Alibaba Cloud DataWorks

Data Development

Issue: 20190920

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 content of the Alibaba Cloud website, including but not limited to works, products, images, archives, information, materials, website architecture, website graphic layout, and webpage design, 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 the Alibaba Cloud website, product programs, or content 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.

Generic conventions

Table -1: Style conventions

Style	Description	Example
	This warning information 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.
	This warning information 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 restore business.
	This indicates warning information, supplementary instructions, and other content that the user must understand.	 Notice: Take the necessary precautions to save exported data containing sensitive information.
	This indicates supplemental instructions, best practices, tips, and other content that is good to know for the user.	 Note: You can use Ctrl + A to select all files.
>	Multi-level menu cascade.	Settings > Network > Set network type
Bold	It is used for buttons, menus , page names, and other UI elements.	Click OK.
`Courier font`	It is used for commands.	Run the `cd /d C:/windows` command to enter the Windows system folder.
Italics	It is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	It indicates that it is a optional value, and only one item can be selected.	`ipconfig` *[-all|-t]*

Style	Description	Example
{} or {a|b}	It indicates that it is a required value, and only one item can be selected.	`swich {stand | slave}`

Contents

1 Data development

1.1 Solution

This topic describes how to operate the data development mode. The data development mode has been upgraded to the three-level structure comprising of project, solution, and business flow. This data development mode abandons the conventional directory organization mode.

Project-solution-business flow

In the latest version of DataWorks , the data development mode is upgraded to integrate different node task types based on business types. This structure improves the facilitation of code development by businesses, and allow the development to be implemented across multiple business flows from a wider perspective. The three-level structure of the project-solution-business flow redefines the development process and improves the users' development experience.

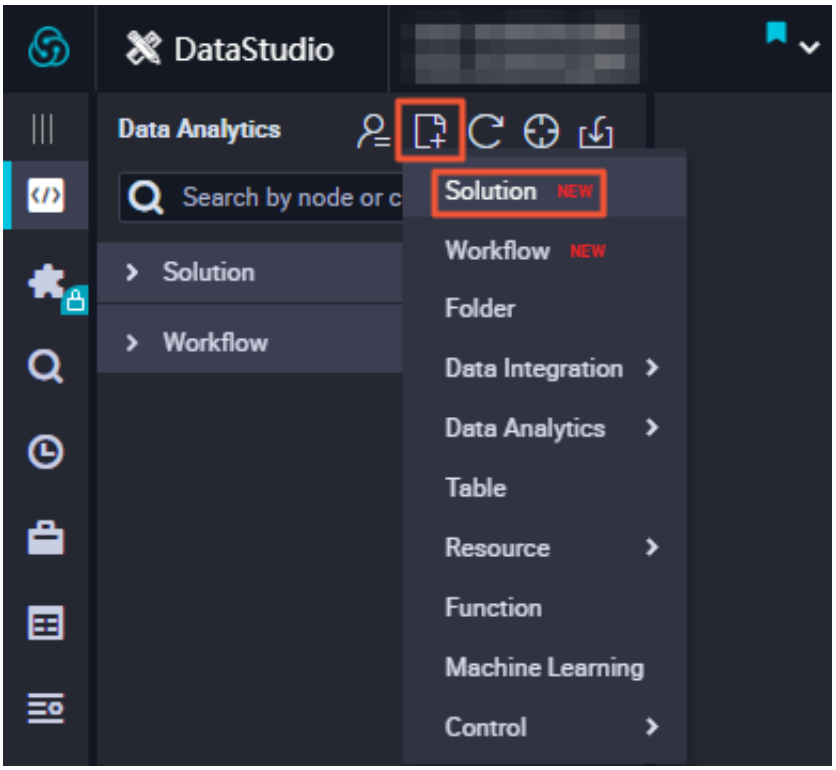
- Project: The basic unit for the permission organization that is used for controlling user permissions, such as development and O&M permissions. In the same project, all project member codes can be developed and managed in a collaborative manner.
- Solution: Users can customize a solution by combining some business flows. The following are the solution advantages:
  - A solution contains multiple business flows.
  - The same business flow can be reused in different solutions.
  - The immersion development can be implemented for a combined solution.

- Business flow: An abstract entity of the business, which allows users to organize data code development from the business perspective. A business flow can be reused by multiple solutions. The following are the business flow advantages:
  - The business flow helps users to organize codes from the business perspective . It provides the task type-based code organization mode. It supports multiple levels of sub-directories (preferentially up to four-levels).
  - The entire business flow can be viewed and optimized from the business perspective.
  - The business flow dashboard is provided to improve the development efficiency.
  - The release and O&M can be organized based on the business flow.

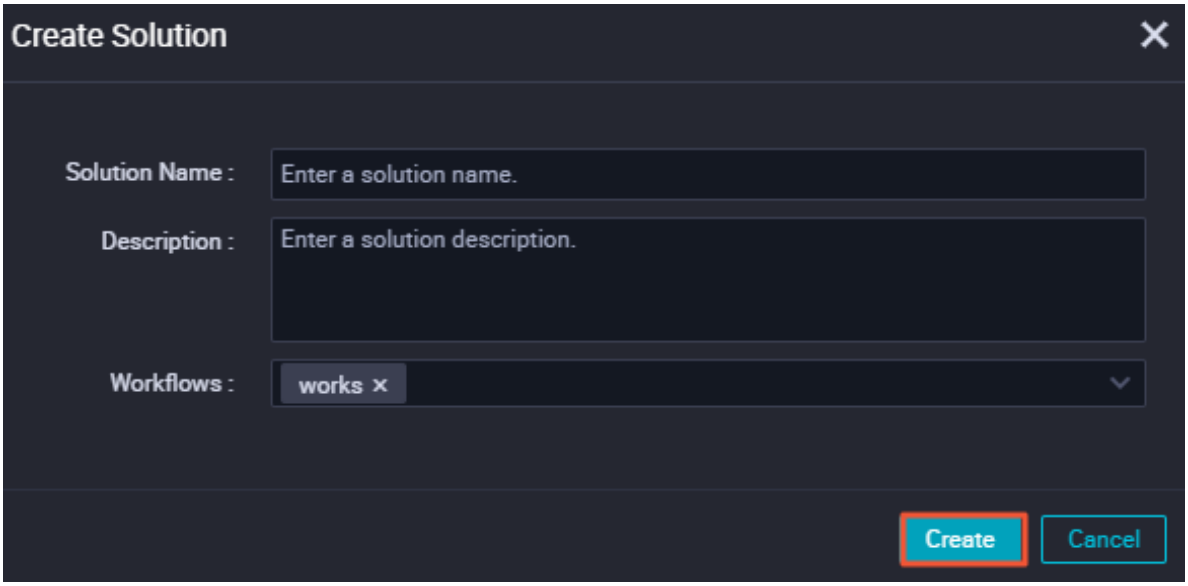
Immersion development experiences

You can double-click any created solution to switch from the development area to the solution area. The directory displays only the current solution content, which provides a clean environment, that is not affected by other project codes that are unrelated to the current solution.

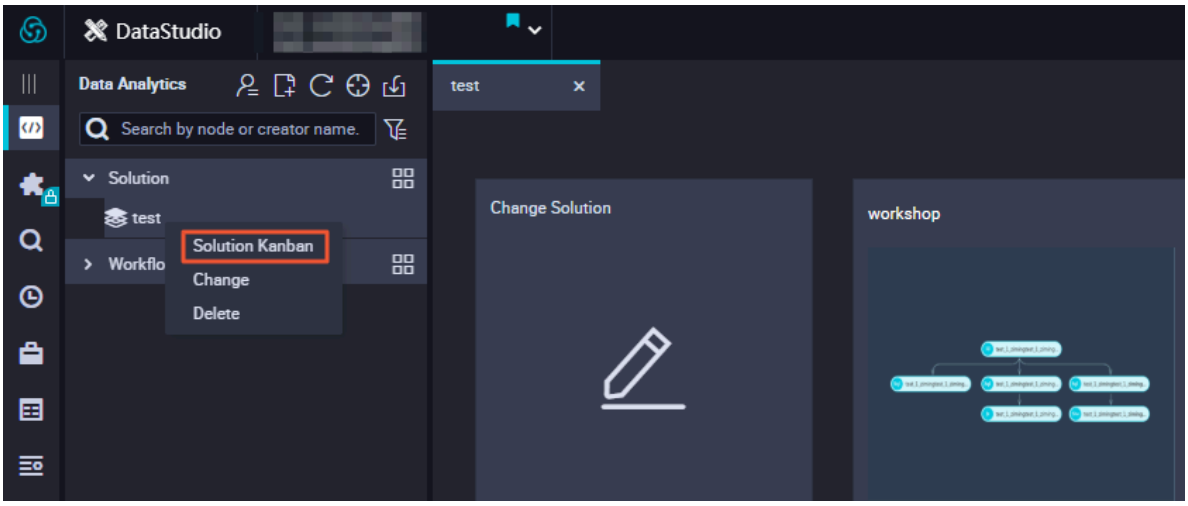
1. Go to the DataStudio page and create a solution.



2. Select the business flow for viewing from the created solution.



3. Right-click View All Business Flows to view nodes of the selected business flow or modify the solution.



4. Go to another page.

   - Click Publish to go to the Task Publish page. Nodes in the To Be Released status under the current solution are displayed on this page.
   - Click O&M to go to the O&M Center > Periodic Instances page. By default, periodic instances of all nodes under the current solution are displayed on this page.

A business flow can be reused by multiple solutions, which allows you to focus on solution development. Other users can edit your referenced business flows, or business flows in other solutions, and implement collaborative development.

1.2 SQL code encoding principles and standards

Short Description: This topic describes the basic SQL code encoding principles and standards.

Encoding principles

The SQL code is encoded as follows:

- **The code is comprehensive and healthy.**
- **The code lines are clear, neat, and orderly.**
- **The code lines are well arranged and have a good hierarchical structure.**
- **The comments must be provided to improve the code's readability.**
- **The principle requires no constraint conventions for developers coding behavior . In practice, the general requirement preconditions are not violated, rational deviations from this convention are acceptable. If they are beneficial to code development then this convention can be continuously improved and supplement ed.**
- **All keywords and reserved words used in SQL codes are in lowercase, such as the following: Select, From, Where, And, Or, Union, Insert, Delete, Group, Having, and Count.**
- **Keywords and reserved codes used in SQL codes, and other codes including field names and table alias must be in lowercase.**
- **Four spaces are equivalent to an indention unit. All indentions must be the integer multiples of an indention unit and aligned according to the code hierarchy.**
- **You are not allowed to use the select asterisk (*) operation. The column name must be specified in all operations.**
- **The corresponding brackets must be on the same column.**

SQL coding specification

The SQL code specification is as follows:

- **Code header**

  The code header must have the following information such as: subject, function description, author, and date. The log and title bars must be reserved so that other

users can edit records. Note that each line must not exceed 80 characters in length. The following is a template:


```
--  MaxCompute ( ODPS )  SQL
--
********************************************************************************
--  Subject :  Transactio  n
--  Function   descriptio  n :  Transactio  n   refund   analysis
 --  Author :  With   code
 --  Create   time :  20170616
 --  Change   log :
 --  Modified   on    Modified   by    Content
--  yyyymmdd   name   comment
--  20170831   Without   code   Add   a   judgment   on   the
 transactio  n   biz_type = 1234
--
********************************************************************************
```

- Field arrangement requirements:
  - Each selected field for the SELECT statement occupies one line.
  - One indention next to the word "select" is followed by the first selected field. That is, the field is two indentions away from the line start .
  - Each alternating field starts with two indentions, followed by a comma (,) and then the field name.
  - The comma (,) between two fields come before the second field.
  - The as statement must be in the same line as the related fields. We recommend that the "as" statements with multiple fields must be aligned in the same column.

```
select  channel_id            as channel_id
       ,trade_channel_desc    as trade_channel_desc
       ,trade_channel_edesc   as trade_channel_edesc
       ,inst_date             as inst_date
       ,trade_iswap           as trade_iswap
       ,channel_type          as channel_type
       ,channel_second_desc   as channel_second_desc
from   (
```

- INSERT sub-statement arrangement requirements

  The INSERT sub-statement must be written in the same row. You are not allowed to use the line feed.

- SELECT sub-statement arrangement requirements

  Sub-statements used by the SELECT statements, include From, Where, Group by, Having, Order by, Join, and Union, must conform to the following requirements:

  - The line feed.
  - The sub-statements must be left-aligned with the SELECT statement.
  - You must reserve two indentions between the first letter of a sub-statement and its subsequent code.
  - The logical operators, such as "AND" and "OR" in a "WHERE" sub-statement must be left-aligned with WHERE.
  - If the length of a sub-statement exceeds two indentions, add a space to the sub-statement, and write the subsequent code. For example: "order by" and "group by"

```
select      trim(channel) channel
            ,min(id)      id
from        ods_trd_trade_base_dd
where       channel is not null
and         dt = ${tmp_uuuummdd}
and         trim(channel) <> ''
group by    trim(channel)
order by    trim(channel)
```

- The spacing requirements before and after an operator as follows: A space must be reserved before and after an arithmetic operator or a logical operator, and operators must be written on the same line unless the line exceeds 80 characters in length.

```
select      trim(channel) channel
            ,min(id)      id
from        ods_trd_trade_base_dd
where       channel is not null
and         dt = ${tmp_uuuummdd}
and         trim(channel) <> ''
group by    trim(channel)
order by    trim(channel)
```

- Compiling the "CASE" statement

  In a "SELECT" statement, the "CASE" statement is used to judge or assign field values. The correct compiling of the "CASE" statement is critical for enhancing the code lines readability.

