writeMode</b>	<p>The write mode. To guarantee the idempotence of write operations, set the writeMode parameter to <b>add/update</b> when you configure Open Search Writer.</p> <ul style="list-style-type: none"> <li><b>add</b>: deletes the existing data record and inserts the new data record to Open Search, which is an atomic operation.</li> <li><b>update</b>: updates the existing data record based on the new data record, which is an atomic operation.</li> </ul> <div>  <b>Note:</b>  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. Currently, Open Search V3 does not support the update mode. </div>	Yes	None
<b>ignoreWriteError</b>	<p>Specifies whether to ignore failed write operations.</p> <p>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.</p>	No	false
<b>version</b>	<p>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.</p>	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
 }
 }
}

```

### 8.3.19 Configure Table Store Writer

This topic describes the data types and parameters supported by Table Store Writer and how to configure it by using the code editor.

Table Store 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. Table Store organizes data into instances and tables. Using data sharding and load balancing technologies, Table Store seamlessly expands the data scale.

Table Store Writer connects to a Table Store server by using the official Java SDK and writes data to the Table Store server by using the SDK. Table Store Writer has greatly optimized the write process, including retry upon write timeout, retry upon exceptions, and batch submission.

Currently, Table Store Writer supports all Table Store data types. The following table lists the data types supported by Table Store Writer.


Category	Table Store data type
Integer	INTEGER
Floating point	Double
String	STRING
Boolean	BOOLEAN
Binary	BINARY

**Note:**

To write data of the INTEGER type, set the data type to INT in the code editor. Then, DataWorks converts the INT type to the Integer type. If you set the data type to INTEGER for the data to be written to Table Store, an error is reported in the log and the sync node fails.

**Parameters**

Parameter	Description	Required	Default value
<b>datasource</b>	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None
<b>endPoint</b>	The endpoint of the Table Store server.	Yes	None
<b>accessId</b>	The AccessKey ID for accessing Table Store.	Yes	None
<b>accessKey</b>	The AccessKey secret for accessing Table Store.	Yes	None
<b>instanceName</b>	The name of the Table Store instance to access.  You access and manage the Table Store service through an instance. After activating Table Store, you can create an instance in the Table Store console and then create and manage tables in the instance. Instances are the basic unit for managing Table Store resources. All access control and resource measurement for applications are completed at the instance level.	Yes	None
<b>table</b>	The name of the destination table. You can specify only one table as the destination table. Multi-table synchronization is not required for Table Store.	Yes	None

Parameter	Description	Required	Default value
<b>primaryKey</b>	<p>The primary keys of the destination table in Table Store. The primary keys are described in a JSON array. Table Store is a NoSQL database service. The field names must be specified for Table Store Writer to import data.</p> <div>  <b>Note:</b>            The primary keys in Table Store only support the STRING and INT types. Therefore, you must set the data type of a primary key to either of the two types in the code editor.         </div> <p>Data Integration supports converting data types. Table Store Writer can convert data that is not of the STRING or INT type to the STRING or INT type. Example:</p> <pre>"primaryKey" : [   {"name":"pk1", "type":"string"},   {"name":"pk2", "type":"int"} ],</pre>	Yes	None
<b>column</b>	<p>The columns in the destination table to which data is written. The columns are described in a JSON array.</p> <p>Format:</p> <pre>{"name":"col2", "type":"INT"},</pre> <p>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 Table Store include String, Int, Double, Boolean, and Binary.</p>	Yes	None

Parameter	Description	Required	Default value
<b>writeMode</b>	<p>The write mode. Valid values: PutRow and UpdateRow.</p> <ul style="list-style-type: none"> <li>PutRow: the PutRow API operation for Table Store, 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.</li> <li>UpdateRow: the UpdateRow API operation for Table Store, 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 deleted as requested.</li> </ul>	Yes	None
<b>requestTotalSizeLimitation</b>	The maximum size of data in a row to be written to Table Store. The value is a numeric.	No	1MB
<b>attributeColumnSizeLimitation</b>	The maximum size of data in an attribute column to be written to Table Store. The value is a numeric.	No	2MB
<b>primaryKeyColumnSizeLimitation</b>	The maximum size of data in a primary key column to be written to Table Store. The value is a numeric.	No	1KB
<b>attributeColumnMaxCount</b>	The maximum number of attribute columns to be written to Table Store. The value is a numeric.	No	1024

### Configure Table Store Writer by using the codeless UI

Currently, the codeless user interface (UI) is not supported for Table Store Writer.

### Configure Table Store Writer by using the code editor

In the following code, a node is configured to write data to Table Store.

```
{
 "type": "job",
 "version": "2.0", // The version number.
 "steps": [
 {
 "stepType": "stream",
 "parameter": {},
 "name": "Reader",
 "category": "reader"
 }
]
}
```

```

{
 "stepType":"ots",// The writer type.
 "parameter":{
 "datasource":"","// The connection name.
 "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 keys of the destination table in Table Store.
 {
 "name":"pk1",
 "type":"STRING"
 },
 {
 "name":"pk2",
 "type":"INT"
 }
]
 },
 "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"
 }
]
 }
}

```



```
}
```

### 8.3.20 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
<code>jdbcUrl</code>	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. <ul style="list-style-type: none"><li>Format for DM databases: <code>jdbc:dm://ip:port/database</code></li><li>Format for Db2 databases: <code>jdbc:db2://ip:port/database</code></li><li>Format for PPAS databases: <code>jdbc:edb://ip:port/database</code></li></ul>	Yes	None
<code>username</code>	The username for connecting to the database.	Yes	None

Parameter	Description	Required	Default value
password	The password for connecting to the database.	Yes	None
table	The name of the destination table.	Yes	None
column	<p>The columns in the destination table to which data is written. Separate the columns with a comma (,).</p> <div>  <b>Note:</b>            We recommend that you do not use the default setting.         </div>	Yes	None
preSql	<p>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.</p> <div>  <b>Note:</b>            If you specify multiple SQL statements in the code editor, the system does not guarantee that they are run in the same transaction.         </div>	No	None
postSql	<p>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.</p> <div>  <b>Note:</b>            If you specify multiple SQL statements in the code editor, the system does not guarantee that they are run in the same transaction.         </div>	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.

```
{
 "type": "job",
 "steps": [
 {
 "stepType": "oracle",
 "parameter": {
 "datasource": "aaa",
 "column": [
 "PROD_ID",
 "name"
],
 "where": "",
 "splitPk": "",
 "encoding": "UTF-8",
 "table": "PENGXI.SALES"
 },
 "name": "Reader",
 "category": "reader"
 },
 {
 "stepType": "rdbms",
 "parameter": {
 "connection": [
 {
 "jdbcUrl": "jdbc:dm://ip:port/database",
 "table": [
 "table"
]
 }
],
 "username": "username",
 "password": "password",
 "table": "table",
 "column": [
 "id",
 "name"
],
 "preSql": [
 "delete from XXX;"
]
 },
 "name": "Writer",
```

```

 "category": "writer"
 }
},
"version": "2.0",
"order": {
 "hops": [
 {
 "from": "Reader",
 "to": "Writer"
 }
]
},
"setting": {
 "errorLimit": {
 "record": ""
 },
 "speed": {
 "concurrent": 2,
 "throttle": false
 }
}
}

```

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, 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"
]
}

```

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
```

### 8.3.21 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 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"
 }
]
}
}

```

### 8.3.22 Configure HybridDB for MySQL Writer

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

HybridDB for MySQL Writer allows you to write data to tables stored in HybridDB for MySQL databases. Specifically, HybridDB for MySQL Writer connects to a remote HybridDB for MySQL database through Java Database Connectivity (JDBC), and runs an `INSERT INTO` or `REPLACE INTO` statement to write data to the HybridDB for MySQL database. HybridDB for MySQL uses the InnoDB engine so that data is written to the database in batches.



#### Note:

You must configure a connection before configuring HybridDB for MySQL Writer. For more information, see [Configure a HybridDB for MySQL connection](#).

HybridDB for MySQL Writer is designed for data developers to import data from data warehouses to HybridDB for MySQL databases. HybridDB for MySQL Writer can also be used as a data migration tool by users such as database administrators (DBAs). HybridDB for MySQL Writer obtains data from a Data Integration reader.



#### Note:

A sync node that uses HybridDB for MySQL Writer must have at least the permission to run 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

Currently, HybridDB for MySQL Writer supports most HybridDB for MySQL data types. Make sure that your data types are supported.

The following table lists the data types supported by HybridDB for MySQL Writer.

Category	HybridDB for 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	BOOLEAN
Binary	TINYBLOB, MEDIUMBLOB, BLOB, LONGBLOB, and VARBINARY

## Parameters

Parameter	Description	Required	Default value
<b>datasource</b>	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None
<b>table</b>	The name of the destination table.	Yes	None
<b>writeMode</b>	The write mode. Valid values: insert into and replace into. <ul style="list-style-type: none"><li>replace into: If no primary key conflict or unique index conflict occurs, the action is the same as that of insert into.</li><li>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.</li></ul>	No	insert into

Parameter	Description	Required	Default value
<b>column</b>	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
<b>preSql</b>	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 UI, and multiple SQL statements in the code editor.	No	None
<b>postSql</b>	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.	No	None
<b>batchSize</b>	The number of data records to write at a time. Setting this parameter can greatly reduce the interactions between Data Integration and the HybridDB for MySQL 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 HybridDB for MySQL Writer by using the codeless UI

### 1. Configure the connections.

Configure the source and destination connections for the sync node.

The screenshot shows the 'Data Source' configuration window in DataWorks. It has two main sections: 'Source' and 'Destination'.  
**Source Section:**  
 - \* Data Source: HybridDB for MySQL (dropdown), HybridDB\_MySQL (text field with help icon)  
 - \* Table: Please select (dropdown)  
 - Filter: Enter a WHERE clause when you need to synchronize incremental data. Do not include the keyword WHERE. (text area with help icon)  
 - Shard Key: The table is sharded for concurrent reading. (text field with help icon)  
 - Preview button  
**Destination Section:**  
 - \* Data Source: HybridDB for MySQL (dropdown), HybridDB\_MySQL (text field with help icon)  
 - \* Table: Please select (dropdown)  
 - Statements Run Before Import: Enter SQL statements. These statements runs before the data is imported. (text area with help icon)  
 - Statements Run After Import: Enter SQL statements. These statements runs after the data is imported. (text area with help icon)  
 - \* Primary Key: INSERT INTO (dropdown)  
 - Violation (text field)

Parameter	Description
<b>Connection</b>	The <b>datasource</b> parameter in the preceding parameter description. Select a connection type, and enter the name of a connection that has been configured in DataWorks.
<b>Table</b>	The <b>table</b> parameter in the preceding parameter description.
<b>Statement Run Before Writing</b>	The <b>preSql</b> parameter in the preceding parameter description. Enter an SQL statement to run before the sync node is run.
<b>Statement Run After Writing</b>	The <b>postSql</b> parameter in the preceding parameter description. Enter an SQL statement to run after the sync node is run.
<b>Solution to Primary Key Violation</b>	The <b>writeMode</b> parameter in the preceding parameter description. Select the expected write mode.

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
<b>Map Fields with the Same Name</b>	Click <b>Map Fields with the Same Name</b> to establish a mapping between fields with the same name. Note that the data types of the fields must match.
<b>Map Fields in the Same Line</b>	Click <b>Map Fields in the Same Line</b> to establish a mapping for fields in the same row. Note that the data types of the fields must match.
<b>Delete All Mappings</b>	Click <b>Delete All Mappings</b> to remove mappings that have been established.
<b>Auto Layout</b>	Click Auto Layout. The fields are automatically sorted based on specified rules.

3. Configure channel control policies.

03 Channel

You can control the sync process by throttling the bandwidth or limiting the dirty data records allowed. [Learn more.](#)

\* Expected Concurrency: 2 ?

\* Bandwidth Throttling: ☒ Disable ☐ Enable

Dirty Data Records Allowed: Dirty data is allowed by default. dirty records, task ends. ?

Resource Group: Default resource group

Parameter	Description
<b>Expected Maximum Concurrency</b>	The maximum number of concurrent threads to read and write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
<b>Bandwidth Throttling</b>	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.
<b>Dirty Data Records Allowed</b>	The maximum number of dirty data records allowed.

Parameter	Description
<b>Resource Group</b>	The servers on which nodes are run. If an excessively large number of nodes are run on the default resource group, some nodes may be delayed due to insufficient resources. In this case, we recommend that you purchase an exclusive resource group for data integration or add a custom resource group. For more information, see <a href="#">#unique_15</a> and <a href="#">Add a custom resource group</a> .

96811

### Configure HybridDB for MySQL Writer by using the code editor

In the following code, a node is configured to write data to the HybridDB for MySQL database. For more information about the parameters, see the preceding parameter description.

```
{
 "type": "job",
 "steps": [
 {
 "parameter": {},
 {
 "parameter": {
 "postSql": [], // The SQL statement to run after the sync node is run.
 "datasource": "px_aliyun_hy***", // The connection name.
 "column": [// The columns to which data is written.
 "id",
 "name",
 "sex",
 "salary",
 "age",
 "pt"
],
 "writeMode": "insert", // The write mode.
 "batchSize": 256, // The number of data records to write at a time.
 "encoding": "UTF-8", // The encoding format.
 "table": "person_copy", // The name of the destination table.
 "preSql": [] // The SQL statement to run before the sync node is run.
 },
 "name": "Writer",
 "category": "writer"
 }
],
 "version": "2.0", // The version number.
 "order": {
 "hops": [
 {
 "from": "Reader",
 "to": "Writer"
 }
]
 },
 "setting": {
 "errorLimit": { // The maximum number of dirty data records allowed.
 "record": ""
 }
 }
 },
}
```

```

 "speed": {
 "concurrent": 7, // The number of concurrent threads.
 "throttle": true, // 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.
 "mbps": 1, // The maximum transmission rate.
 }
 }
}

```

### 8.3.23 Configure AnalyticDB for PostgreSQL Writer

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

AnalyticDB for PostgreSQL Writer allows you to write data to AnalyticDB for PostgreSQL databases. AnalyticDB for PostgreSQL Writer connects to a remote AnalyticDB for PostgreSQL database through Java Database Connectivity (JDBC), and runs an SQL statement to write data to the AnalyticDB for PostgreSQL database. On the public cloud, Relational Database Service (RDS) provides the AnalyticDB for PostgreSQL storage engine.



#### Note:

You must configure a connection before configuring AnalyticDB for PostgreSQL Writer. For more information, see [Configure an AnalyticDB for PostgreSQL Connection](#).

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

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

#### Data types

AnalyticDB for PostgreSQL Writer supports most AnalyticDB for PostgreSQL data types. Make sure that your data types are supported.

The following table lists the data types supported by AnalyticDB for PostgreSQL Writer.

Category	AnalyticDB for 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 and time	DATE, TIME, and TIMESTAMP
Boolean	BOOLEAN
Byte	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 identical to 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	<p>The write mode. Valid values: insert and copy.</p> <ul style="list-style-type: none"> <li>insert: runs the insert into...values ... statement to write data to the AnalyticDB for 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.</li> <li>copy: copies data between tables and the standard input or standard 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.</li> </ul>	No	insert
column	<p>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":["*"].</p>	Yes	None
preSql	<p>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 UI, and multiple SQL statements in the code editor.</p>	No	None

Parameter	Description	Required	Default value
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.	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 AnalyticDB for PostgreSQL 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 AnalyticDB for PostgreSQL Writer by using the codeless UI

#### 1. Configure the connections.

Configure the source and destination connections for the sync node.

Parameter	Description
<b>Connection</b>	The <b>datasource</b> parameter in the preceding parameter description. Select a connection type, and enter the name of a connection that has been configured in DataWorks.
<b>Table</b>	The <b>table</b> parameter in the preceding parameter description.
<b>Statement Run Before Writing</b>	The <b>preSql</b> parameter in the preceding parameter description. Enter an SQL statement to run before the sync node is run.
<b>Statement Run After Writing</b>	The <b>postSql</b> parameter in the preceding parameter description. Enter an SQL statement to run after the sync node is run.
<b>Write Method</b>	The <b>writeMode</b> parameter in the preceding parameter description. You can select <b>insert</b> or <b>copy</b> .

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
<b>Map Fields with the Same Name</b>	Click <b>Map Fields with the Same Name</b> to establish a mapping between fields with the same name. Note that the data types of the fields must match.
<b>Map Fields in the Same Line</b>	Click <b>Map Fields in the Same Line</b> to establish a mapping for fields in the same row. Note that the data types of the fields must match.
<b>Delete All Mappings</b>	Click <b>Delete All Mappings</b> to remove mappings that have been established.
<b>Auto Layout</b>	Click Auto Layout. The fields are automatically sorted based on specified rules.

3. Configure channel control policies.

03 Channel

You can control the sync process by throttling the bandwidth or limiting the dirty data records allowed. [Learn more.](#)

\* Expected Concurrency: 2 ?

\* Bandwidth Throttling: ☒ Disable ☐ Enable

Dirty Data Records Allowed: Dirty data is allowed by default. dirty records, task ends. ?

Resource Group: Default resource group

Parameter	Description
<b>Expected Maximum Concurrency</b>	The maximum number of concurrent threads to read and write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
<b>Bandwidth Throttling</b>	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.
<b>Dirty Data Records Allowed</b>	The maximum number of dirty data records allowed.



Parameter	Description
<b>Resource Group</b>	The servers on which nodes are run. If an excessively large number of nodes are run on the default resource group, some nodes may be delayed due to insufficient resources. In this case, we recommend that you purchase an exclusive resource group for data integration or add a custom resource group. For more information, see <a href="#">#unique_15</a> and <a href="#">Add a custom resource group</a> .

### Configure AnalyticDB for PostgreSQL Writer by using the code editor

```
{
 "type": "job",
 "steps": [
 {
 "parameter": {},
 "name": "Reader",
 "category": "reader"
 },
 {
 "parameter": {
 "postSql": [], // The SQL statement to run after the sync node is run.
 "datasource": "test_004", // The connection name.
 "column": [// The columns to which data is written.
 "id",
 "name",
 "sex",
 "salary",
 "age"
],
 "table": "public.person", // The name of the destination table.
 "preSql": [] // The SQL statement to run before the sync node is run.
 },
 "name": "Writer",
 "category": "writer"
 }
],
 "version": "2.0", // The version number.
 "order": {
 "hops": [
 {
 "from": "Reader",
 "to": "Writer"
 }
]
 },
 "setting": {
 "errorLimit": { // The maximum number of dirty data records allowed.
 "record": ""
 },
 "speed": {
 "concurrent": 6, // The number of concurrent threads.
 "throttle": false, // Specifies whether to enable bandwidth throttling.
 }
 }
}
```

```
}
```

### 8.3.24 Configure POLARDB Writer

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

POLARDB Writer allows you to write data to tables stored in POLARDB databases. Specifically, POLARDB Writer connects to a remote POLARDB database through Java Database Connectivity (JDBC), and runs an INSERT INTO or REPLACE INTO statement to write data to the POLARDB database. Internally, data is submitted to the POLARDB database in batches by using the InnoDB engine.

**Note:**

You must configure a connection before configuring POLARDB Writer. For more information, see [Configure a POLARDB connection](#).

POLARDB Writer is designed for extract-transform-load (ETL) developers to import data from data warehouses to POLARDB databases. POLARDB Writer can also be used as a data migration tool by users such as database administrators (DBAs). POLARDB Writer obtains data from a Data Integration reader, and writes the data to the destination database based on the writeMode parameter that you have configured.

**Note:**

A sync node that uses POLARDB Writer must have at least the permission to run 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

Similar to POLARDB Reader, POLARDB Writer supports most POLARDB data types. Make sure that your data types are supported.

The following table lists the data types supported by POLARDB Writer.

Category	POLARDB data type
Integer	INT, TINYINT, SMALLINT, MEDIUMINT, BIGINT, and YEAR
Floating point	FLOAT, DOUBLE, and DECIMAL
String	VARCHAR, CHAR, TINYTEXT, TEXT, MEDIUMTEXT, and LONGTEXT

Category	POLARDB data type
Date and time	DATE, DATETIME, TIMESTAMP, and TIME
Boolean	BOOLEAN
Binary	TINYBLOB, MEDIUMBLOB, BLOB, LONGBLOB, and VARBINARY

## Parameters

Parameter	Description	Required	Default value
<b>datasource</b>	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None
<b>table</b>	The name of the destination table.	Yes	None
<b>writeMode</b>	<p>The write mode. Valid values: <b>insert</b>, <b>replace</b>, and on duplicate key update.</p> <ul style="list-style-type: none"> <li>• <b>REPLACE INTO</b>: If no primary key conflict or unique index conflict occurs, the action is the same as that of <b>INSERT INTO</b>. If a conflict occurs, original rows are deleted and new rows are inserted.</li> <li>• <b>INSERT INTO</b>: If a primary key conflict or unique index conflict occurs, data cannot be written to the conflicting rows and is regarded as dirty data.</li> <li>• <b>INSERT INTO table (a,b,c) VALUES (1, 2,3) ON DUPLICATE KEY UPDATE...:</b> If no primary key conflict or unique index conflict occurs, the action is the same as that of <b>INSERT INTO</b>. If a conflict occurs, specified fields in original rows are updated.</li> </ul>	No	insert
<b>column</b>	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 a space character 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
<b>preSql</b>	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 UI, and multiple SQL statements in the code editor.	No	None
<b>postSql</b>	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.	No	None
<b>batchSize</b>	The number of data records to write at a time. Setting this parameter can greatly reduce the interactions between Data Integration and the POLARDB 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 POLARDB Writer by using the codeless UI

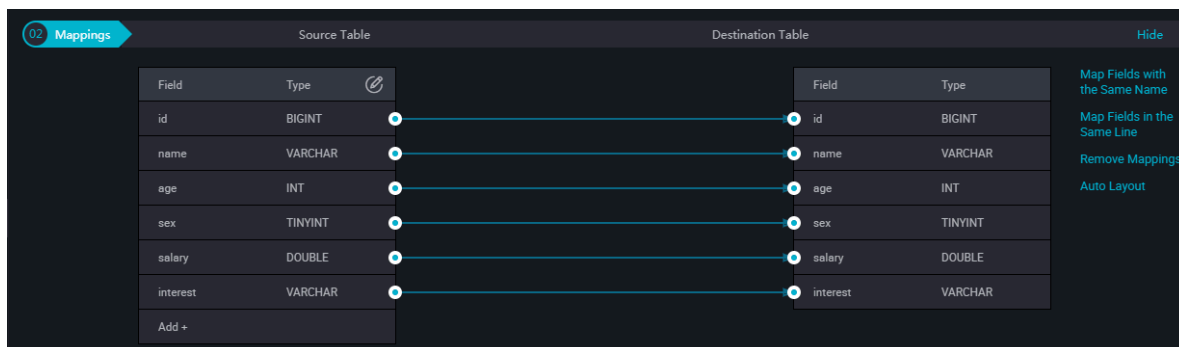
### 1. Configure the connections.

Configure the source and destination connections for the sync node.