  The following conventions are stipulated for compiling the "CASE" statement:

  - The "WHEN" sub statement is in the same line as the "CASE" statement and starts after one indention.
  - Each "WHEN" sub statement occupies one line. The line feed is acceptable if the statement is too long.
  - A "CASE" statement must contain the "ELSE" sub statement. The "ELSE" sub statement must be aligned with the "WHEN" sub statement.

```
,case   when p1.trade_from = '3008' and p1.trade_email is null then 2
        when p1.trade_from = '4000' and p1.trade_email is null then 1
        when p9.trade_from_id is not null then p9.trade_from_id
 end                                as trade_from_id
,p1.trade_email                     as partner_id
```

- Nesting query compiling specification

  The nesting sub-query is often used in Extract, transform, load (ETL) development of the data warehouse system. Therefore, it is important to arrange codes in a hierarchical manner. For example:

```
select      p.channel
            ,rownumber()  order_id
from        (
                select    s1.channel
                          ,s1.id
                from      (
                                select  trim(channel)       as channel
                                        ,min(id)            as id
                                from    ods_trd_trade_base_dd
                                where   channel is not null
                                and     dt = ${tmp_yyyymmdd}
                                and     trim(channel) <> ''
                                group by trim(channel)
                          ) s1
                left outer join
                          dim_trade_channel s2
                on        s1.channel = s2.trade_channel_edesc
                where     s2.trade_channel_edesc is null
                order by id
            ) p
;
```

- Table alias definition convention

  - The alias must be added to all tables. Once an alias is defined for an operation table in a "SELECT" statement, the alias must be used whenever there are table

statement references. To facilitate the code compiling, the alias must be simple and concise whenever possible and keywords must be avoided.

- The table alias is defined with simple characters. We recommend that aliases are defined in alphabetical order.
- The hierarchy must be shown before using the multi-layered nesting sub-query of an alias. The SQL statement alias is defined by the layer. Layer 1 to 4 are represented by P (Part), S (Segment), U (Unit), and D (Detail), respectively. Alternatively, Layer 1 to 4 can be represented by a, b, c, and d. Sub-statements in the same layer are differentiated from each other by numbers, such as 1, 2, 3, and 4 behind the letter that represents the layer. A comment can be added for a table alias.

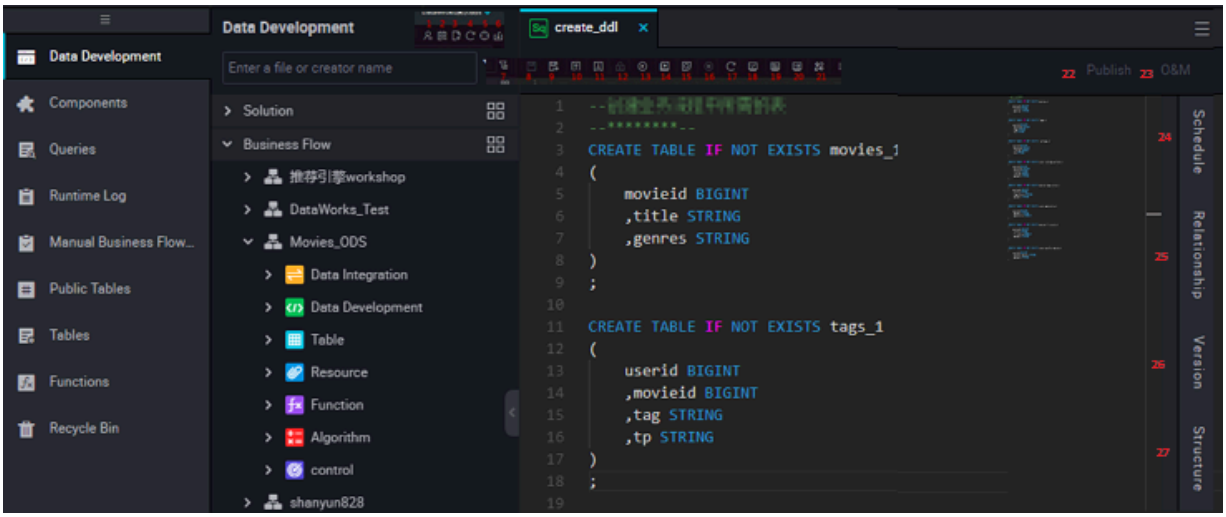
```
select     p.channel
           ,rownumber()  order_id
from       (
               select   s1.channel
                        ,s1.id
               from     (
                               select  trim(channel)      as channel
                                       ,min(id)           as id
                               from    ods_trd_trade_base_dd
                               where   channel is not null
                               and     dt = ${tmp_yyyymmdd}
                               and     trim(channel) <> ''
                               group by trim(channel)
                        ) s1
               left outer join
                       dim_trade_channel s2
               on      s1.channel = s2.trade_channel_edesc
               where   s2.trade_channel_edesc is null
               order by id
           ) p
;
```

- Comments within the SQL statement
  - The comment must be added for each SQL statement.
  - The comment for each SQL statement exclusively occupies a single line, and is placed in front of the statement.
  - The field comment must be added behind the field.
  - Comments must be added to branch condition expressions that are difficult to understand.
  - Comments must be added to describe important calculation functions.
  - If a function is too long, the statement must be segmented based on the implemented functions, and comments must be added to describe each segment .
  - Comments must be added to a constant or variable to explain the saved value, but comments are optional for a valid value range.

```
-- ********************************************************************************
-- STEP1:    Clean up data partition on tmp_dws_tbd_alijr_user_relation_dd_5
                                      that day.
-- ********************************************************************************
```

1.3 Console functions

1.3.1 Introduction to console



The interface function points are described below:

No.	Feature	Description
1	Show my files	View nodes under your account in the current column.

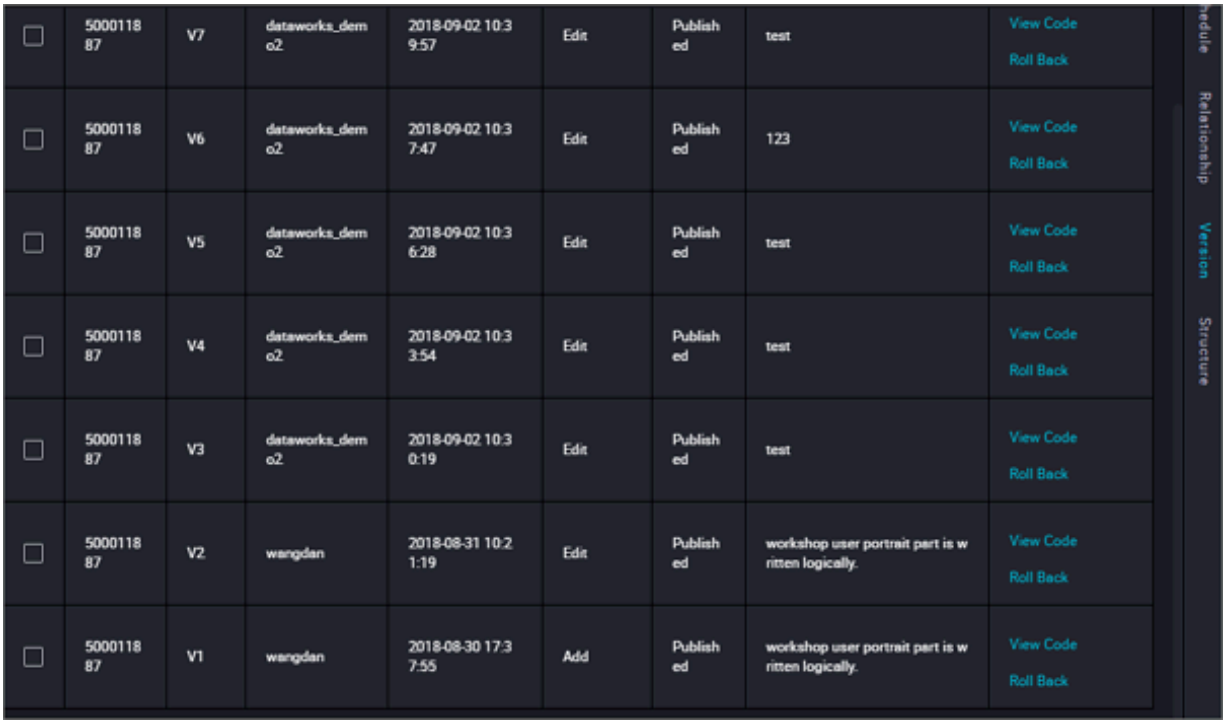
No.	Feature	Description
2	Code search	Search for a code or a code segment.
3	[+]	Creates a solution, business flow, folder, node, table, resource, or function entry.
4	Reload	Refreshes the current directory tree.
5	Locate	Locates the selected file position.
6	Import	Imports local data to an online table. Note: The encoding format.
7	Filter	Filter nodes based on the specified conditions.
8	Save	Saves the current code.
9	Save as query file	Saves the current code as a temporary file, which is displayed in the temporary query column.
10	Submit	Submits the current node.
11	Submit and unlock	Submits the current node and unlocks the node to edit the code.
12	Steal lock	Edits a node that you do not have ownership over.
13	Run	Runs the current node code.
14	Run after setting parameters	Runs the code of the current node with the configured parameters.
15	Precompile	Edit and test the current node parameters.
16	Stop run	Stops the run code.
17	Reload	Refreshes the page and returns to the previously saved page.
18	Run smoke test in development environment	Tests the current node code in the development environment.
19	View smoke test ; og in development environment	Views the run log of a node in the development environment.
20	Go to scheduling system of development environment	Goes to the O&M center of the development environment.

No.	Feature	Description
21	Format	The sequence codes of the current node. It is often used when the code on a single line is too long.
22	Publish	Publishes the submitted code. After the code is published, the code is under the production environment.
23	O&M	Goes to the O&M center of the production environment.
24	Scheduling Configuration	Configures the scheduling attributes, parameters , and Resource Groups of a node.
25	Relationship	View the relationship between tables used by the code.
26	Version	View the submission and publish records of the current node.
27	Structure	View the code structure of the current node. If the code is too long, you can quickly locate a code segment based on the key information in the structure.

1.3.2 Version

A version is a submission and release record of the current node, where each submission generates a new version. You can check the related status, change type, and release remarks as required to facilitate operations on the node.

Note:
Only a submitted node has the version information.

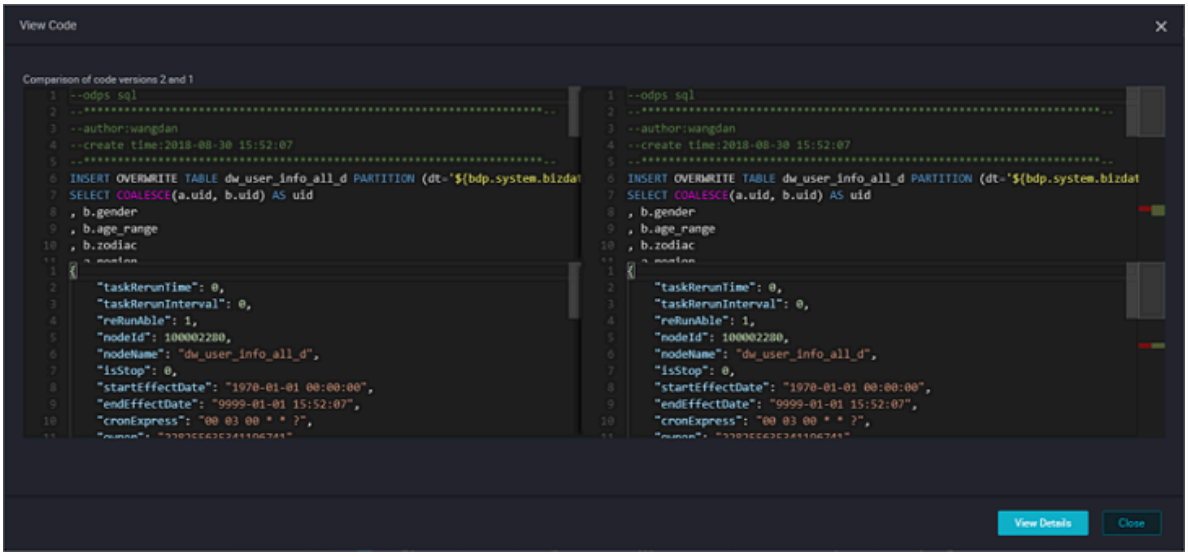


☐	500011887	V7	dataworks_demo2	2018-09-02 10:39:57	Edit	Published	test	View Code Roll Back
☐	500011887	V6	dataworks_demo2	2018-09-02 10:37:47	Edit	Published	123	View Code Roll Back
☐	500011887	V5	dataworks_demo2	2018-09-02 10:36:28	Edit	Published	test	View Code Roll Back
☐	500011887	V4	dataworks_demo2	2018-09-02 10:33:54	Edit	Published	test	View Code Roll Back
☐	500011887	V3	dataworks_demo2	2018-09-02 10:30:19	Edit	Published	test	View Code Roll Back
☐	500011887	V2	wangdan	2018-08-31 10:21:19	Edit	Published	workshop user portrait part is written logically.	View Code Roll Back
☐	500011887	V1	wangdan	2018-08-30 17:37:55	Add	Published	workshop user portrait part is written logically.	View Code Roll Back

Schedule Relationship Version Structure

- File ID: The current node ID.
- Version: A new version is generated for each release. The first release is V1, the second modification is V2, and so on.
- Submitter: The operator who submits and releases the node.
- Submission time: The version release time. If a version is submitted and then released, the release time covers the submission time. By default, the last release time of the operation is recorded.
- Change type: The operation history of the current node. It is set to Added if the node is first released, and set to Modified if the node is modified.
- Status: The operation status record of the current node.
- Remarks: Changes the description of the current node when submitted. It facilitates other personnel to locate the related version when operating the node.
- Action: You can select Code and Roll Back in this column.
  - View code: Click it to view the version code and precisely search for a record version to be roll back.
  - Roll back: Click it to roll back the current node to a previous version as required. You must submit the node for release again after roll back.

- Compare: Click it to compare the code and parameters of two versions.



Click View Details to go to the details page and compare the code and scheduling attribute changes.

> Note:
> Only two versions can be compared. You cannot compare only one or more than three nodes.

1.3.3 Structure

The structure is based on the current Code, which parses the process diagram that runs under SQL, helps users quickly review the edited SQL situation, so that it can be easily modified and viewed.

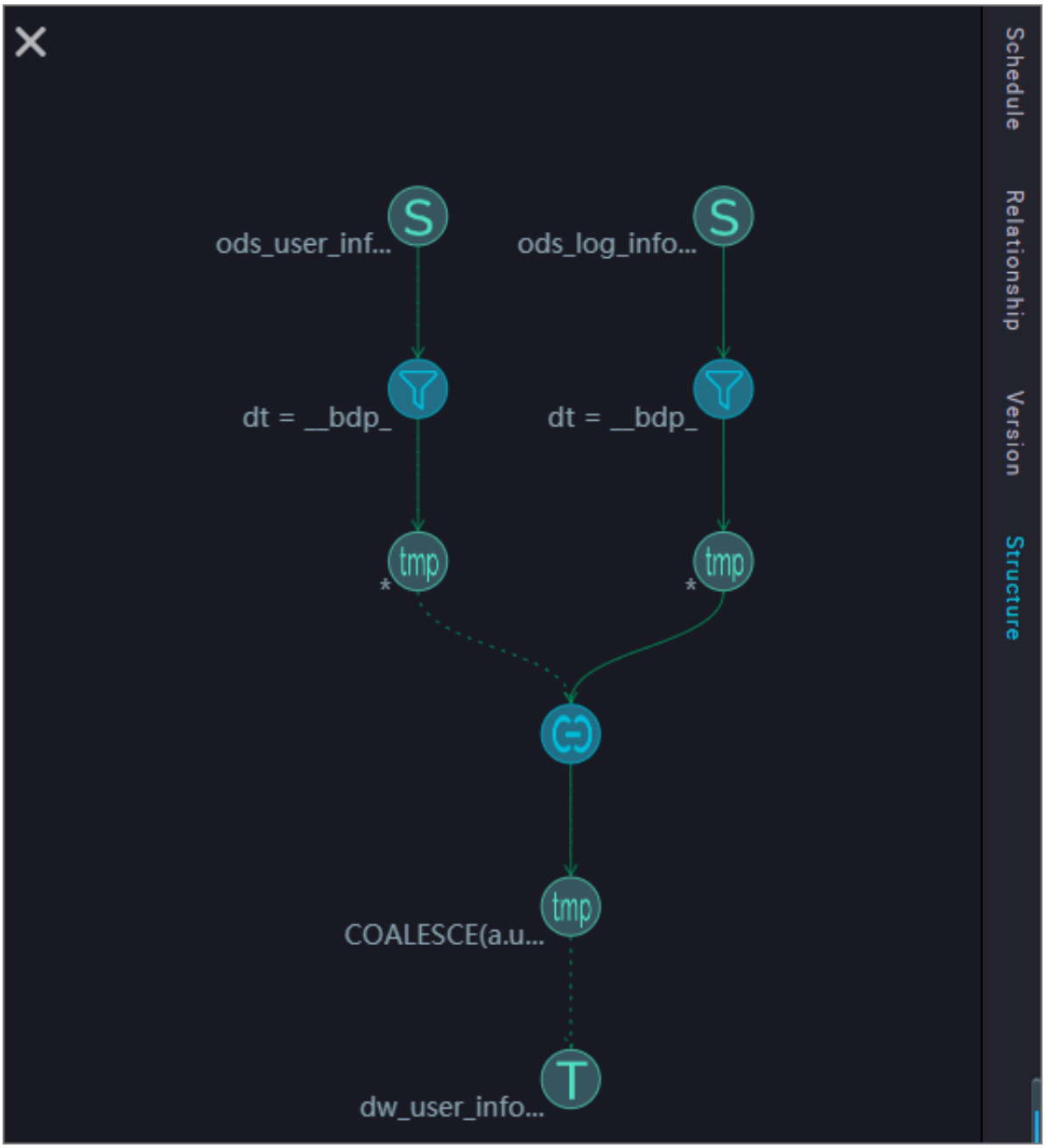
Structure

As shown in SQL:

```
INSERT   OVERWRITE   TABLE   dw_user_in  fo_all_d   PARTITION  ( dt
='${ bdp . system . bizdate }')
SELECT   COALESCE ( a . uid ,  b . uid )  AS   uid
 ,  b . gender
 ,  b . age_range
 ,  B .  flavdiac
 ,  a . region
 ,  a . device
 ,  a . identity
 ,  a . method
 ,  a . url
 ,  a . referer
 ,  a . time
FROM  (
  VALUES
  From   fig
  WHERE   dt  = ${ bdp . system . bizdate }
```

```
)  a
 LEFT   OUTER   JOIN  (
   VALUES
   FROM   ods_user_i  nfo_d
   WHERE   dt  = ${ bdp . system . bizdate }
)  b
 on   a . uid  =  b . uid  ;
```

According to this Code, the structure is parsed:



When the mouse is placed in a circle, the corresponding explanation is displayed:

1. **Source table: The target table for the SELECT query.**
2. **Filter: Filters the specific partitions in the table that you want to query.**

3. In the first part of the intermediate table (query view): Place the query data results into a temporary table.
4. Join: The mosaic of the results in the two-part query through join.
5. In the second section, the intermediate table (the query view): Summarizes the results of join in a temporary table. This temporary table exists for three days and is automatically cleared three days later.
6. Target table (insert): Inserts data obtained in the second part of the table in insert override.

1.3.4 Relationship

This topic describes relationships that displays the relations between the current node and other nodes. This relationship displays two parts: The dependency diagram and the internal relationship diagram.

Dependency graph

Depending on the node dependency, the dependency graph shows whether the current node dependency meets expectations. If the dependency graph does not meet expectations, you can return to the schedule configuration interface to reset.

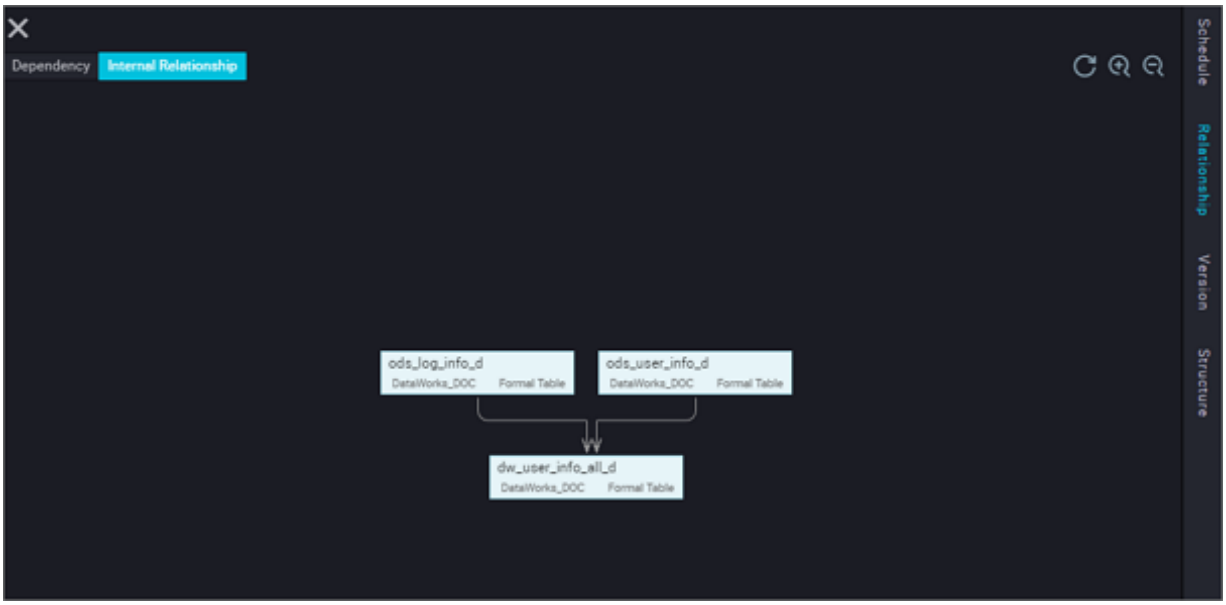
Internal relationship diagram

The internal relationship diagram is parsed based on the node code, for example:

```
INSERT  OVERWRITE  TABLE  dw_user_in  fo_all_d  PARTITION  ( dt
='${ bdp . system . bizdate }')
SELECT  COALESCE ( a . uid ,  b . uid )  AS  uid
```

```
  ,  b . gender
  ,  b . age_range
  ,  B .  flavdiac
  ,  a . region
  ,  a . device
  ,  a . identity
  ,  a . method
  ,  a . url
  ,  a . referer
  ,  a . time
 FROM  (
   VALUES
   From   fig
   WHERE   dt  = ${ bdp . system . bizdate }
)  a
 LEFT   OUTER   JOIN  (
   VALUES
   FROM   ods_user_i  nfo_d
   WHERE   dt  = ${ bdp . system . bizdate }
)  b
 on   a . uid  =  b . uid  ;
```

According to the preceding SQL, the parsed internal relationship map join " dw_user_info_all_d" with "ods_log_info_d", and export table as follows :



1.4 Business flow

1.4.1 Business flow

A business flow integrates different node task types by business type, such a structure improves business code development facilitation. The system organizes data development centered by the business flow, and provides container dashboards of various types of development nodes. In this way, tools, optimization operations, and

management operations are arranged based on data dashboards objects, making development and management more convenient and intelligent.

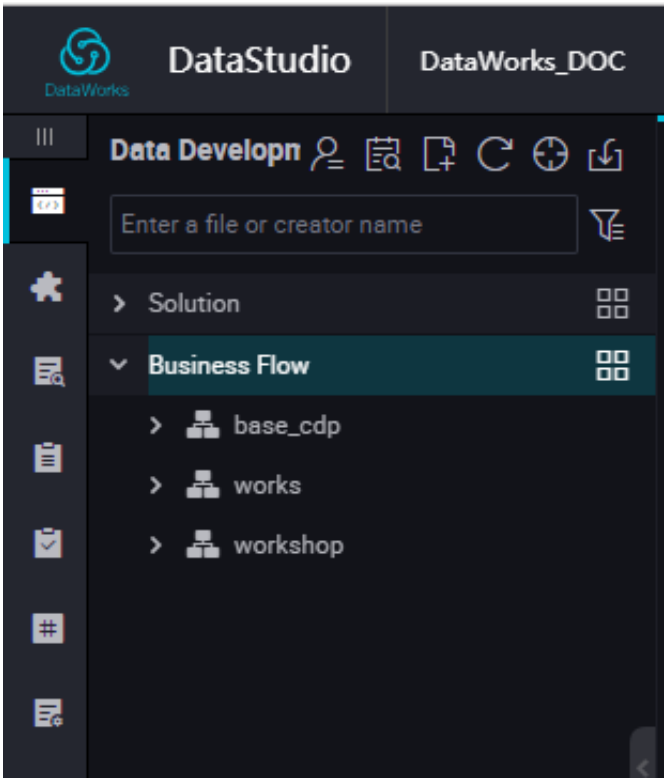
DataWorks code structure

A work project supports multiple types of computing engines. A work project contains multiple business flows, each of which is a collection of various types of objects that are systematically associated with each other. You can view each business flow in the automatically generated flowcharts. Objects in a process can be any of the following types: data integration task, data development task, table, resource, function, algorithm, and operation flow.