The screenshot shows the 'Data Source' configuration for a POLARDB Writer. The 'Source' tab is active, displaying fields for 'Data Source' (POLARDB), 'Table' (polaris\_person), 'Filter' (a text area with instructions), and 'Shard Key' (id). The 'Destination' tab is also visible, showing fields for 'Data Source' (POLARDB), 'Table' (polaris\_person), 'Statements Run Before Import' (a text area), 'Statements Run After Import' (a text area), and 'Primary Key' (INSERT INTO). A 'Preview' button is located at the bottom of the 'Source' tab.

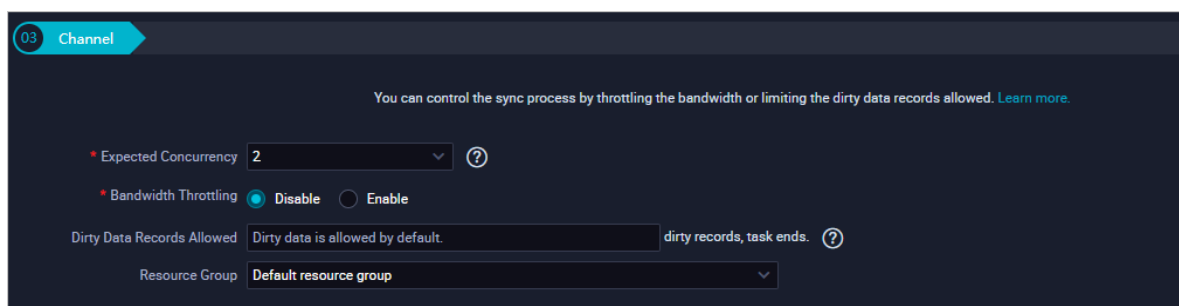
Parameter	Description
<b>Connection</b>	The <b>datasource</b> parameter in the preceding parameter description. Select a connection type, and enter the name of a connection that has been configured in DataWorks.
<b>Table</b>	The <b>table</b> parameter in the preceding parameter description.
<b>Statement Run Before Writing</b>	The <b>preSql</b> parameter in the preceding parameter description. Enter an SQL statement to run before the sync node is run.
<b>Statement Run After Writing</b>	The <b>postSql</b> parameter in the preceding parameter description. Enter an SQL statement to run after the sync node is run.
<b>Solution to Primary Key Violation</b>	The <b>writeMode</b> parameter in the preceding parameter description. Select the expected write mode.

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.



Parameter	Description
<b>Map Fields with the Same Name</b>	Click <b>Map Fields with the Same Name</b> to establish a mapping between fields with the same name. Note that the data types of the fields must match.
<b>Map Fields in the Same Line</b>	Click <b>Map Fields in the Same Line</b> to establish a mapping for fields in the same row. Note that the data types of the fields must match.
<b>Delete All Mappings</b>	Click <b>Delete All Mappings</b> to remove mappings that have been established.
<b>Auto Layout</b>	Click Auto Layout. The fields are automatically sorted based on specified rules.

3. Configure channel control policies.



Parameter	Description
<b>Expected Maximum Concurrency</b>	The maximum number of concurrent threads to read and write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.

Parameter	Description
<b>Bandwidth Throttling</b>	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.
<b>Dirty Data Records Allowed</b>	The maximum number of dirty data records allowed.
<b>Resource Group</b>	The servers on which nodes are run. If an excessively large number of nodes are run in the default resource group, some nodes may be delayed due to insufficient resources. In this case, we recommend that you purchase an exclusive resource group for data integration or add a custom resource group. For more information, see <a href="#">#unique_15</a> and <a href="#">Add a custom resource group</a> .

### Configure POLARDB Writer by using the code editor

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

```
{
 "type": "job",
 "steps": [
 {
 "parameter": {},
 "name": "Reader",
 "category": "reader"
 },
 {
 "parameter": {
 "postSql": [], // The SQL statement to run after the sync node is run.
 "datasource": "test_005", // The connection name.
 "column": [// The columns to which data is written.
 "id",
 "name",
 "age",
 "sex",
 "salary",
 "interest"
],
 "writeMode": "insert", // The write mode.
 "batchSize": 256, // The number of data records to write at a time.
 "encoding": "UTF-8", // The encoding format.
 "table": "POLARDB_person_copy", // The name of the destination table.
 "preSql": [] // The SQL statement to run before the sync node is run.
 },
 "name": "Writer",
 "category": "writer"
 }
],
 "version": "2.0", // The version number.
 "order": {
 "hops": [
```

```
{
 "from": "Reader",
 "to": "Writer"
}
],
"setting": {
 "errorLimit": { // The maximum number of dirty data records allowed.
 "record": ""
 },
 "speed": {
 "concurrent": 6, // The number of concurrent threads.
 "throttle": false, // Indicates whether to throttle the transmission rate.
 }
}
}
```

### 8.3.25 Configure TSDB Writer

TSDB Writer allows you to write data points to Time Series Database (TSDB) databases developed by Alibaba.

TSDB is a high-performance, cost-effective, stable, and reliable online database service. It features high read/write performance and high compression ratio for data storage. It also enables the interpolation and aggregation of time series data. TSDB is applied in various scenarios, such as the device monitoring system of IoT, energy management system (EMS) for enterprises, security monitoring system in production, and electricity detection system.

TSDB provides you with the capability to write millions of data points within seconds, together with the benefits of high compression ratio, low-cost data storage, downsampling, interpolation, multi-dimensional aggregation, and visualized query results. These features help you resolve issues such as high storage cost and low writing and query efficiency caused by large amounts of data-collecting points on devices and high frequency of the data collection.

Currently, you can only configure TSDB Writer in the code editor. For more information about TSDB, see [What is TSDB](#).

#### How it works

TSDB Writer connects to a TSDB instance through HTTP and writes data points through the `/api/put` operation.

#### Limits


Currently, only TSDB 2.4.x and later are supported.





**Data types**

Category	Data Integration data type	TSDB data type
String	STRING	TIMESTAMP, METRIC, TAGS, and VALUE

**Parameters**

Source data store	Parameter	Description	Required	Default value
TSDB and RDB	<b>sourceDbType</b>	The type of the source data store.	No	TSDB <div>  <b>Note:</b>  Currently, the valid values include TSDB and RDB. TSDB refers to OpenTSDB, InfluxDB, Prometheus, and TimeScale databases. RDB refers to relational databases such as MySQL, Oracle, PostgreSQL, and Distributed Relational Database Service (DRDS) databases. </div>
	<b>endpoint</b>	The endpoint for connecting to the destination TSDB database through HTTP.	Yes. The endpoint is in the http://IP:Port format.	None
	<b>batchSize</b>	The number of data records to write at a time.	No. The data type is INT and the value must be greater than 0.	100

Source data store	Parameter	Description	Required	Default value
	<b>maxRetryTime</b>	The maximum number of retries allowed after a failure.	No. The data type is INT and the value must be greater than 1.	3
	<b>ignoreWriteError</b>	Specifies whether to ignore write errors. If you set this parameter to true, TSDB Writer continues to perform the write operation when a write error occurs. If the write operation fails after multiple retries, the sync node ends.	No. The data type is BOOLEAN.	false
RDB	<b>endpoint</b>	The endpoint for connecting to the destination TSDB database through HTTP.	Yes. The endpoint is in the http://IP:Port format.	None
	<b>column</b>	The names of fields in the relational database.	Yes	None <div>  <b>Note:</b>  The order of the fields specified here must be the same as the order of the fields configured in TSDB Reader. </div>

Source data store	Parameter	Description	Required	Default value
	<b>columnType</b>	<p>The types of the fields in the destination TSDB database. The supported types are as follows:</p> <ul style="list-style-type: none"> <li>• <b>timestamp</b>: a timestamp.</li> <li>• <b>tag</b>: a tag.</li> <li>• <b>metric_num</b>: a metric whose value is a number.</li> <li>• <b>metric_string</b>: a metric whose value is a string.</li> </ul>	Yes	<p>None</p> <div>  <b>Note:</b>            The order of the fields specified here must be the same as the order of the fields configured in TSDB Reader.         </div>
	<b>batchSize</b>	The number of data records to write at a time.	No. The data type is INT and the value must be greater than 0.	100

### Configure TSDB Writer by using the codeless UI

Currently, the codeless user interface (UI) is not supported for TSDB Writer.

### Configure TSDB Writer by using the code editor

In the following code, a node is configured to write data to a TSDB database.

```
```json
{
  "order": {
    "hops": [
      {
        "from": "Reader",
        "to": "Writer"
      }
    ]
  },
  "setting": {
    "errorLimit": {
      "record": "0"
    },
    "speed": {
      "concurrent": 1,
```

```

    "throttle": true
  }
},
"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 report

- Performance data features
 - Metric: a metric.
 - tag_k and tag_v: the keys and values. The first four rows constitute a timeline of 2,000,000 data points, that is, 10 (zones) × 20 (clusters) × 100 (groups) × 100 (apps). The ip in the last row corresponds to the index of the 2,000,000 data points, starting from 1.

tag_k	tag_v
zone	z1 to z10
cluster	c1 to c20
group	g1 to g100
app	a1 to a100

tag_k	tag_v
ip	ip1 to ip2,000,000

- value: a measured value, which is random in the range of 1 to 100.
- interval: a collection period of 10 seconds. If data collection lasts for 3 hours, a total of 2,160,000,000 data points, that is $3 \times 60 \times 60 / 10 \times 2,000,000$, are collected.
- Performance test results

Number of channels	Data integration speed (records per second)	Data integration traffic (Mbit/s)
1	129,753	15.45
2	284,953	33.70
3	385,868	45.71

8.3.26 Configure AnalyticDB for MySQL 3.0 Writer

This topic describes the data types and parameters supported by AnalyticDB for MySQL 3.0 Writer and how to configure it by using the codeless user interface (UI) and code editor.

You must configure a connection before configuring AnalyticDB for MySQL 3.0 Writer.



Data types


The following table lists the data types supported by AnalyticDB for MySQL 3.0 Writer.

Category	AnalyticDB for MySQL 3.0 data type
Integer	INT, INTEGER, TINYINT, SMALLINT, and BIGINT
Floating point	FLOAT, DOUBLE, and DECIMAL
String	VARCHAR
Date and time	DATE, DATETIME, TIMESTAMP, and TIME
Boolean	BOOLEAN

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 destination table.	Yes	None

Parameter	Description	Required	Default value
writeMode	<p>The write mode. Valid values: insert into and replace into.</p> <ul style="list-style-type: none"> 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. 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 into
column	<p>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":["*"].</p> <div>  Note: If the field name contains select, enclose the field name in grave accents (` `). Example: <code>`item_select_no`</code>. </div>	Yes	None
preSql	<p>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 UI, and multiple SQL statements in the code editor.</p> <div>  Note: If you specify multiple SQL statements in the code editor, the system does not guarantee that they are run in the same transaction. </div>	No	None

Parameter	Description	Required	Default value
postSql	<p>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.</p> <div>  Note: If you specify multiple SQL statements in the code editor, the system does not guarantee that they are run in the same transaction. </div>	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 AnalyticDB for MySQL 3.0 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 AnalyticDB for MySQL 3.0 Writer 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.
Statement Run Before Writing	The preSql parameter in the preceding parameter description. Enter an SQL statement to run before the sync node is run.
Statement Run After Writing	The postSql parameter in the preceding parameter description. Enter an SQL statement to run after the sync node is run.
Solution to Primary Key Violation	The writeMode parameter in the preceding parameter description. Select the expected write mode.

Parameter	Description
Data Records per Write	The number of data records to write at a time. The batchSize parameter in the preceding parameter description. This parameter takes effect only when writeMode is set to insert into.

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.
3. Configure channel control policies.

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read 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 servers on which nodes are run. If an excessively large number of nodes are run in the default resource group, some nodes may be delayed due to insufficient resources. In this case, we recommend that you purchase an exclusive resource group for data integration or add a custom resource group. For more information, see #unique_15 and Add a custom resource group .

Configure AnalyticDB for MySQL 3.0 Writer by using the code editor

In the following code, a node is configured to write data to an AnalyticDB for MySQL 3.0 database. For more information about the parameters, see the preceding parameter description.

```
{
  "type": "job",
  "steps": [
    {
      "stepType": "stream",
      "parameter": {},
      "name": "Reader",
      "category": "reader"
    },
    {
      "stepType": "analyticdb_for_mysql", // The writer type.
      "parameter": {
        "postSql": [], // The SQL statement to run after the sync node is run.
        "tableType": null, // The reserved field. Default value: null.
        "datasource": "hangzhou_ads", // The connection name.
        "column": [ // The columns in the destination table to which data is written.
          "id",
          "value"
        ],
        "guid": null,
        "writeMode": "insert", // The write mode. For more information, see the
description of the writeMode parameter.
        "batchSize": 2048, // The number of data record to write at a time. For more
information, see description of the batchSize parameter.
        "encoding": "UTF-8", // The encoding format.
        "table": "t5", // The name of the destination table.
        "preSql": [] // The SQL statement to run before the sync node is run.
      },
      "name": "Writer",
      "category": "writer"
    }
  ],
  "version": "2.0", // The version number.
  "order": {
    "hops": [
      {
        "from": "Reader",
        "to": "Writer"
      }
    ]
  },
  "setting": {
    "errorLimit": {
      "record": "0" // The maximum number of dirty data records allowed.
    },
    "speed": {
      "concurrent": 2, // 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
parameter to true.
    }
  }
}
```

```
}
```

8.3.27 Configure GDB Writer

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

Graph Database (GDB) is a real-time and reliable online database service that makes it easy to store and navigate the relationships between highly connected datasets. GDB supports the Property Graph model and its query language Apache TinkerPop Gremlin, allowing you to easily build queries that efficiently navigate highly connected datasets.

**Note:**

- You must configure a connection before configuring GDB Writer.
- You must configure two sync nodes to synchronize data about vertices and edges separately.

Limits

- You must run a sync node to synchronize vertex data before running a sync node to synchronize edge data.
- Limits on vertices:
 - A vertex must have a vertex name that is specified by the label parameter.
 - A vertex must have a unique primary key of the STRING type. If the primary key is not a string, GDB Writer forcibly converts the data to a string.
 - Exercise caution when you set the **idTransRule** parameter. If you set this parameter to None, make sure that the primary key of each vertex is unique among global vertices.

- Limits on edges:
 - An edge must have an edge name that is specified by the label parameter.
 - The primary key is optional for an edge.
 - If you specify a primary key for an edge, make sure that the ID is unique among global edges.
 - If no primary key is specified, GDB Writer automatically generates a universally unique identifier (UUID) of the STRING type for the edge. If the UUID is not a string, GDB Writer forcibly converts the data to a string.
 - Exercise caution when you set the **idTransRule** parameter. If you set this parameter to None, make sure that the primary key of each edge is unique among global edges.
 - The **srcIdTransRule** and **dstIdTransRule** parameters are required for an edge, and their values must be the same as that of the **idTransRule** parameter for a vertex.
- Unless otherwise specified, field names or enumerated values in this topic are case-sensitive.
- Currently, GDB Writer only supports UTF-8 encoding. Source data must be encoded in UTF-8.
- Sync nodes for data integration can only run on exclusive resource groups due to network constraints. Purchase an exclusive resource group for data integration and bind it to a Virtual Private Cloud (VPC) where the GDB instance resides. Scheduling nodes can run on the default resource group.

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
label	The label, that is, the name of the vertex or edge. GDB Writer can obtain labels from columns in the source table. For example, if you set this parameter to \${0}, GDB Writer uses the value of the first column as the label. The column index starts from 0.	Yes	None

Parameter	Description	Required	Default value
labelType	<p>The type of the label. Valid values: VERTEX and EDGE.</p> <ul style="list-style-type: none"> • VERTEX: a vertex. • EDGE: an edge. 	Yes	None
srcLabel	<ul style="list-style-type: none"> • The name of the start vertex in an edge when the labelType parameter is set to EDGE. <p>In this case, this parameter can be left empty if srcIdTransRule is set to None. If srcIdTransRule is set to another value, this parameter is required.</p> <ul style="list-style-type: none"> • Leave this parameter empty if the labelType parameter is set to VERTEX. 	No	None
dstLabel	<ul style="list-style-type: none"> • The name of the end vertex in an edge when the labelType parameter is set to EDGE. <p>In this case, this parameter can be left empty if dstIdTransRule is set to None. If dstIdTransRule is set to another value, this parameter is required.</p> <ul style="list-style-type: none"> • Leave this parameter empty if the labelType parameter is set to VERTEX. 	No	None
writeMode	<p>The mode in which GDB Writer processes data records with duplicate primary keys. Valid values: INSERT and MERGE.</p> <ul style="list-style-type: none"> • INSERT: returns an error message. The number of error data records is increased by 1. • MERGE: overrides the existing data record with the new one. 	Yes	INSERT

Parameter	Description	Required	Default value
idTransRule	<p>The rule for converting the primary key. Valid values: labelPrefix and None.</p> <ul style="list-style-type: none"> labelPrefix: converts the primary key to the {label}-{column in source} format. None: does not convert the primary key. 	Yes	None
srcIdTransRule	<p>The rule for converting the primary key of the start vertex when the labelType parameter is set to EDGE. Valid values: labelPrefix and None.</p> <ul style="list-style-type: none"> labelPrefix: converts the primary key to the {label}-{column in source} format. None: does not convert the primary key. In this case, the srcLabel parameter can be left empty. 	Required when the labelType parameter is set to EDGE	None
dstIdTransRule	<p>The rule for converting the primary key of the end vertex when the labelType parameter is set to EDGE. Valid values: labelPrefix and None.</p> <ul style="list-style-type: none"> labelPrefix: converts the primary key to the {label}-{column in source} format. None: does not convert the primary key. In this case, the dstLabel parameter can be left empty. 	Required when the labelType parameter is set to EDGE	None

Parameter	Description	Required	Default value
column	<p>The mappings between columns in the source and vertices or edges.</p> <ul style="list-style-type: none"> • name: the name of the vertex or edge property. • value: the value of the vertex or edge property. You can customize a value only in the code editor. <ul style="list-style-type: none"> - <code>\${N}</code>: uses the value of the Nth column in the source as the value of the vertex or edge property. N indicates the column index, which starts from 0. - <code>\${0}</code>: uses the value of the first column in the source as the value of the vertex or edge property. - <code>test-\${0}</code>: appends a fixed string such as test- before or after <code>\${0}</code>. - <code>\${0}-\${1}</code>: combines the values of multiple columns in the source as the value of vertex or edge property. You can also add fixed strings at any locations, for example, test-\${0}-test1-\${1}-test2. • type: the data type of the vertex or edge property. <p>The primary key can only be of the STRING type. If the value obtained from the source is not a string, GDB Writer forcibly converts the data to a string. Make sure that the value can be converted to a string.</p> <p>Other properties can be of the INT, LONG, FLOAT, DOUBLE, BOOLEAN, or STRING type.</p> <ul style="list-style-type: none"> • columnType: the category of the vertex or edge property. <ul style="list-style-type: none"> - For both the vertex and edge <p>primaryKey: the primary key.</p> - For the vertex <ul style="list-style-type: none"> ■ vertexProperty: a common property of the vertex. 	Yes	None
538	<ul style="list-style-type: none"> - For the vertex <ul style="list-style-type: none"> ■ vertexProperty: a common property of the vertex. 		Issue: 20200628

Configure GDB Writer by using the code editor

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

- Configure a sync node to write data about vertices to a GDB database

```
{
  "order":{
    "hops":[
      {
        "from":"Reader",
        "to":"Writer"
      }
    ]
  },
  "setting":{
    "errorLimit":{
      "record":"100" // The maximum number of dirty data records allowed.
    },
    "jvmOption":"",
    "speed":{
      "concurrent":3,
      "throttle":false
    }
  },
  "steps":[
    {
      "category":"reader",
      "name":"Reader",
      "parameter":{
        "column":[
          "*"
        ],
        "datasource":"_ODPS",
        "emptyAsNull":true,
        "guid":"",
        "isCompress":false,
        "partition":[],
        "table":""
      },
      "stepType":"odps"
    },
    {
      "category":"writer",
      "name":"Writer",
      "parameter": {
        "datasource": "testGDB", // The connection name.
        "label": "person", // The label, that is, the name of the vertex.
        "srcLabel": "", // Use the default setting.
        "dstLabel": "", // Use the default setting.
        "labelType": "VERTEX", // The type of the label. The value of VERTEX indicates
a vertex.
        "writeMode": "INSERT", // The mode in which GDB Writer processes data
records with duplicate primary keys.
        "idTransRule": "labelPrefix", // The rule for converting the primary key of the
vertex.
        "srcIdTransRule": "none", // Use the default setting.
        "dstIdTransRule": "none", // Use the default setting.
        "column": [
          {
```

```

        "name": "id", // The name of the vertex property.
        "value": "${0}", // The value of the first column in the source is used as the
        value of the vertex property. If multiple columns are specified, they can be concatenat
        ed. In this example, 0 is the column index.
        "type": "string", // The data type of the vertex property.
        "columnType": "primaryKey" // The category of the vertex property. The
        value of primaryKey indicates the primary key.
    }, // The primary key of the vertex. The value must be an ID of the STRING
    type, and the record must exist.
    {
        "name": "person_age",
        "value": "${1}", // The value of the second column in the source is used
        as the value of the vertex property. If multiple columns are specified, they can be
        concatenated.
        "type": "int",
        "columnType": "vertexProperty" // The category of the vertex property.
        The value of vertexProperty indicates a common vertex property.
    }, // A common property of the vertex. The value can be of the INT, LONG,
    FLOAT, DOUBLE, BOOLEAN, or STRING type.
    {
        "name": "person_credit",
        "value": "${2}", // The value of the third column in the source is used
        as the value of the vertex property. If multiple columns are specified, they can be
        concatenated.
        "type": "string",
        "columnType": "vertexProperty"
    }, // A common property of the vertex.
    ]
    }
    "stepType": "gdb"
  }
  ],
  "type": "job",
  "version": "2.0"
}

```

- Configure a sync node to write data about edges to a GDB database

```

{
  "order": {
    "hops": [
      {
        "from": "Reader",
        "to": "Writer"
      }
    ]
  },
  "setting": {
    "errorLimit": {
      "record": "100" // The maximum number of dirty data records allowed.
    },
    "jvmOption": "",
    "speed": {
      "concurrent": 3,
      "throttle": false
    }
  },
  "steps": [
    {
      "category": "reader",
      "name": "Reader",
      "parameter": {
        "column": [