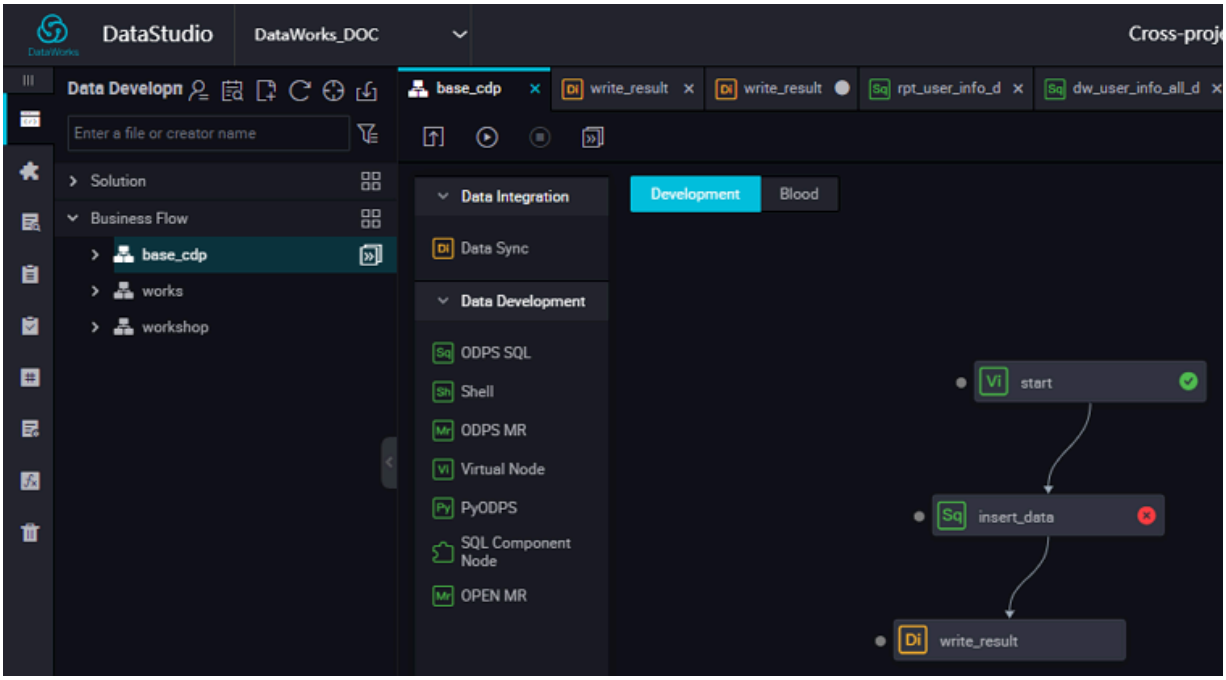
Each object type corresponds to an independent folder, in which sub-folders can be created. To facilitate management, we recommend that you create a maximum of four layers of sub-folders. The planned business flow structure becomes too complex when more than four layers of sub-folders are created. We recommend that you split the business flow into one or more business flows and manage the related business flows in one solution. This business flow organization method is more efficient.

Business flow composition

1. Data Integration: For more information about Data Integration, see #unique_13.
2. Data Development: For more information about Data Integration, see #unique_14.
3. Table: For more information about Data Integration, see #unique_15.
4. Resources: For more information about Data Integration, see Introduction to resources.
5. Functions: For more information about Data Integration, see Introduction to functions.

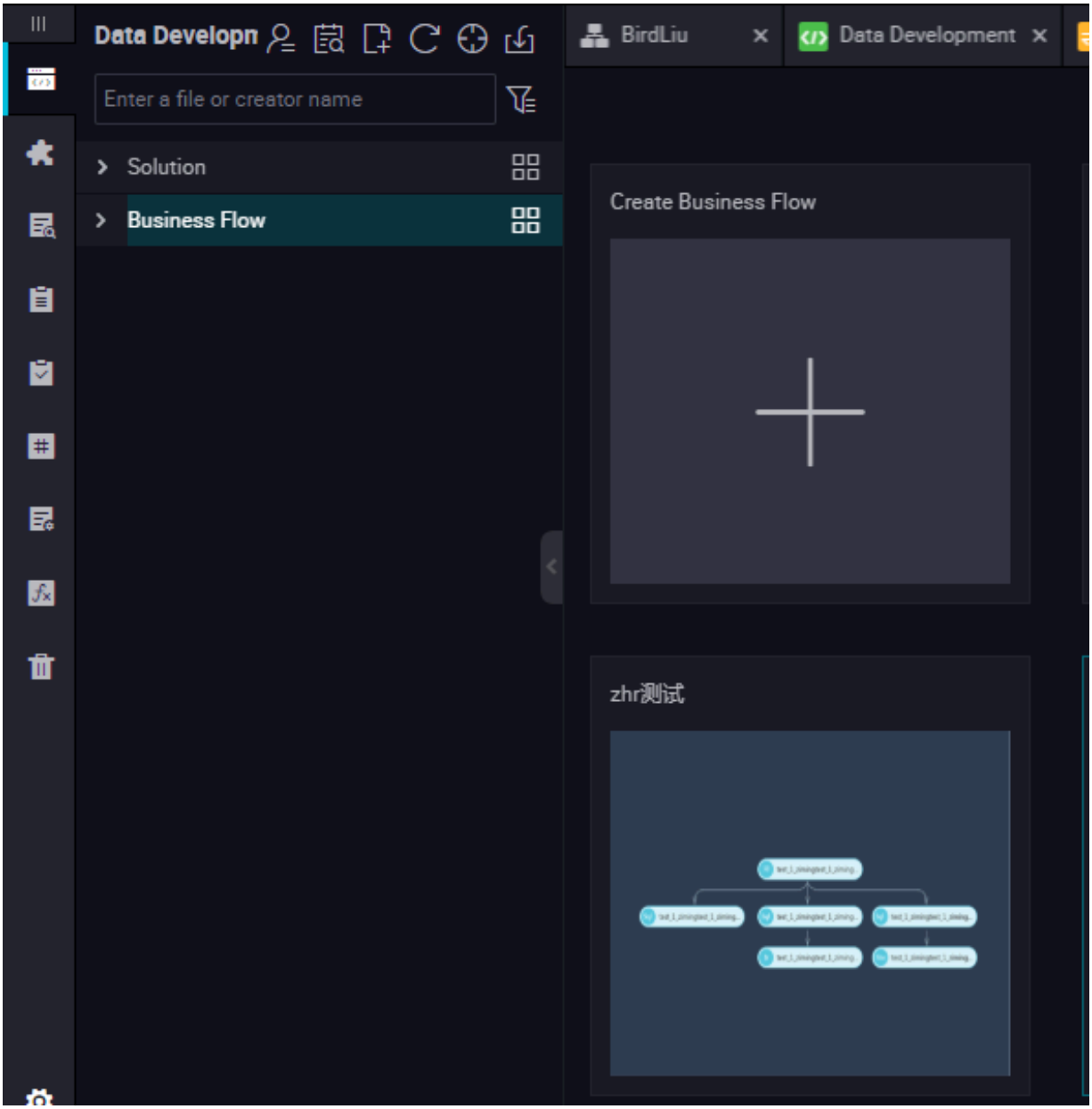


Double-click the name of a Business Flow node to view the relationship between nodes of the business flow in a workflow chart.



Business flow dashboard

You can check all business flows under a project on the business flow dashboard.

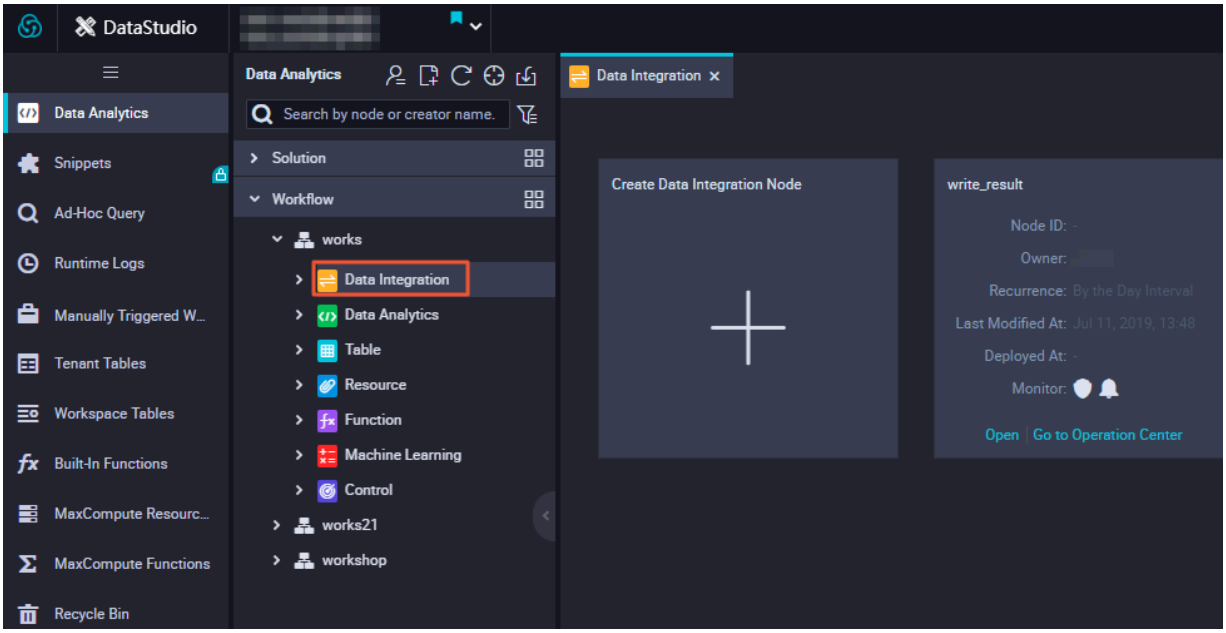


Business flow object dashboard

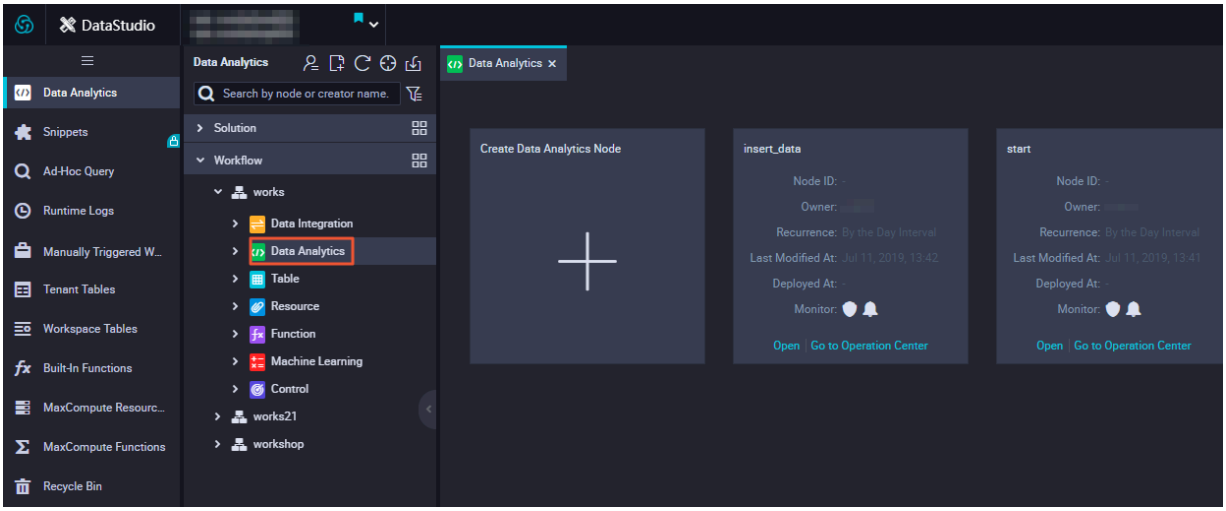
An object set dashboard is created for each object type in a business flow, and each object corresponds to an object card on the dashboard. You can attach the operation and optimization suggestions to the corresponding object, so that the object management is intelligent and convenient.

For example, on the object card of the data development task, the baseline strong protection and custom reminder icons are displayed, facilitating you to understand the current task protection status. You can double-click the icon of each object under the Business Flow to open the dashboard of the object type.