```



```

    },
    "datasource": "_ODPS",
    "emptyAsNull": true,
    "guid": "",
    "isCompress": false,
    "partition": [],
    "table": ""
  },
  "stepType": "odps"
},
{
  "category": "writer",
  "name": "Writer",
  "parameter": {
    "datasource": "testGDB", // The connection name.
    "label": "use", // The label, that is, the name of the edge.
    "labelType": "EDGE", // The type of the label. The value of EDGE indicates an
edge.
    "srcLabel": "person", // The name of the start vertex in the edge.
    "dstLabel": "software", // The name of the end vertex in the edge.
    "writeMode": "INSERT", // The mode in which GDB Writer processes data
records with duplicate primary keys.
    "idTransRule": "labelPrefix", // The rule for converting the primary key of the
edge.
    "srcIdTransRule": "labelPrefix", // The rule for converting the primary key of
the start vertex.
    "dstIdTransRule": "labelPrefix", // The rule for converting the primary key of
the end vertex.
    "column": [
      {
        "name": "id", // The name of the edge property.
        "value": "${0}", // The value of the first column in the source is used as the
value of the edge property. If multiple columns are specified, they can be concatenat
ed.
        "type": "string", // The data type of the edge property.
        "columnType": "primaryKey" // The category of the edge property. The
value of primaryKey indicates the primary key.
      }, // The primary key of the edge. The value must be an ID of the STRING
type, and the record must exist.
      {
        "name": "id",
        "value": "${1}", // The value of the second column in the source is used
as the value of the edge property. If multiple columns are specified, they can be
concatenated. The mapping rule must be the same as that configured when you
import the vertex.
        "type": "string",
        "columnType": "srcPrimaryKey" // The category of the edge property. The
value of srcPrimaryKey indicates the primary key of the start vertex.
      }, // The primary key of the start vertex. The value must be an ID of the
STRING type, and the record must exist.
      {
        "name": "id",
        "value": "${2}", // The value of the third column in the source is used
as the value of the edge property. If multiple columns are specified, they can be
concatenated. The mapping rule must be the same as that configured when you
import the vertex.
        "type": "string",
        "columnType": "dstPrimaryKey" // The category of the edge property. The
value of dstPrimaryKey indicates the primary key of the end vertex.
      }, // The primary key of the end vertex. The value must be an ID of the
STRING type, and the record must exist.
      {
        "name": "person_use_software_time",

```

```

        "value": "${3}", // The value of the fourth column in the source is used
        as the value of the edge property. If multiple columns are specified, they can be
        concatenated.
        "type": "long",
        "columnType": "edgeProperty" // The category of the edge property. The
        value of edgeProperty indicates a common edge property.
    }, // A common property of the edge. The value can be of the INT, LONG,
    FLOAT, DOUBLE, BOOLEAN, or STRING type.
    {
        "name": "person_regist_software_name",
        "value": "${4}", // The value of the fifth column in the source is used
        as the value of the edge property. If multiple columns are specified, they can be
        concatenated.
        "type": "string",
        "columnType": "edgeProperty"
    }, // A common property of the edge.
    {
        "name": "id",
        "value": "${5}", // The value of the sixth column in the source is used
        as the value of the edge property. If multiple columns are specified, they can be
        concatenated.
        "type": "long",
        "columnType": "edgeProperty"
    }, // A common property of the edge. The value is an ID. Different from the
    primary key, this property is optional.
    ]
    }
    "stepType": "gdb"
  }
  ],
  "type": "job",
  "version": "2.0"
}

```

8.3.28 Configure Hive Writer

Hive Writer allows you to write data to Hadoop Distributed File System (HDFS) and load the data to Hive. This topic describes how Hive Writer works, its parameters, and how to configure it by using the code editor.

Hive is a Hadoop-based data warehouse tool used to process large amounts of structured logs. Hive maps structured data files to a table and allows users to run SQL statements to query data in the table.

Essentially, Hive convert Hive Query Language (HQL) or SQL statements to MapReduce programs.

- Hive stores processed data in an HDFS.
- Hive uses MapReduce programs to analyze data at the underlying layer.
- Hive runs MapReduce programs on Yarn.



How it works

Hive Writer accesses the Hive metadata database, parses the configuration to obtain the file storage path, file format, and column delimiter of the HDFS file to which data is written, and writes data to the HDFS file.

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
jdbcUrl	The Java Database Connectivity (JDBC) URL of the Hive metadata database. Currently, Hive Writer can only access Hive metadata databases of the MySQL type. Make sure that the sync node is connected to the Hive metadata database over a network and has access permissions on the metadata database.	Yes	None
username	The username for connecting to the Hive metadata database.	Yes	None
password	The password for connecting to the Hive metadata database.	Yes	None
column	The columns to which data is written. Example: "column": ["id", "name"]. <ul style="list-style-type: none">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 parameter must explicitly specify a set of columns to which data is written. The parameter cannot be left empty.	Yes	None

Parameter	Description	Required	Default value
table	<p>The name of the Hive table to which data is written.</p> <div>  Note: The name is case-sensitive. </div>	Yes	None
partition	<ul style="list-style-type: none"> 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
writeMode	<p>The mode in which data is loaded to the Hive table. After data is written to the HDFS file, Hive Writer runs the LOAD DATA INPATH (overwrite) INTO TABLE statement to load data to the Hive table. The writeMode parameter specifies the data upload mode.</p> <ul style="list-style-type: none"> 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. <div>  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. </div>	Yes	None

Parameter	Description	Required	Default value
hiveConfig	<p>The extended parameters for Hive, including hiveCommand, jdbcUrl, username, and password.</p> <ul style="list-style-type: none"> hiveCommand: the full path of the Hive client. After you run the <code>hive -e</code> command, the <code>LOAD DATA INPATH</code> statement is run to load data based on the mode specified by the writeMode parameter. <p>The client specified by the hiveCommand parameter provides access information about Hive.</p> <ul style="list-style-type: none"> jdbcUrl, username, and password: the information for accessing Hive through JDBC. After accessing Hive through JDBC, Hive Writer runs the <code>LOAD DATA INPATH</code> statement to load data based on the mode specified by the writeMode parameter. <pre>"hiveConfig": { "hiveCommand": "", "jdbcUrl": "", "username": "", "password": "" }</pre>	Yes	None

Configure Hive Writer by using the code editor

In the following code, a node is configured to write data to Hive in JSON format.

```
{
  "order": {
    "hops": [
      {
        "from": "Reader",
        "to": "Writer"
      }
    ]
  },
  "setting": {
    "errorLimit": {
      "record": "0"
    },
    "speed": {
      "concurrent": 1,
      "throttle": false
    }
  }
}
```

```

    },
    "steps": [
      {
        "category": "reader",
        "name": "Reader",
        "parameter": {},
        "stepType": "stream", // The writer type.
      },
      {
        "category": "writer",
        "name": "Writer",
        "parameter": {
          "username": "",
          "password": "",
          "jdbcUrl": "jdbc:mysql://host:port/database",
          "table": "",
          "partition": "", // The partition name of the destination table.
          "column": [
            "id",
            "name"
          ],
          "writeMode": "append", // The write mode.
          "hiveConfig": {
            "hiveCommand": "",
            "jdbcUrl": "",
            "username": "",
            "password": ""
          }
        },
        "stepType": "hive"
      }
    ],
    "type": "job",
    "version": "2.0"
  }
}

```

8.3.29 Configure MaxCompute Writer

This topic describes the data types and parameters supported by MaxCompute Writer and how to configure it by using the codeless user interface (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 of the GB or TB level to MaxCompute.



Note:

You must configure a connection before configuring MaxCompute Writer. For more information, see [#unique_70](#). For more information about MaxCompute, see [#unique_177](#).

Based on the specified information such as the source project, table, partition, and field, MaxCompute Writer writes data to MaxCompute through Tunnel. For more information about common Tunnel commands, see [Tunnel commands](#).

Data Integration reads data from tables with strict schemas in a source database such as MySQL or MaxCompute to the memory, and converts the obtained data to the format supported by the destination data store before writing data to the destination data store.

If the data conversion fails or the data fails to be written to the destination data store, the data is regarded as dirty data. You can process the dirty data based on the maximum number of dirty data records allowed.

**Note:**

If data records contain the null value, MaxCompute Writer does not support data of the VARCHAR type.

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 destination table. The name is case -insensitive. You can specify only one table as the destination table.	Yes	None
partition	<p>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 last-level partition information, such as <code>pt=20150101, type=1, biz=2</code>.</p> <ul style="list-style-type: none">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

Parameter	Description	Required	Default value
column	<p>The columns in the destination table to which data is written. 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": ["*"]. 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 a comma (,). Example: "column": ["id","name"].</p> <ul style="list-style-type: none"> MaxCompute Writer supports filtering columns and changing 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 as follows: "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	<p>To guarantee the idempotence of write operations, set the value to true. That is, set the truncate parameter as follows: "truncate": "true". When a failed sync node is rerun due to a write failure, MaxCompute Writer deletes the data that has been written before importing the source data again. This guarantees that the same data is written for each rerun.</p> <p>MaxCompute Writer uses MaxCompute SQL to delete data. MaxCompute SQL cannot guarantee 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.</p> <p>To avoid this issue, we recommend that you do not run concurrent Data Definition Language (DDL) nodes to write data to the same partition. You can create different partitions for nodes that need to run concurrently.</p>	Yes	None


Configure MaxCompute Writer by using the codeless UI

1. Configure the connections.

Configure the source and destination connections for the sync node.

The screenshot shows the '01 Data Source' configuration window. It has two tabs: 'Source' and 'Destination'.
 Under the 'Source' tab:
 - * Data Source: Oracle (dropdown)
 - * Table: Please select (dropdown)
 - Data Filtering: `id=1` (text input)
 - Sharding Key: Based on this key, data is sharded for concurrent re. (text input)
 - A 'Preview' button is at the bottom.
 Under the 'Destination' tab:
 - * Data Source: ODPS (dropdown)
 - * Table: `table_name` (dropdown)
 - * Partition: `dt = ${bizdate}` (text input)
 - Clearance Rule: Clear Existing Data Before Writing (Insert Over... (dropdown)
 - Compression: ☒ Disable ☐ Enable
 - Consider Empty String as Null: ☐ Yes ☒ No

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.
Partition Key Column	<p>To write data to all the columns in the destination table, enter "column": ["*"]. The partition parameter supports wildcards and includes one or more partitions.</p> <ul style="list-style-type: none"> "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. <p>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.</p> <ul style="list-style-type: none"> To write data to all partitions, enter pt=*. To write data to some of the partitions, specify the corresponding dates.

Parameter	Description
Writing Rule	<ul style="list-style-type: none"> • 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 <code>INSERT OVERWRITE</code> statement. • Write with Original Data Retained (Insert Into): No data is deleted before data import. New data is always appended with each run. This rule is equivalent to the <code>INSERT INTO</code> statement. <div>  Note: <ul style="list-style-type: none"> • 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 through Tunnel, instead of by using the <code>INSERT INTO</code> statement. You can view the complete data in the destination table only after a sync node is run. Pay attention to the node dependencies. </div>
Convert Empty Strings to Null	Default value: Yes.

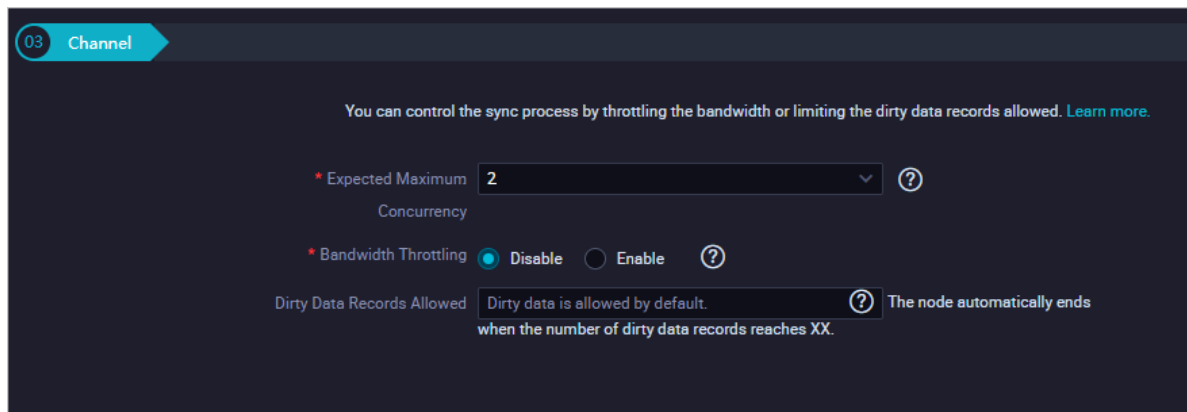
2. Configure field mapping, that is, the **column** parameter in the preceding parameter description.

Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right.

02 Mappings			Source Table	Target Table	Hide
Field	Type		Field	Type	
bizdate	DATE	○	age	BIGINT	
region	VARCHAR	○	job	STRING	
pv	BIGINT	○	marital	STRING	
uv	BIGINT	○	education	STRING	
browse_size	BIGINT	○	default	STRING	
Add +			housing	STRING	

Map Fields with the Same Name
Map Fields in the Same Line
Delete All Mappings
Auto Layout

3. Configure channel control policies.



Configure MaxCompute Writer by using the code editor

In the following code, a node is configured to write data to a MaxCompute table. 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 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"
    }
  ]
}
}
```

If you want to specify the [Tunnel endpoint](#) of MaxCompute, you can configure the connection in the code editor. To configure the connection, replace "datasource": "", in the preceding code with detailed parameters of the connection. Example:

```
"accessId":"*****",
"accessKey":"*****",
"endpoint":"http://service.eu-central-1.maxcompute.aliyun-inc.com/api",
"odpsServer":"http://service.eu-central-1.maxcompute.aliyun-inc.com/api",
"tunnelServer":"http://dt.eu-central-1.maxcompute.aliyun.com",
"project":"*****",
```

Additional instructions

- Column filter

By configuring MaxCompute Writer, you can perform operations that MaxCompute does not support, such as filtering columns, reordering columns, and setting empty fields to null. To write data to all the columns in the destination table, set the column parameter as follows: "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 as follows:

"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 guarantee high data reliability, MaxCompute Writer returns an error message if more columns are to be written than expected. For example, a MaxCompute table has three columns: a, b, and c. MaxCompute Writer returns an error message if it is configured to write data to more than three columns.

- Partition configuration

MaxCompute Writer can only write data to the last-level partition, and does not support writing 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 last-level partition information, such as `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 guarantee 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 importing the source data again. This guarantees 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. By setting the `truncate` parameter to `true`, you can guarantee the idempotence of write operations and the data integrity.

**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`.

8.3.30 Configure Kafka Writer

Kafka Writer allows you to write data to Kafka by using the Java SDK for Kafka. This topic describes how Kafka Writer works, its parameters, and how to configure it by using the code editor.

Apache Kafka is a fast, scalable, and distributed message system with high throughput and fault-tolerance. It is used to publish and subscribe to messages. With the high-throughput, built-in partition, data replica, and fault-tolerance features, Kafka is suitable for processing a large amount of messages. Kafka Writer is currently in public preview.

How it works

Kafka Writer writes data to Kafka by using the following version of the Java SDK for Kafka:

```
<dependency>
  <groupId>org.apache.kafka</groupId>
  <artifactId>kafka-clients</artifactId>
  <version>2.0.0</version>
```

```
</dependency>
```

Parameters

Parameter	Description	Required
server	The broker server address of Kafka in the format of ip:port .	Yes
topic	The name of the topic to which data is written in Kafka. Kafka maintains feeds of messages in categories called topics. Each message published to the Kafka cluster is assigned to a topic. Each topic contains a group of messages.	Yes
KeyIndex	The column that is specified as the key.	Yes
valueIndex	The column that is specified as the value. If this parameter is not specified, all columns are concatenated by using the delimiter specified by fieldDelimiter to form the value.	No
fieldDelimiter	The delimiter used to separate columns when data is written to Kafka. Default value: \t.	No
keyType	The type of the key in Kafka. Valid values: BYTEARRAY, DOUBLE, FLOAT, INTEGER, LONG, and SHORT.	Yes
valueType	The type of the value in Kafka. Valid values: BYTEARRAY, DOUBLE, FLOAT, INTEGER, LONG, and SHORT.	Yes
batchSize	The number of data records that are written at a time. Default value: 1024.	No

Configure Kafka Writer by using the codeless UI

Currently, the codeless user interface (UI) is not supported for Kafka Writer.

Configure Kafka Writer by using the code editor

In the following code, a node is configured to write data to a Kafka topic.

```
{
  "type": "job",
  "version": "2.0", // The version number.
  "steps": [
    {
      "stepType": "stream",
      "parameter": {},
    }
  ]
}
```

```

    "name": "Reader",
    "category": "reader"
  },
  {
    "stepType": "Kafka", // The writer type.
    "parameter": {
      "server": "ip:9092", // The broker server address of Kafka
      "keyIndex": 0, // The column used as the key.
      "valueIndex": 1, // The column used as the value.
      "keyType": "Integer", // The type of the key in Kafka.
      "valueType": "Short", // The type of the value in Kafka.
      "topic": "t08", // The name of the destination topic.
      "batchSize": 1024 // 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 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"
    }
  ]
}
}

```

8.3.31 Configure Maxgraph Writer

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



Note:

If you want Maxgraph Writer to write data from a MaxCompute table to Maxgraph, grant the Maxgraph build account the read permission on the source table in the MaxCompute project. Contact the Maxgraph administrator to obtain the Maxgraph build account with the read permission on MaxCompute tables.

To write data from MaxCompute tables to Maxgraph, create data records and upload the data records by following these steps:

1. Create a MapReduce job to map columns in a MaxCompute table to the vertices or edges in Maxgraph. The MapReduce job converts the data records to the format supported in Maxgraph.
2. Upload the data records converted by the MapReduce job to the storage of Maxgraph.

Parameters

Parameter	Description	Required	Default value
endpoint	The endpoint of Maxgraph.	Yes	None
graphName	The name of the Maxgraph instance.	Yes	None
accessId	The AccessKey ID for accessing Maxgraph.	Yes	None
accessKey	The AccessKey secret for accessing Maxgraph.	Yes	None
label	The label, that is, the name of the vertex or edge.	Yes	None
labelType	The type of the label . Valid values: vertex and edge.	Yes	None
srcLabel	The label of the start vertex in an edge. This parameter only takes effect when you import data about edges.	Yes	None
dstLabel	The label of the end vertex in an edge. This parameter only takes effect when you import data about edges.	Yes	None
splitSize	The size of a shard in the MapReduce job.	No	256MB

Parameter	Description	Required	Default value
onlineMode	<p>The mode in which data is uploaded to the storage of Maxgraph. Valid values: partition and type.</p> <ul style="list-style-type: none">• partition: Before an upload operation is completed, both the existing data records and the newly uploaded data records can be queried. Data consistency is guaranteed after the synchronization ends. This mode has a fast data uploading speed.• type: Before an upload operation is completed, only the existing data records can be queried, if any. After the operation is completed, the new data records can be queried. This mode has a slow data uploading speed.	No	type
platform	The name of the destination data store. Valid value: odps.	No	odps

Parameter	Description	Required	Default value
column	The name of the vertex property. This parameter only takes effect when you import data about vertices.	Yes	None
name	The name of the property.	Required only for importing data about edges	None
propertyType	The type of the property. Valid values: srcPrimary Key, dstPrimaryKey, and edgeProperty.	Required only for importing data about edges	None
srcPrimaryKey	The primary key of the start vertex. This parameter takes only effect when you import data about edges.	Required only for importing data about edges	None
dstPrimaryKey	The primary key of the end vertex. This parameter only takes effect when you import data about edges.	Required only for importing data about edges	None
edgeProperty	The properties of the edge. This parameter can be left empty if the edge has no properties.	No	None

Configure Maxgraph Writer by using the code editor

The following examples provide the code for writing data about vertices and edges to Maxgraph, respectively.

- Configure a sync node to write data about vertices to Maxgraph

```
{
  "job": {
    "setting": {
```

```

"speed": {
  "channel": 1 // Set the value to 1.
},
"errorLimit": {
  "record": 1000
},
"content": [
  {
    "reader": {
      "name": "odpsreader",
      "parameter": {
        "accessId": "*****",
        "accessKey": "*****",
        "project": "maxgraph_dev",
        "table": "maxgraph_demo_person",
        "column": [ // The names of columns in the table in MaxCompute. The value
of this parameter has a one-to-one mapping with that of the column parameter of
Maxgraph Writer.
          "id",
          "name",
          "age"
        ],
        "packageAuthorizedProject": "biggraph_dev",
        "splitMode": "record",
        "odpsServer": "*****"
      }
    },
    "writer": {
      "name": "maxgraphwriter",
      "parameter": {
        "endpoint": "http://graph.alibaba.net",
        "graphName": "xxx",
        "accessId": "xxx",
        "accessKey": "xxx",
        "label": "person",
        "labelType": "vertex",
        "onlineMode": "partition",
        "platform": "odps",
        "splitSize": "256",
        "column": [ // The names of vertex properties in Maxgraph. The value of
this parameter has a one-to-one mapping with that of the column parameter of
MaxCompute Reader.
          "id",
          "name",
          "age"
        ]
      }
    }
  }
]
}

```

- Configure a sync node to write data about edges to Maxgraph

```

{
  "job": {
    "setting": {
      "speed": {
        "channel": 1 // Set the value to 1.
      },

```

```

"errorLimit": {
  "record": 1000
},
"content": [
  {
    "reader": {
      "name": "odpsreader",
      "parameter": {
        "accessId": "*****",
        "accessKey": "*****",
        "project": "maxgraph_dev",
        "table": "maxgraph_demo_knows",
        "column": [
          "person_id",
          "person_id2",
          "weight",
          "id"
        ],
        "packageAuthorizedProject": "biggraph_dev",
        "splitMode": "record",
        "odpsServer": "*****"
      }
    },
    "writer": {
      "name": "maxgraphwriter",
      "parameter": {
        "endpoint": "http://graph.alibaba.net",
        "graphName": "xxx",
        "accessId": "xxx",
        "accessKey": "xxx",
        "label": "knows",
        "labelType": "edge",
        "srcLabel": "person",
        "dstLabel": "person",
        "onlineMode": "partition",
        "platform": "odps",
        "splitSize": "256",
        "column": [
          {
            "name": "id", // The name of the edge property in Maxgraph.
            "propertyType": "srcPrimaryKey" // The type of the edge property in Maxgraph
          },
          {
            "name": "id",
            "propertyType": "dstPrimaryKey"
          },
          {
            "name": "weight",
            "propertyType": "edgeProperty"
          },
          {
            "name": "id",
            "propertyType": "edgeProperty"
          }
        ]
      }
    }
  }
]

```

. This parameter specifies whether a data record is the start vertex, end vertex, or a property of an edge.

```
}
```

8.3.32 Configure Vertica Writer

Vertica is a column-oriented database using the Massively Parallel Processing (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.

Specifically, Vertica Writer connects to a remote Vertica database through Java Database Connectivity (JDBC), and runs 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 extract-transform-load (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 database administrators (DBAs).

How it works

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 run 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 through 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 identical to the name of the added connection. You can add connections in the code editor.	Yes	None
jdbcUrl	<p>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.</p> <ul style="list-style-type: none"> 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: <code>jdbc:vertica://127.0.0.1:3306/database</code>. 	Yes	None
username	The username for connecting to the database.	Yes	None
password	The password for connecting to the database.	Yes	None
table	<p>The names of the destination tables, which are described in a JSON array.</p> <div>  Note: You do not need to set this parameter because the system automatically obtains the value from the connection parameter. </div>	Yes	None
column	The columns in the destination table to which data is written. Separate the columns with a comma (,). Example: <code>"column": ["id", "name", "age"]</code> .	Yes	None
preSql	The SQL statement to run before the sync node is run. Use <code>@table</code> to specify the name of the table to be modified in the SQL statement. When running this SQL statement, DataWorks replaces <code>@table</code> with the name of the target table.	No	None
postSql	The SQL statement to run after the sync node is run.	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 Vertica 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 Vertica Writer by using the codeless UI

Currently, the codeless user interface (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 run before the sync node is run.
          "delete from @table where db_id = -1"
        ],
        "postSql": [ // The SQL statement to run 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 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"
        }
      ]
    }
  }
}

```

8.3.33 Configure Gbase8a Writer

This topic describes the data types and parameters supported by Gbase8a Writer and how to configure it by using the code editor.