Data Integration task dashboard



Data Development task dashboard

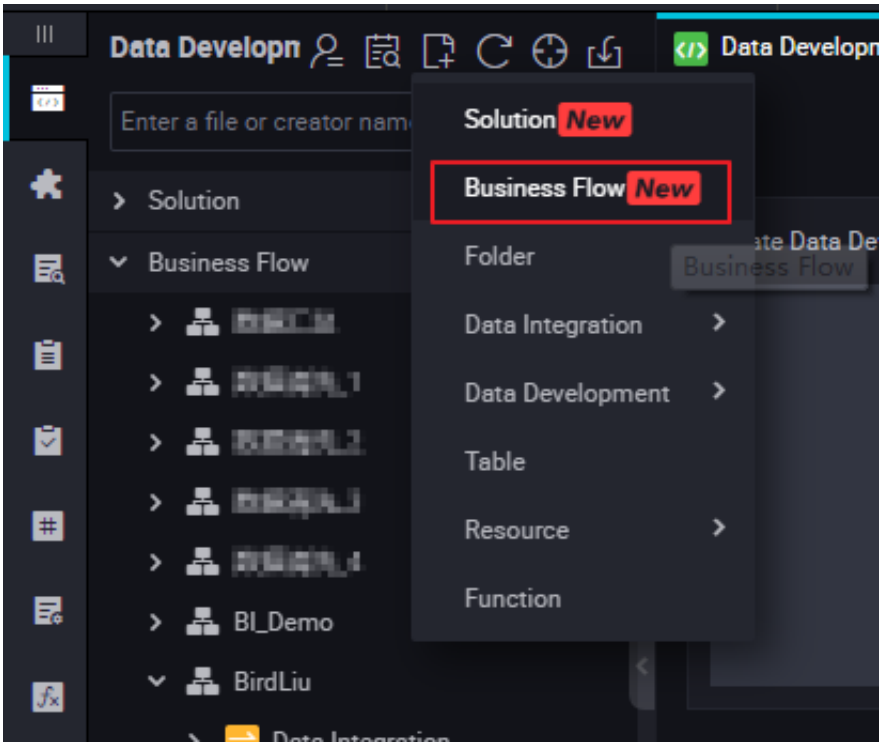


Note:

The number of nodes in a single business flow cannot exceed 100.

Create a business flow

Right-click Business Flow under Data Development, select Create Business Flow.



1.4.2 Resource

This topic describes how to create, upload, reference, and download resources.

If you want to use .jar, you need to upload the file to the project resource. You can upload text files, MaxCompute tables, and various compressed package formats, including .zip, .tgz, .tar.gz, .tar, and .jar as different types of resources to MaxCompute. Then, you can read or use these resources while running UDFs or MapReduce.

MaxCompute provides APIs for reading and using resources. The following types of MaxCompute resources are available:

- File
- Archive: The compression type is identified by the extension in the resource name. The following compressed file types are supported: .zip, .tgz, .tar.gz, .tar, and .jar.
- JAR: The compiled Java jar packages.

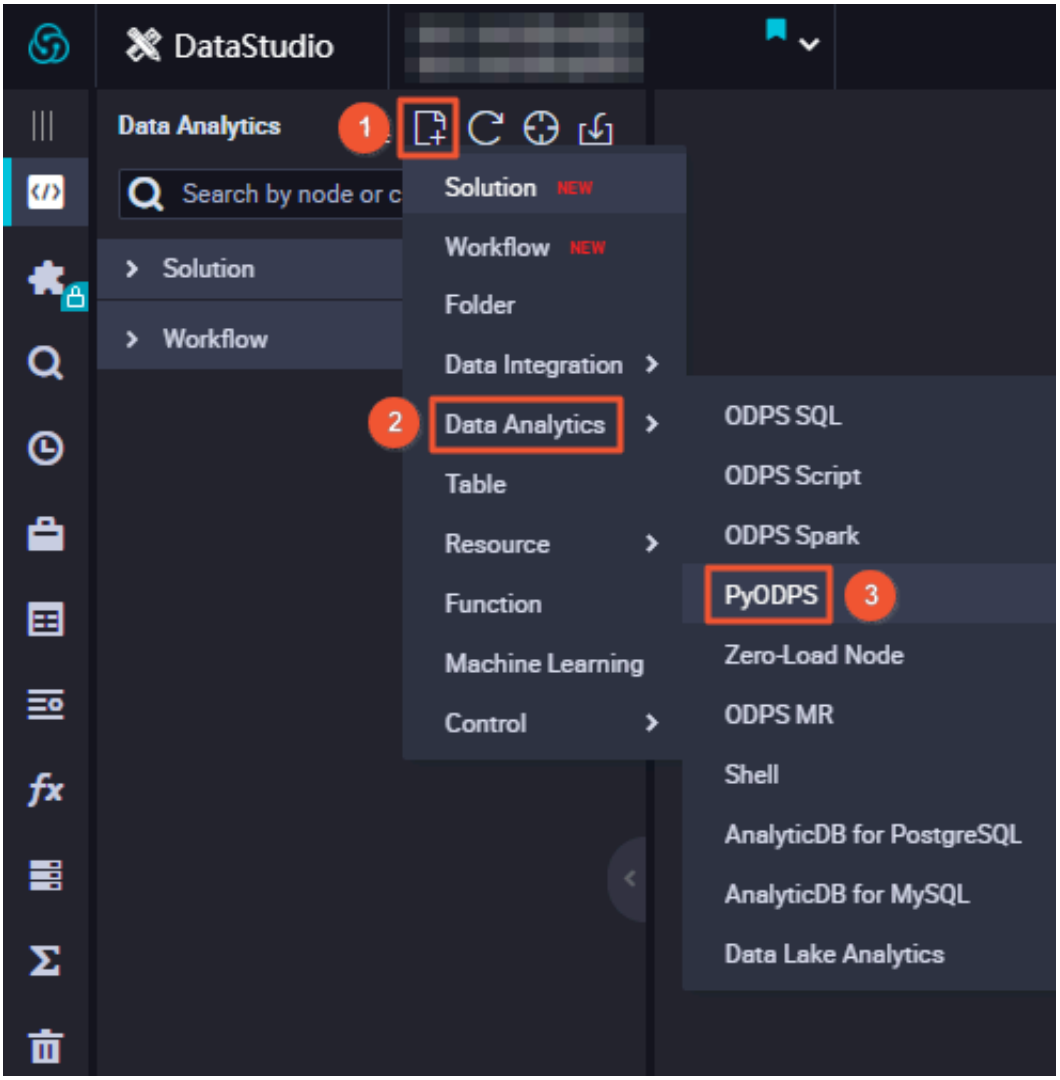
In DataWorks, to create a resource you need to add a resource. Currently, DataWorks supports adding three resource types in a visual manner, including the .jar and file resources. The newly created entries are the same, but the differences are as follows:

- JAR resource: You need to compile the Java code in the offline Java environment, compress the code into a JAR package, and upload the package as the JAR resource to ODPS.
- Small files: These resources are directly edited on DataWorks.
- File resource: Select a large file when you create file resources. You can also upload local resource files.

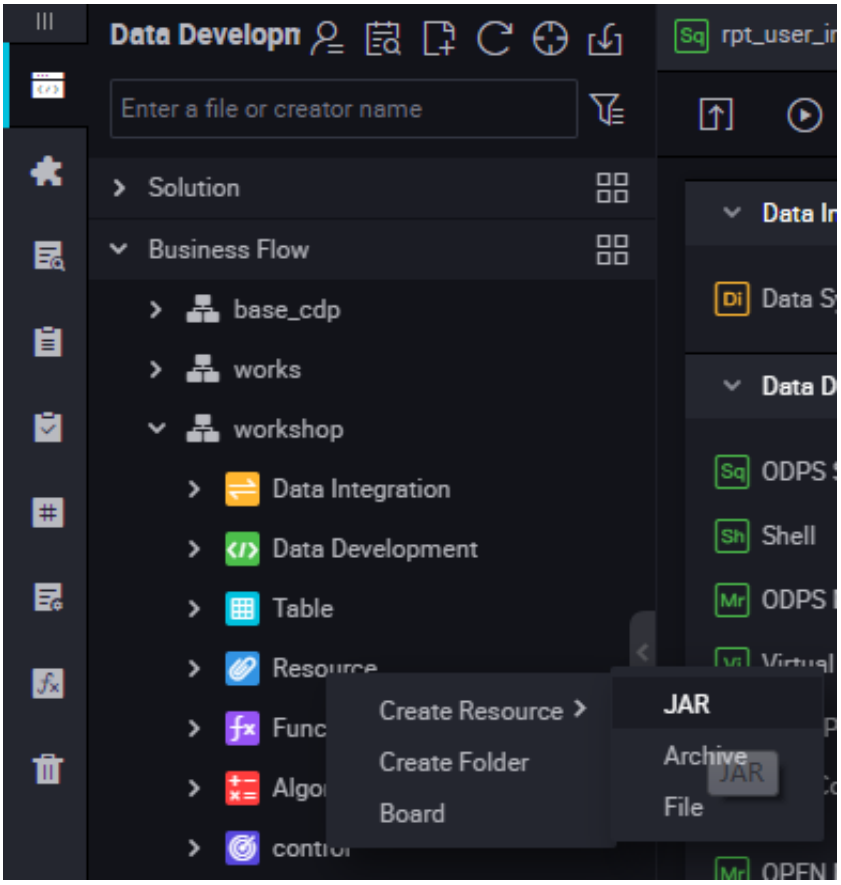
Note:
The resource package for upload cannot exceed 30 MB.

Create a resource instance

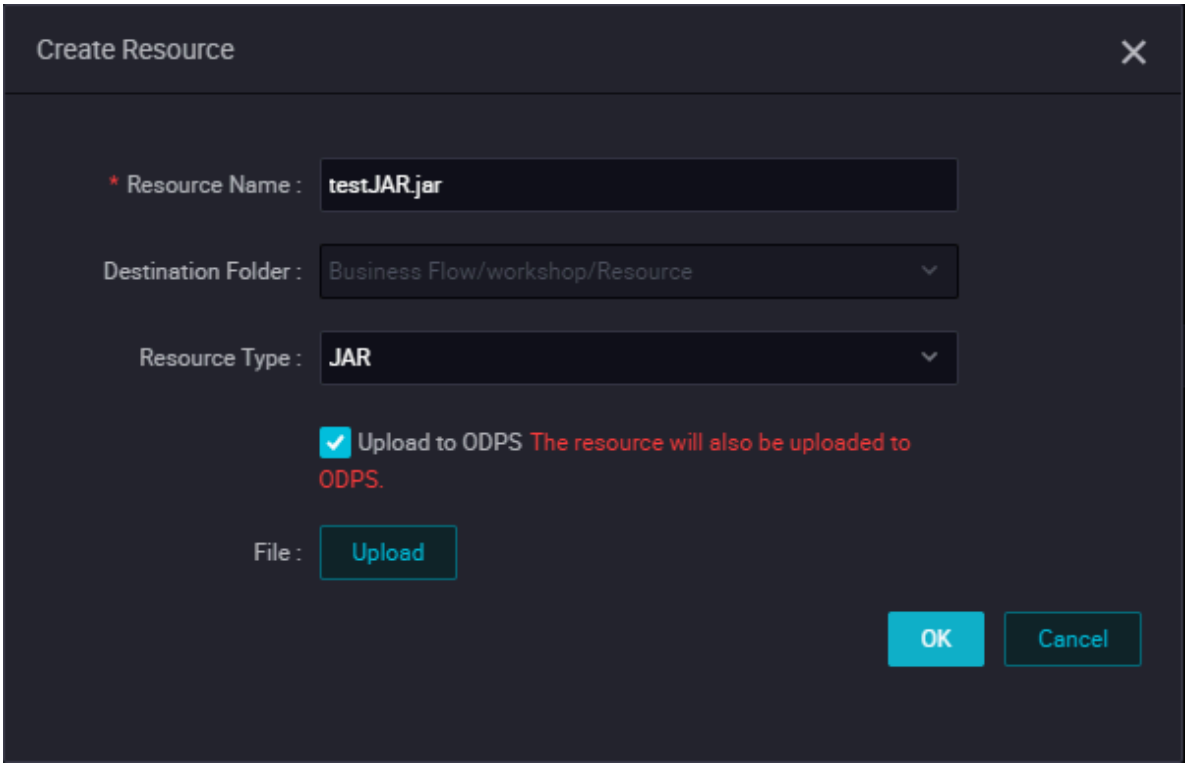
1. Right-click Business Flow under Data Development, and select Create Business Flow.