Gbase8a is a new type of column-oriented analytical database. Gbase8a Writer allows you to write data to Gbase8a databases.



Note:

Currently, Gbase8a Writer does not support the default resource group. Use exclusive resource groups for data integration or custom resource groups. For more information, see [Use exclusive resource groups for data integration](#) and [Add a custom resource group](#).

Specifically, Gbase8a Writer connects to a remote Gbase8a database through Java Database Connectivity (JDBC), and runs an `INSERT INTO` statement to write data to a Gbase8a database. Internally, data is written to the database in batches.

Gbase8a Writer is designed for extract-transform-load (ETL) developers to import data from data warehouses to Gbase8a databases. Gbase8a Writer can also be used as a data migration tool by users such as database administrators (DBAs).

Gbase8a Writer obtains data from a Data Integration reader, and generates the `INSERT INTO` statement based on your configurations.

Limits

- **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 Gbase8a database.



Note:



A sync node that uses Gbase8a Writer must have at least the permission to run 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.

- Gbase8a Writer does not support the **writeMode** parameter.
- Gbase8a Writer accesses a Gbase8a database through the MySQL database driver. Confirm the compatibility between the driver version and your Gbase8a database. Gbase8a Writer uses the following version of the MySQL database driver:

```
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>5.1.22</version>
</dependency>
```

Parameters

Parameter	Description	Required	Default value
jdbcUrl	<p>The JDBC URL for connecting to the Gbase8a database. You do not need to set this parameter because the system automatically obtains the value from the connection parameter.</p> <ul style="list-style-type: none"> • You can configure only one JDBC URL for a database . Gbase8a Writer cannot write data to a database with multiple primary databases. • The format must be in accordance with Gbase8a official specifications. You can also specify the information of the attachment facility. Example: <code>jdbc:mysql://127.0.0.1:3306/database</code>. 	Yes	None
username	The username for connecting to the database.	Yes	None
password	The password for connecting to the database.	Yes	None

Parameter	Description	Required	Default value
table	<p>The names of the destination tables, which are described in a JSON array.</p> <div>  Note: You do not need to set this parameter because the system automatically obtains the value from the connection parameter. </div>	Yes	None
column	<p>The columns in the destination table to which data is written. Separate the columns with a comma (,). Example: "column": ["id", "name", "age"].</p> <div>  Note: The column parameter cannot be left empty. </div>	Yes	None
preSql	The SQL statement to run before the sync node is run. Use @table to specify the name of the table to be modified in the SQL statement. When running this SQL statement, DataWorks replaces @table with the name of the target table.	No	None
postSql	The SQL statement to run 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 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 user interface (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 a Gbase8a database.

```
{
  "type": "job",
  "version": "2.0", // The version number.
  "steps": [
    {
      "stepType": "stream",
      "parameter": {},
      "name": "Reader",
```

```

    "category": "reader"
  },
  {
    "stepType": "gbase8a", // 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.
            "Gbase8a_table"
          ],
          "jdbcUrl": "jdbc:mysql://ip:port/database"
        }
      ],
      "preSql": [ // The SQL statement to run before the sync node is run.
        "delete from @table where db_id = -1"
      ],
      "postSql": [// The SQL statement to run 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 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"
    }
  ]
}
}

```

```
}
```

8.3.34 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, its parameters, and how to configure it by using the code editor.

**Notice:**

Currently, ClickHouse Writer does not support the default resource group. Use exclusive resource groups for data integration or custom resource groups. For more information, see [Use exclusive resource groups for data integration](#) and [Add a custom resource group](#).

ClickHouse Writer allows you to write data to ClickHouse databases. Specifically, ClickHouse Writer connects to a remote ClickHouse database through Java Database Connectivity (JDBC), and runs 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 (DBAs).

ClickHouse Writer obtains data from a Data Integration reader, and writes the data to the destination ClickHouse database by running the `INSERT INTO` statement that is generated based on your configurations.

Limits



- ClickHouse Writer connects to a ClickHouse database through JDBC, and can only write data to a destination table by using JDBC Statement.
- ClickHouse Writer supports filtering columns and changing the order of columns. You can enter columns as needed.
- Considering the load on ClickHouse, we recommend that you throttle the throughput per second (TPS) to 1,000 when the `INSERT INTO` write mode is used.
- After all tasks of a node are run, ClickHouse Writer performs a single-process POST Flush operation to update the data records in the ClickHouse database.
- Confirm the compatibility between the driver version and your ClickHouse database.

ClickHouse Writer uses 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>

Parameters

Parameter	Description	Required	Default value
jdbcUrl	<p>The JDBC URL for connecting to the ClickHouse database. You do not need to set this parameter because the system automatically obtains the value from the connection parameter.</p> <ul style="list-style-type: none"> You can configure only one JDBC URL for a database. The format must be in accordance with ClickHouse official specifications. You can also specify the information of the attachment facility. Example: jdbc:clickhouse://127.0.0.1:3306/database. 	Yes	None
username	The username for connecting to the database.	Yes	None
password	The password for connecting to the database.	Yes	None
table	<p>The names of the destination tables, which are described in a JSON array.</p> <div>  Note: You do not need to set this parameter because the system automatically obtains the value from the connection parameter. </div>	Yes	None
column	<p>The columns in the destination table to which data is written. Separate the columns with a comma (,). Example: "column": ["id", "name", "age"].</p> <div>  Note: The column parameter cannot be left empty. </div>	Yes	None
preSql	The SQL statement to run before the sync node is run. Use @table to specify the name of the table to be modified in the SQL statement. When running this SQL statement, DataWorks replaces @table with the name of the target table.	No	None
postSql	The SQL statement to run after the sync node is run.	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 ClickHouse 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 ClickHouse Writer by using the codeless UI

Currently, the codeless user interface (UI) is not supported for ClickHouse Writer.

Configure ClickHouse Writer by using the code editor

In the following code, a 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 columns to which data is written.
          "id",
          "name"
        ],
        "connection": [
          {
            "table": [// The name of the destination table.
              "ClickHouse_table"
            ],
            "jdbcUrl": "jdbc:clickhouse://ip:port/database"
          }
        ],
        "preSql": [ // The SQL statement to run before the sync node is run.
          "delete from @table where db_id = -1"
        ],
        "postSql": [// The SQL statement to run after the sync 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": 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"
      }
    ]
  }
}
```

8.3.35 Configure ApsaraDB for OceanBase Writer

ApsaraDB for OceanBase is a financial-grade distributed relational database developed by Alibaba Cloud and Ant Financial. This topic describes how ApsaraDB for OceanBase Writer works, its parameters, and how to configure it by using the code editor.

**Notice:**

Currently, ApsaraDB for OceanBase Writer does not support the default resource group. Use exclusive resource groups for data integration or custom resource groups. For more information, see [Use exclusive resource groups for data integration](#) and [Add a custom resource group](#).

Background

ApsaraDB for OceanBase implements automated and non-disruptive disaster recovery across cities with the Five Data Centers in Three Regions solution. It provides high availability for financial services through conventional hardware. With the online scaling capability, ApsaraDB for OceanBase is a Chinese database that has undergone strict verification in terms of functionality, stability, scalability, and performance.

ApsaraDB for OceanBase Writer is designed for extract-transform-load (ETL) developers to import data from data warehouses to ApsaraDB for OceanBase databases. ApsaraDB for OceanBase Writer can also be used as a data migration tool by users such as database administrators (DBAs).

ApsaraDB for OceanBase Writer obtains data from a Data Integration reader, and generates the INSERT INTO statement based on your configurations.

Limits

- **INSERT INTO**: If a primary key conflict or unique index conflict occurs, data cannot be written to the conflicting rows. ApsaraDB for OceanBase in Oracle mode supports only the write mode of INSERT INTO.
- **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. ApsaraDB for OceanBase Writer in MySQL mode supports both the write modes of INSERT INTO and ON DUPLICATE KEY UPDATE.
- Data can be written only to tables stored in the primary ApsaraDB for OceanBase database.



Note:



A sync node that uses ApsaraDB for OceanBase Writer must have at least the permission to run 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.


- We recommend that you write data to the destination table in **batch** mode. In this mode, ApsaraDB for OceanBase Writer submits a write request when the number of rows reaches a specific threshold.
- ApsaraDB for OceanBase supports Oracle and MySQL modes. When you configure the **preSql** and **postSql** parameters, make sure that the SQL statements specified in the parameters comply with the SQL syntax constraints on the corresponding mode. Otherwise, the SQL statements may fail to be run.
- ApsaraDB for OceanBase Writer accesses an ApsaraDB for OceanBase database through the OceanBase driver. Confirm the compatibility between the driver version and your ApsaraDB for OceanBase database. ApsaraDB for OceanBase Writer uses the following version of the OceanBase database driver:

```
<dependency>
  <groupId>com.alipay.OceanBase</groupId>
  <artifactId>OceanBase-connector-java</artifactId>
  <version>3.1.0</version>
```


</dependency>

Parameters

Parameter	Description	Required	Default value
datasource	<p>The connection name. It must be identical to the name of the ApsaraDB for OceanBase connection that you added in Dataworks.</p> <p>You can connect to the ApsaraDB for OceanBase database based on the settings of the jdbcUrl and username parameters.</p>	No	None
jdbcUrl	<p>The Java Database Connectivity (JDBC) URL for connecting to the ApsaraDB for OceanBase database. You do not need to set this parameter because the system automatically obtains the value from the connection parameter.</p> <ul style="list-style-type: none"> You can configure only one JDBC URL for a database . ApsaraDB for OceanBase Writer cannot write data to a database with multiple primary databases. The format must be in accordance with ApsaraDB for OceanBase official specifications. You can also specify the information of the attachment facility. Example: jdbc:mysql://127.0.0.1:3306/database. 	Yes	None
username	The username for connecting to the database.	Yes	None
password	The password for connecting to the database.	Yes	None
table	<p>The names of the destination tables, which are described in a JSON array.</p> <div>  Note: You do not need to set this parameter because the system automatically obtains the value from the connection parameter. </div>	Yes	None
column	<p>The columns in the destination table to which data is written. Separate the columns with a comma (,). Example: "column": ["id", "name", "age"].</p> <div>  Note: The column parameter cannot be left empty. </div>	Yes	None

Parameter	Description	Required	Default value
writeMode	The write mode. Valid values: <code>insert into</code> and <code>on duplicate key update</code> .	Yes	None
preSql	The SQL statement to run before the sync node is run. Use <code>@table</code> to specify the name of the table to be modified in the SQL statement. When running this SQL statement, DataWorks replaces <code>@table</code> with the name of the target table.	No	None
postSql	The SQL statement to run after the sync node is run.	No	None
batchSize	<p>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 OceanBase database over the network, and increase the throughput.</p> <div>  Note: A value larger than 2048 may lead to the out of memory (OOM) error during the data synchronization process. </div>	No	1024

Configure ApsaraDB for OceanBase Writer by using the codeless UI

Currently, the codeless user interface (UI) is not supported for ApsaraDB for OceanBase Writer.

Configure ApsaraDB for OceanBase Writer by using the code editor

In the following code, a node is configured to write data to an ApsaraDB for OceanBase database.

```
{
  "type": "job",
  "version": "2.0", // The version number.
  "steps": [
    {
      "stepType": "stream",
      "parameter": {},
      "name": "Reader",
      "category": "reader"
    },
    {
      "stepType": "apsaradb_for_OceanBase", // The writer type.
      "parameter": {
        "datasource": "The connection name.",
        "column": [ // The columns to which data is written.
          "id",
          "name"
        ]
      }
    }
  ]
}
```

```

    },
    "table": "apsaradb_for_OceanBase_table", // The name of the destination table.
    "preSql": [ // The SQL statement to run before the sync node is run.
        "delete from @table where db_id = -1"
    ],
    "postSql": [ // The SQL statement to run after the sync node is run.
        "update @table set db_modify_time = now() where db_id = 1"
    ],
    "writeMode": "insert",
  },
  "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"
    }
  ]
}
}
}

```

8.3.36 Hologres Writer

Hologres Writer allows you to write data to Hologres. You can import data from multiple data stores to Hologres and use Hologres to analyze data in real time.



Notice:

Currently, you can only use exclusive resource groups for Data Integration for Hologres Writer. The default resource group and custom resource groups are not supported. For more information, see [Use exclusive resource groups for data integration](#) and [Add a custom resource group](#).

How it works

Hologres Writer obtains data from a Data Integration reader, and writes data to the destination database based on the values of the **writeMode** and **conflictMode** parameters.

- If the **writeMode** parameter is set to **SDK (Fast Write)**, Hologres Writer writes data to Hologres through the HoloHub API. This mode provides the optimal performance for you.
- If the **writeMode** parameter is set to **SQL (INSERT INTO)**, Hologres Writer writes data to Hologres through the `INSERT INTO` statement provided by PostgreSQL.

You can use the **conflictMode** parameter to specify how to process the conflicting data when a primary key conflict occurs.

- If the **conflictMode** parameter is set to **Replace**, the new data overwrites the existing data.
- If the **conflictMode** parameter is set to **Ignore**, the existing data is retained and the new data is ignored.

In different write modes, different methods are used to specify how to process the conflicting data. If the **writeMode** parameter is set to **SDK (Fast Write)**, you need to configure the properties of the Hologres table to specify how to process the conflicting data.


**Notice:**

The **conflictMode** parameter is available only when the table has a primary key.

Parameters

Parameter	Description	Required	Default value
endpoint	<p>The endpoint used to connect to the destination Hologres instance, in the format of instance-id-region-endpoint.hologres.aliyuncs.com:port. You can view the endpoints of a Hologres instance on the configuration page of the instance in the Hologres console.</p> <p>The endpoint of a Hologres instance varies with the network types, including the classic network, Internet, and Virtual Private Cloud (VPC). Select an appropriate endpoint based on the network where the resource group for Data Integration and the Hologres instance reside. Otherwise, the connection may fail or the performance may be poor. The formats of these three types of endpoints are as follows:</p> <ul style="list-style-type: none">Public endpoint: instance-id-region-endpoint.hologres.aliyuncs.com:portClassic network endpoint: instance-id-region-endpoint-internal.hologres.aliyuncs.com:portVPC endpoint: instance-id-region-endpoint-vpc.hologres.aliyuncs.com:port <p>We recommend that you deploy the resource group for Data Integration and the Hologres instance in the same zone of the same region to guarantee network connection and optimal performance.</p>	Yes	None
accessId	The AccessKey ID of the account used to access Hologres.	Yes	None

Parameter	Description	Required	Default value
accessKey	The AccessKey secret of the account used to access Hologres. Specify an AccessKey secret of an account that is authorized to write data to the destination table.	Yes	None
database	The name of the destination database in the Hologres instance.	Yes	None
table	The name of the destination Hologres table. You can specify the table name in the format of Schema name.Table name.	Yes	None
writeMode	<p>The write mode. Valid values: SDK (Fast Write) and SQL (INSERT INTO). For more information, see How it works.</p> <p>In the code editor, you can set the following parameters if you use the SDK (Fast Write) mode.</p> <ul style="list-style-type: none"> • maxCommitSize: specifies the maximum size of data that can be written to Hologres at a time. Default value: 1048576. This parameter is optional. • maxRetryCount: specifies the maximum number of retries for writing data to Hologres in the SDK (Fast Write) mode. Default value: 500. This parameter is optional. • retryInterval: specifies the retry interval at which data is written in the SDK (Fast Write) mode. Unit: milliseconds. Default value: 1000. This parameter is optional. 	Yes	None
conflictMode	The mode in which the conflicting data is processed. Valid values: Replace and Ignore . For more information, see How it works .	Yes	None

Parameter	Description	Required	Default value
column	The columns in the destination Hologres table to which data is written. The primary key columns of the destination table must be included. 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
partition	<p>The partition key column and the corresponding value of the destination Hologres table, in the format of column=value. This parameter is valid for partitioned tables.</p> <div>  Note: <ul style="list-style-type: none"> Currently, Hologres only supports list partitioning and you can only specify a single column as the partition key column. The data type of the partition key column must be INT4 or TEXT. The parameter value must match the partition expression in the data definition language (DDL) statements used to create the destination table. </div>	No	Null, indicating a non-partitioned table.

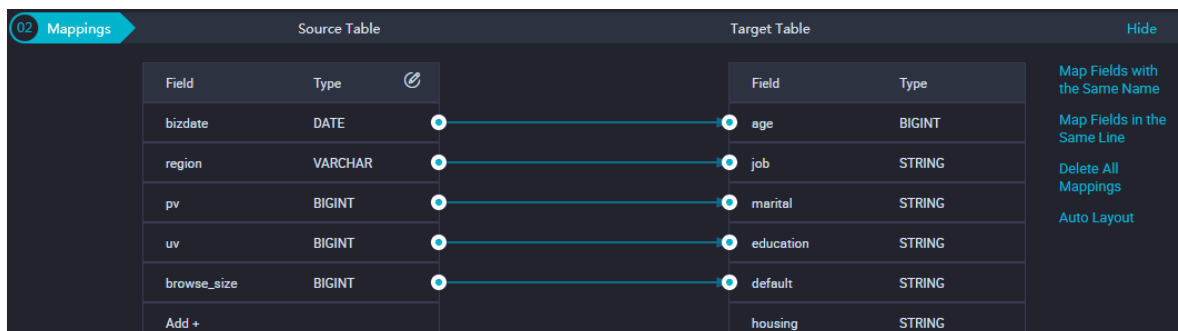
Configure Hologres Writer by using the codeless UI

1. Configure the connections.

Configure the source and destination connections for the sync node.

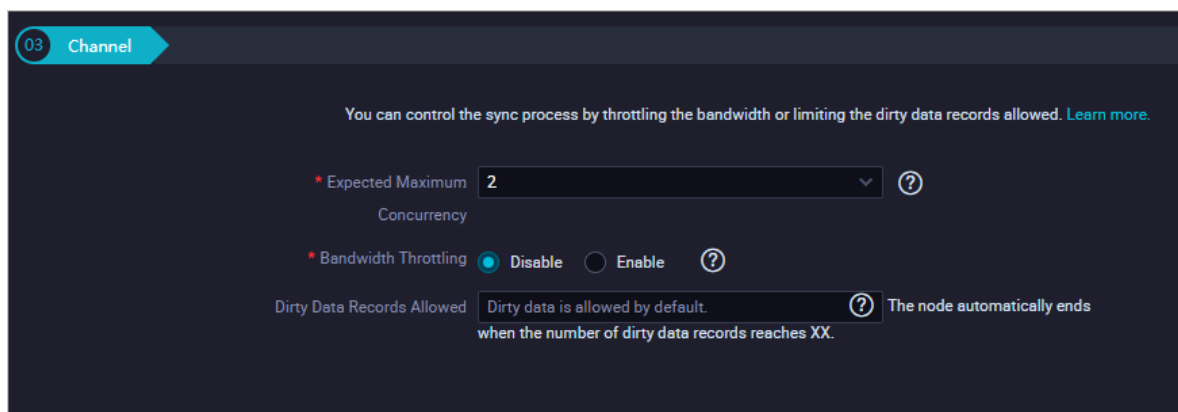
Parameter	Description
Connection	The name of the connection.
Table	The table parameter in the preceding parameter description.
Write Mode	The writeMode parameter in the preceding parameter description.
Solution to Data Write Conflicts	The conflictMode parameter in the preceding parameter description.

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.



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. The fields are automatically sorted based on specified rules.

3. Configure channel control policies.



Parameter	Description
Expected Maximum 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).

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 Hologres Writer by using the code editor

For more information about how to use the code editor, see [Create a sync node by using the code editor](#).

- Use a non-partitioned table as the destination table
 - In the following code, a node is configured to synchronize data from the memory to a non-partitioned Hologres table in the **SDK (Fast Write)** mode.

```
{
  "type": "job",
  "version": "2.0", // The version number.
  "steps": [
    {
      "stepType": "stream",
      "parameter": {},
      "name": "Reader",
      "category": "reader"
    },
    {
      "stepType": "holo",
      "parameter": {
        "endpoint": "instance-id-region-endpoint.hologres.aliyuncs.com:port",
        "accessId": "<yourAccessKeyId>", // The AccessKey ID of the account used
        "accessKey": "<yourAccessKeySecret>", // The AccessKey secret of the
        "database": "postgres",
        "table": "<yourTableName>",
        "writeMode": "sdk",
        "conflictMode": "replace",
        "column": [
          "tag",
          "id",
          "title"
        ],
        "maxCommitSize": 1048576,
        "maxRetryCount": 500
      },
      "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"
    }
  ]
}
}

```

- The following section provides the sample DDL statements used to create a non-partitioned Hologres table.

```

begin;
drop table if exists test_holowriter_sdk_replace;
create table test_holowriter_sdk_replace(
tag text not null,
id int not null,
body text not null
primary key (tag, id));
call set_table_property('test_holowriter_sdk_replace', 'orientation', 'column');
call set_table_property('test_holowriter_sdk_replace', 'shard_count', '3');
commit;

```

- Use a partitioned table as the destination table
 - In the following code, a node is configured to synchronize data from the memory to a child partitioned table in Hologres in the **SDK (Fast Write)** mode.


Note:

Exercise caution when you set the **partition** parameter.

```

{
  "type": "job",
  "version": "2.0", // The version number.
  "steps": [
    {
      "stepType": "stream",
      "parameter": {},
      "name": "Reader",
      "category": "reader"
    },
    {
      "stepType": "holo",
      "parameter": {
        "endpoint": "instance-id-region-endpoint.hologres.aliyuncs.com:port",
        "accessId": "<yourAccessKeyId>", // The AccessKey ID of the account used
        to access Hologres.
      }
    }
  ]
}

```

```

        "accessKey": "<yourAccessKeySecret>", // The AccessKey secret of the
account used to access Hologres.
        "database": "postgres",
        "table": "<yourTableName>",
        "writeMode": "sdk",
        "conflictMode": "ignore",
        "column": [
            "*"
        ],
        "partition": "tag=foo",
        "maxCommitSize": 1048576,
        "maxRetryCount": 500
    },
    "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"
        }
    ]
}
}
}

```

- The following section provides the sample DDL statements used to create a partitioned Hologres table.

```

begin;
drop table if exists test_holowriter_part_table_sdk_ignore;
create table test_holowriter_part_table_sdk_ignore(
    tag text not null,
    id int not null,
    title text not null,
    body text,
    primary key (tag, id))
partition by list( tag );
call set_table_property('test_holowriter_part_table_sdk_ignore', 'orientation', '
column');
call set_table_property('test_holowriter_part_table_sdk_ignore', 'shard_count', '3
');
commit;

```