2. Right-click Resource, and select Create Resource > JAR.



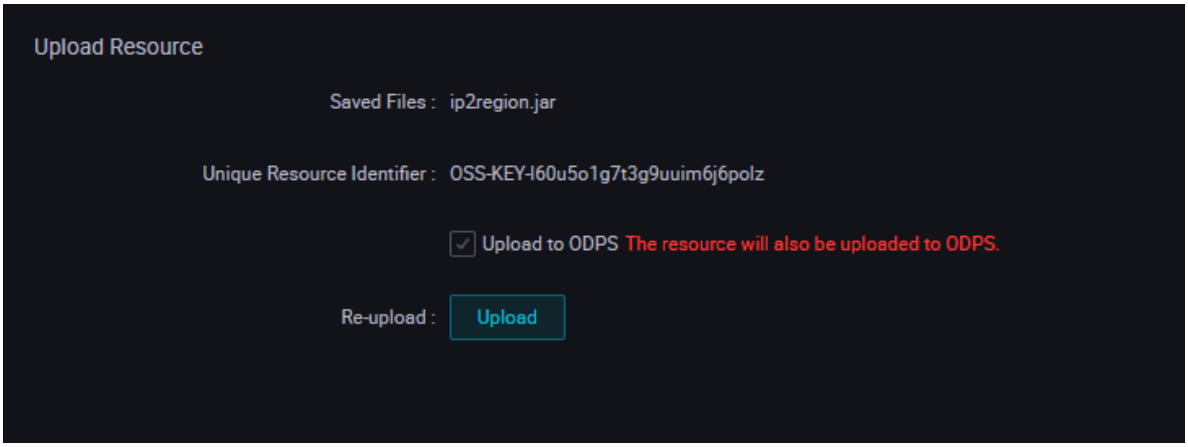
3. The Create Resource dialog box is displayed. Enter the resource name according to the naming convention, and set the resource type to JAR. Select a local JAR package for upload, and click OK to submit the package in the development environment.



Note:

- If this JAR package has been uploaded to the MaxCompute client, you must deselect Upload to ODPS . Otherwise, an error will occur during the upload process.
- The resource name is not always the same as the uploaded file.
- The naming convention for a resource name: A string can contain 1 to 128 characters, including letters, numbers, underscores (_), and periods (.). The name is case insensitive. If the resource is a JAR resource, the extension is .jar.

4. Click OK to submit the resource to the development scheduling server.



5. Release a node task.

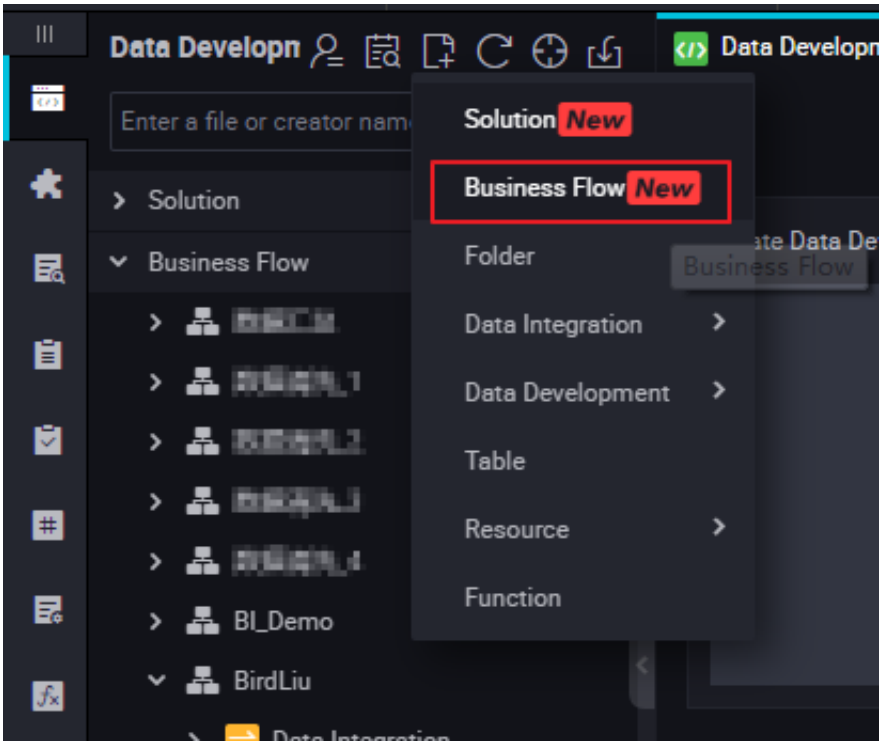
   For more information about operations, see #unique_19.

1.4.3 Register the UDFs

Currently, the Python and Java APIs support UDFs implementation. To compile a UDF program, you can upload the UDF code by Adding resources and then register the UDF.

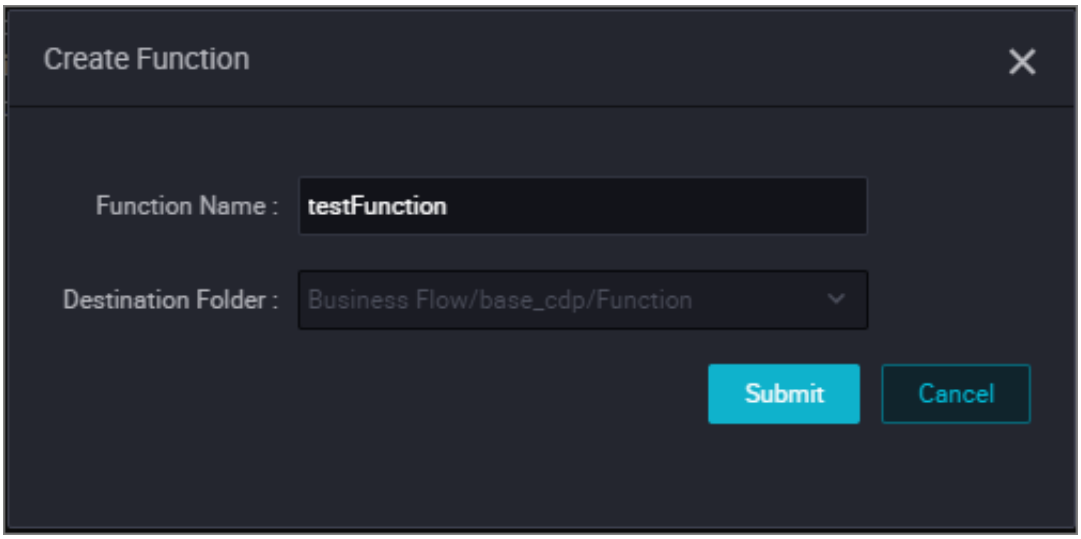
UDF registration procedure:

1. Right-click Business Flow under Data Development, and select Create Business Flow.

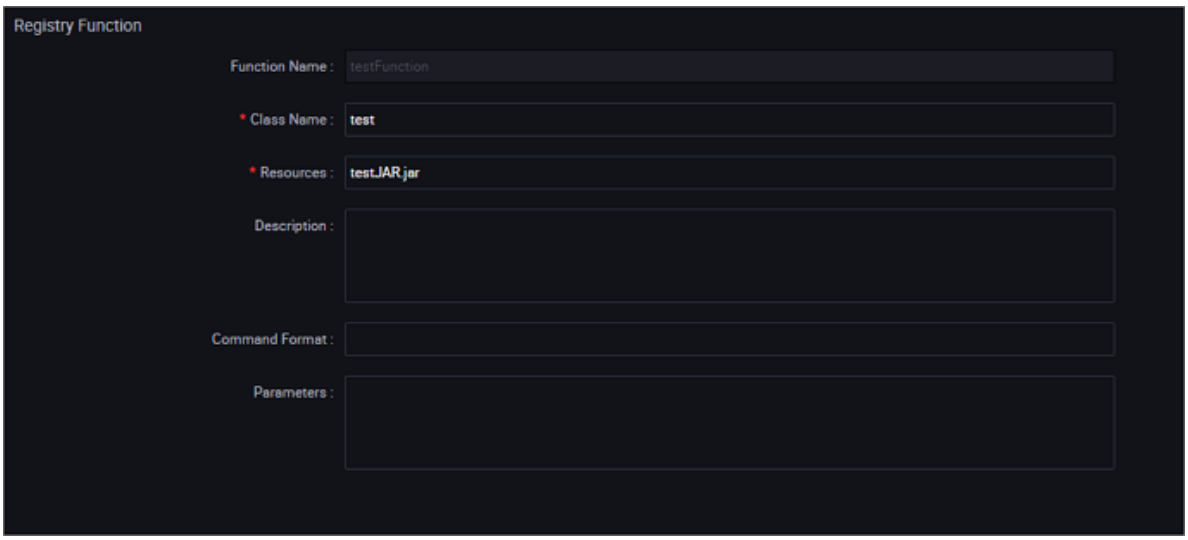


2. In the offline Java environment, edit the program, compress the program into a JAR package, create a JAR resource, submit and release the program. For more information about the Java environment, see Create resources.
3. Create a function.

   Right-click Function, select Create Function, and enter the new function configuration.



4. Edit the function configuration.



   - Class name: The main class name that implements the UDF.
   - Resource list: The resource name in the second step. If there are multiple resources, separate them with commas (,).
   - Description: The UDF description. It is optional.

5. Submit the task.

   After the configuration is completed, click Save in the upper-left corner of the page or press Ctrl+S to Submit (and Unlock) the node in the development environment.

6. Release a task

   For more information about the operation, see #unique_19.

1.5 Node type

1.5.1 Node types overview

This topic describes how to apply the seven different node types in DataWorks in different scenarios.

Virtual node

A virtual node is a control node that does not generate any data. The virtual node is generally used as the root node for planning the overall node workflow. For more information about virtual nodes, see #unique_23.

Note:
The final workflow output table contains multiple branch input tables. Virtual nodes are usually used if these input tables do not have any dependencies between them.

ODPS SQL node

An ODPS SQL task allows you to edit and maintain the SQL code on the Web, and easily implement code runs, debug, and collaboration. DataWorks also provides code version management, automatic resolution of upstream and downstream dependencies, and other features. For more information about the examples, see #unique_24.

By default, DataWorks uses the MaxCompute project as the space for development and production, so that the code content of the MaxCompute SQL node follows the MaxCompute SQL syntax . MaxCompute SQL syntax is similar to Hive, which can be considered a subset of the standard SQL. However, MaxCompute SQL cannot be equated with a database because it does not possess the following database features: transactions, primary key constraints, and indexes.

For more information about MaxCompute SQL syntax, see SQL overview.

ODPS MR node

MaxCompute supports MapReduce programmed APIs, whose Java APIs can be used to compile the MapReduce program for data processing in MaxCompute. You can create MaxCompute MR nodes and use them for task scheduling. For more information about the examples, see #unique_25.

PyODPS node

The Python SDKin MaxCompute can be used to operate MaxCompute.

The PyODPS node in DataWorks can be integrated with MaxCompute Python SDK. You can edit the Python code to operate MaxCompute on a PyODPS node in DataWorks. For more information, see #unique_26.

SQL component node

An SQL component node is an SQL code process template that contains multiple input and output parameters. To handle an SQL code process, you need to import, filter, join, and aggregate one or more data source tables to form a target table required for new business. For more information, see#unique_27.

Data integration node

A data integration node is a stable, efficient, and automatically scalable external data synchronization cloud service provided by the Alibaba Cloud DTplus platform. With the data synchronization node, you can easily synchronize data in the business system to MaxCompute. For more information, see #unique_13.

1.5.2 Data integration node

Currently, the data integration task supports the following data sources: MaxCompute, MySQL, DRDS, SQL Server, PostgreSQL, Oracle, MongoDB, DB2, OTS, OTS Stream, OSS, FTP, Hbase, LogHub, HDFS, and Stream. For details about more supported data sources, see #unique_29.

Configure a integration task

For more information, see #unique_30/unique_30_Connect_42_section_tfn_1kc_p2b

Node scheduling configuration.

Click the Scheduling Configuration on the right of the node task editing area to go to the node scheduling configuration page. For more information, see Scheduling configuration.

Submit the node

After the configuration is completed, click Save in the upper left corner of the page or press Ctrl+S to submit (and unlock) the node to the development environment.

Publish a node task

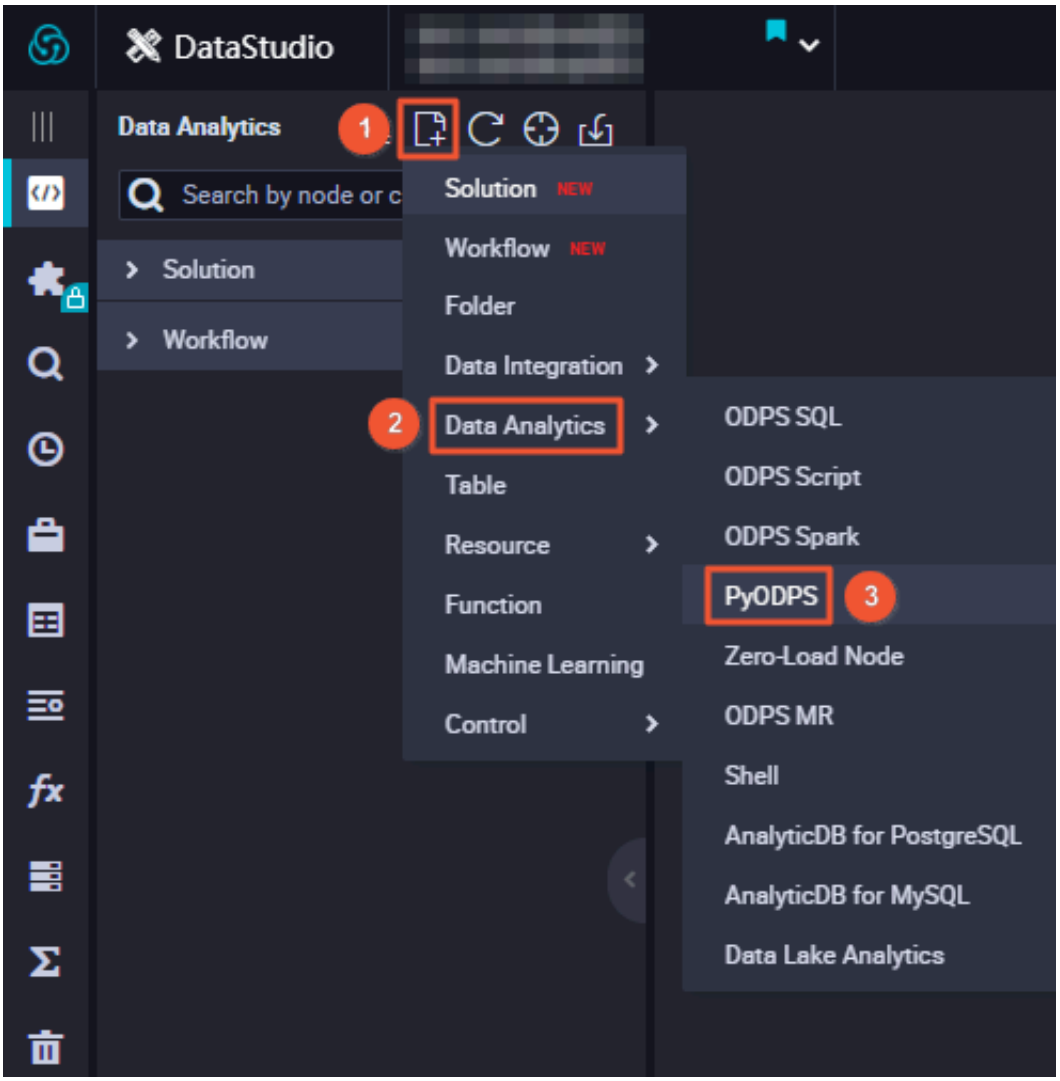
For more information about the operation, see Release management.

Test in the production environment.

For more information about the operation, see #unique_32.

1.5.3 MaxCompute SCRIPT node

1. On the DataStudio page, move the cursor over the Create icon and select Business Flow. The Create Business Flow dialog box appears.



2. On the DataStudio page, move the cursor over the Create icon and choose Data Analytics > ODPS SCRIPT.

3. Edit the MaxCompute SCRIPT node.

   You can edit the script code of the node. For more information, see #unique_34.

4. Set scheduling parameters of the node.

   Click Schedule on the right of the node editing area to go to the node scheduling configuration page. For more information, see Scheduling configuration.

5. Submit the node.

   After the scheduling configuration is completed, click Save in the upper-left corner of the page to submit and unlock the node to the development environment.

6. Publish the node.

   For more information, see Publish management.

7. Test the node in the production environment.

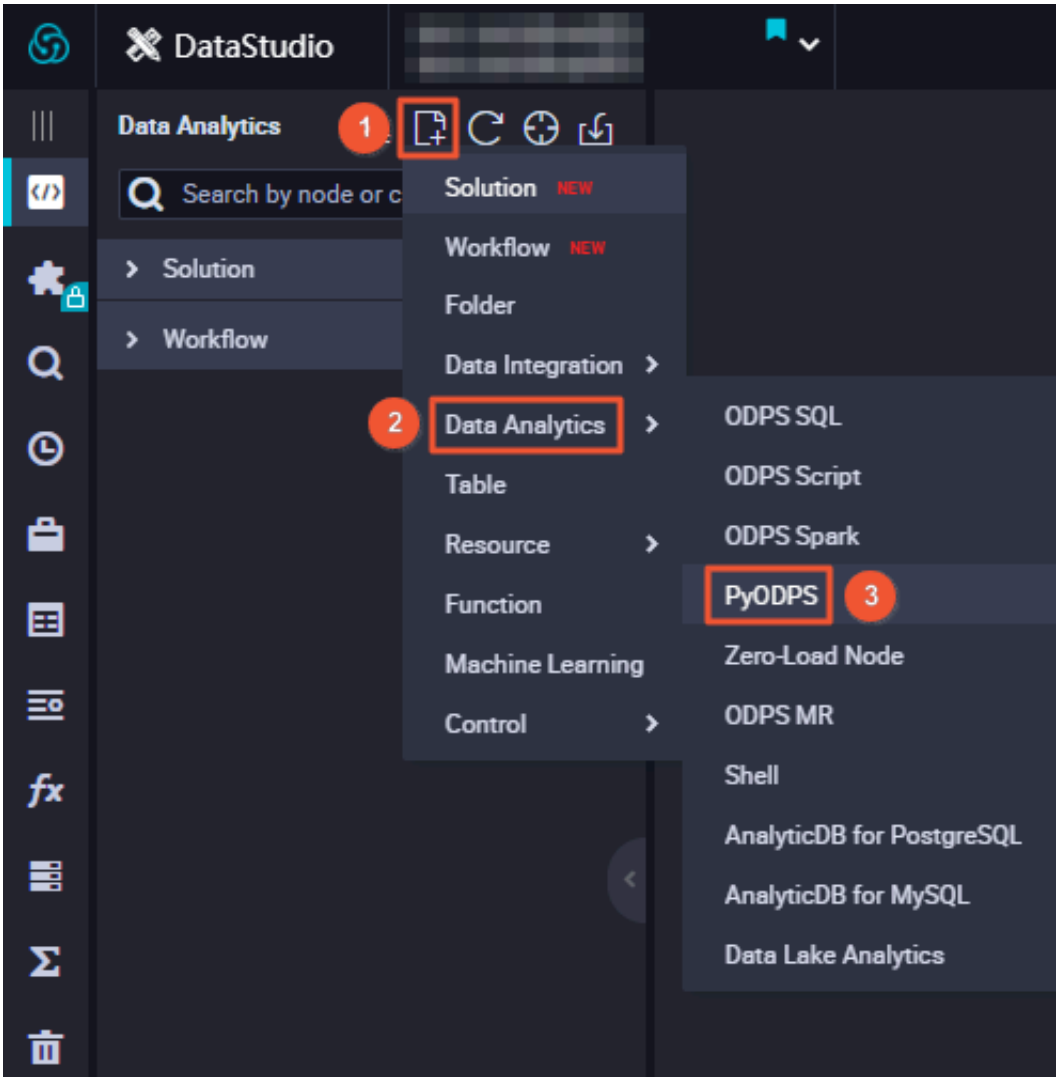
   For more information, see #unique_32.

1.5.4 ODPS SQL node

This topic describes the ODPS SQL node functions. The ODPS SQL node syntax is similar to SQL, and is suited for distributed scenario with massive data volume at the TB-level, but has low real-time requirements. The ODPS SQL node is an OLAP application oriented throughput. We recommend you use ODPS SQL if your business needs to handle tens of thousands transactions because it requires a long period to complete the job process from preparation to submission.

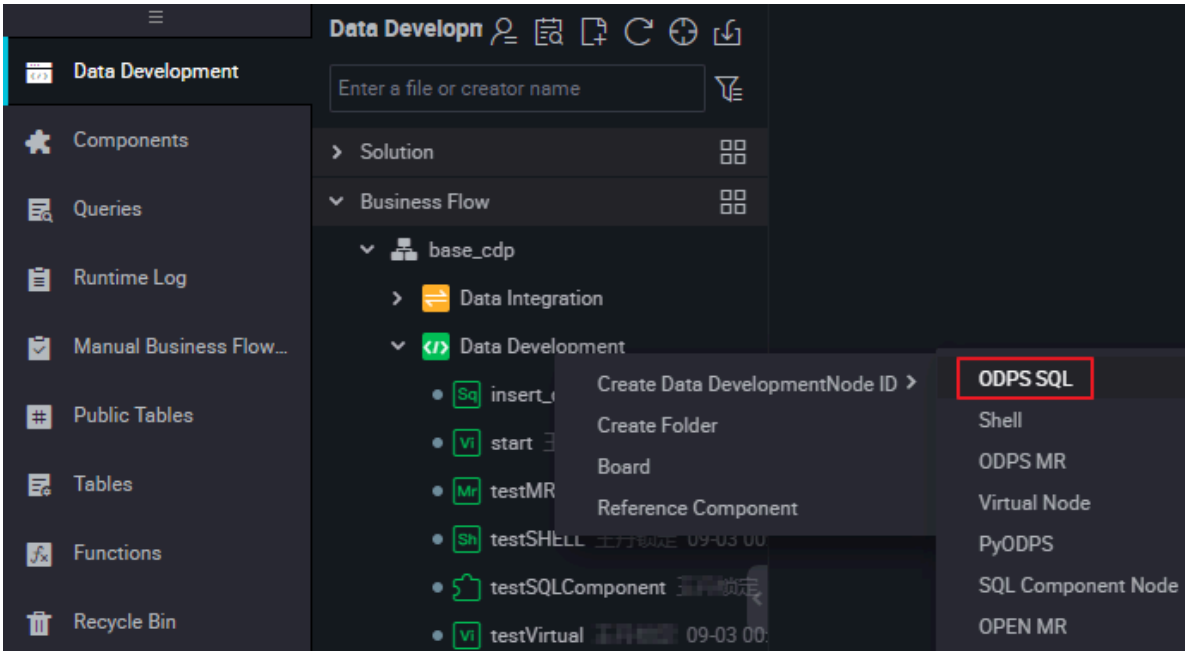
1. Create a business flow.

   Right-click Business Flow under Data Development, and select Create Business Flow.



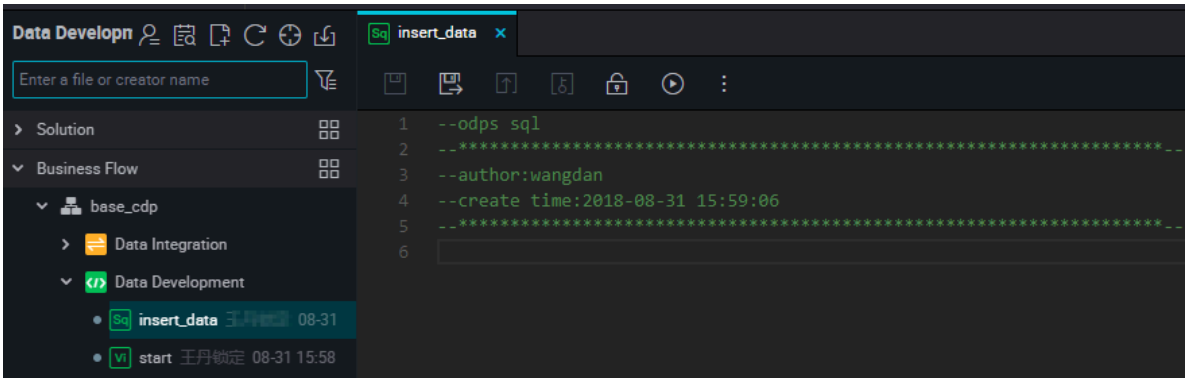
2. Create ODPS SQL node.

   Right-click Data Development, and select Create Data Development Node > ODPS SQL.



3. Edit the node code.

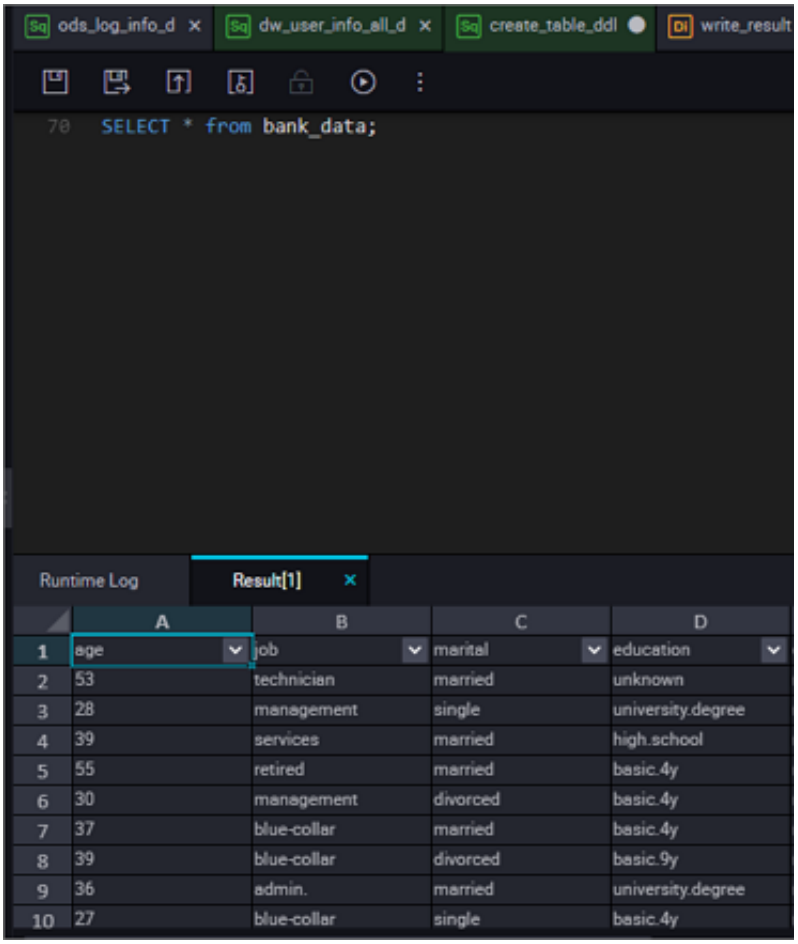
   For more information about the SQL syntax statements, see MaxCompute SQL statements.



4. Query result display

   DataWorks query results are connected to the spreadsheet function, making it easier for users to operate the data results.

   The query results are displayed in spreadsheet style. Users can perform operations in DataWorks, open it in a spreadsheet, or freely copy content stations in local excel files.



	A	B	C	D
1	age	job	marital	education
2	53	technician	married	unknown
3	28	management	single	university.degree
4	39	services	married	high.school
5	55	retired	married	basic.4y
6	30	management	divorced	basic.4y
7	37	blue-collar	married	basic.4y
8	39	blue-collar	divorced	basic.9y
9	36	admin.	married	university.degree
10	27	blue-collar	single	basic.4y

   - Hide column: Select one or more columns to hide the column.
   - Copy row: Select one or more rows that need to be copied to the left side, and click Copy Row.
   - Copy column: The top column selects a column or more points that need to be copied to the selected column.
   - Copy: You can freely copy the selected content.
   - Search: The search bar is displayed in the upper-right corner of the query results for facilitating data search in the table.

5. Node scheduling configuration.

   Click Schedule on the right of the node task editing area to go to the Node Scheduling Configuration page. For more information about node scheduling configuration, see Scheduling configuration.

6. Submit the node.

   After the configuration is completed, click Save in the upper-left corner of the page or press Ctrl+S to submit (and unlock) the node to the development environment.

7. Publish a node task.

   For more information about the operation, see Release management.

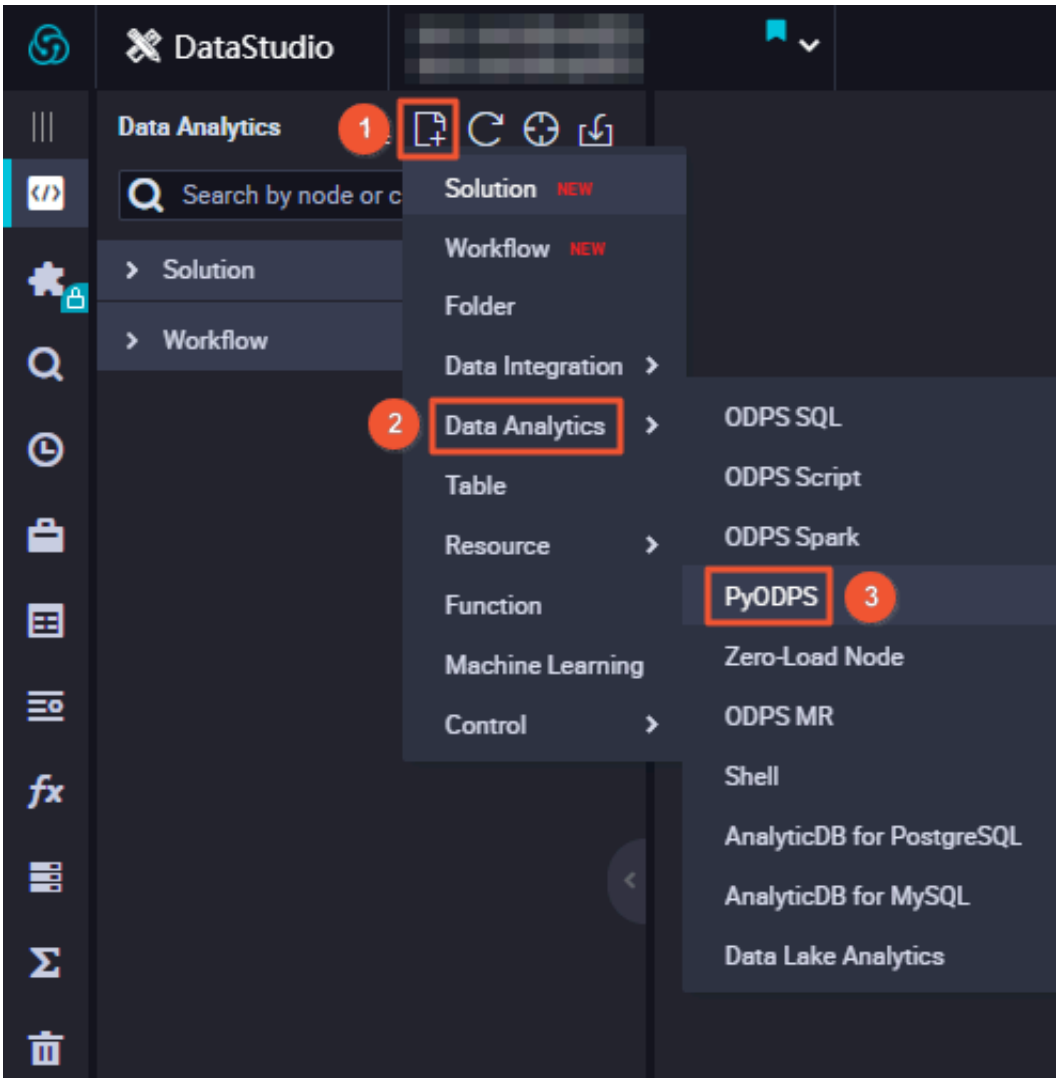
8. Test in the production environment.

   For more information about the operation, see #unique_32.

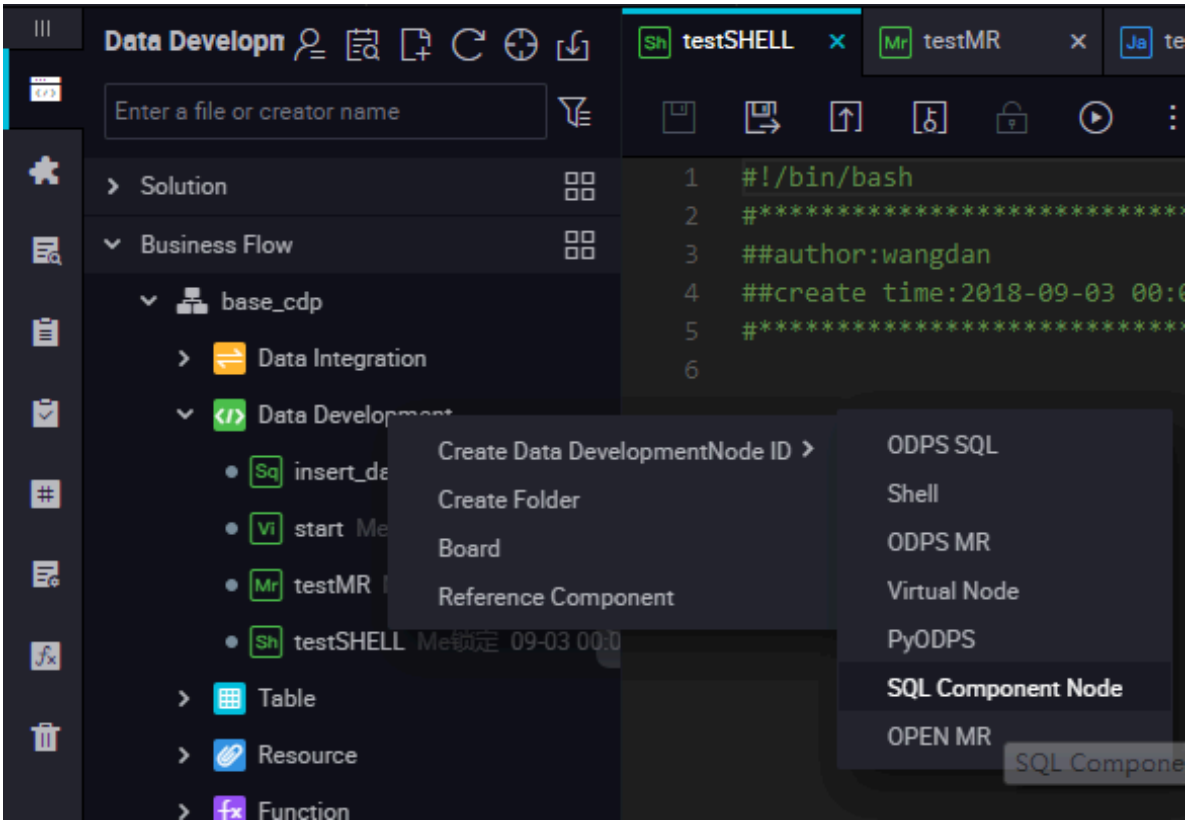
1.5.5 SQL Component node

Procedure

1. Right-click the Business Flow under Data Development, and select Create Business Flow.



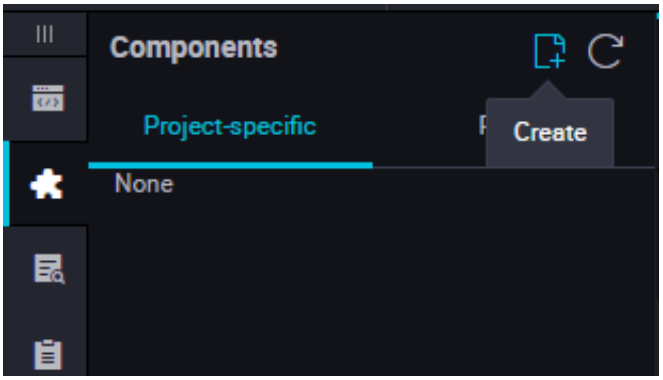
2. Right-click Data Development, and select Create Data Development Node > SQL Component Node.



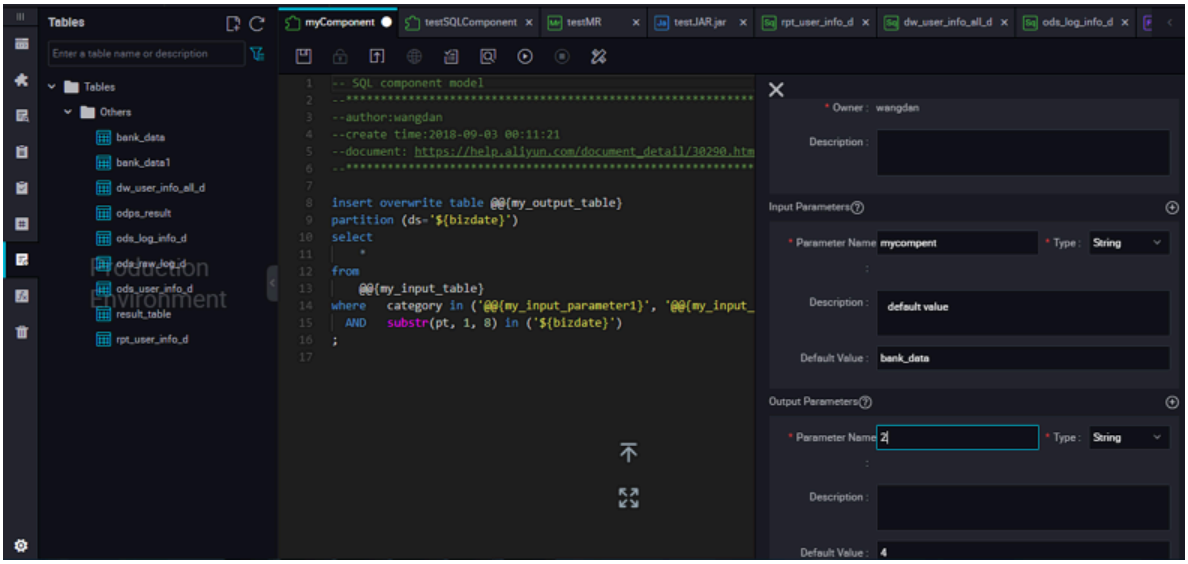
3. To improve the development efficiency, the data task developers can use components contributed by project and tenant members to create data processing nodes.

   - Components created by members of the local project are under Project Components.
   - Components created by tenant members are located under Public Components.

   When you create a node, set the node type to SQL Component node, and specify the node name.



   Specify parameters for the selected component.



   Enter the parameter name, and set the parameter type to Table or String.

   Specify the three get_top_n parameters in sequence.

   Specify the following input table for the Table type: test_project.test_table parameters.

4. Node scheduling configuration.

   Click the Scheduling Configuration on the right of the node task editing area to go to the Node Scheduling Configuration page. For more information, see Scheduling configuration.

5. Submit a node.

   After completing the configuration, click Save in the upper-left corner of the page or press Ctrl+S to submit (and unlock) the node in the development environment.

6. Publish a node task.

   For more information about the operation, see Publish management.

7. Test in a production environment.

   For more information about the operation, see #unique_32.

Upgrade the SQL component node version

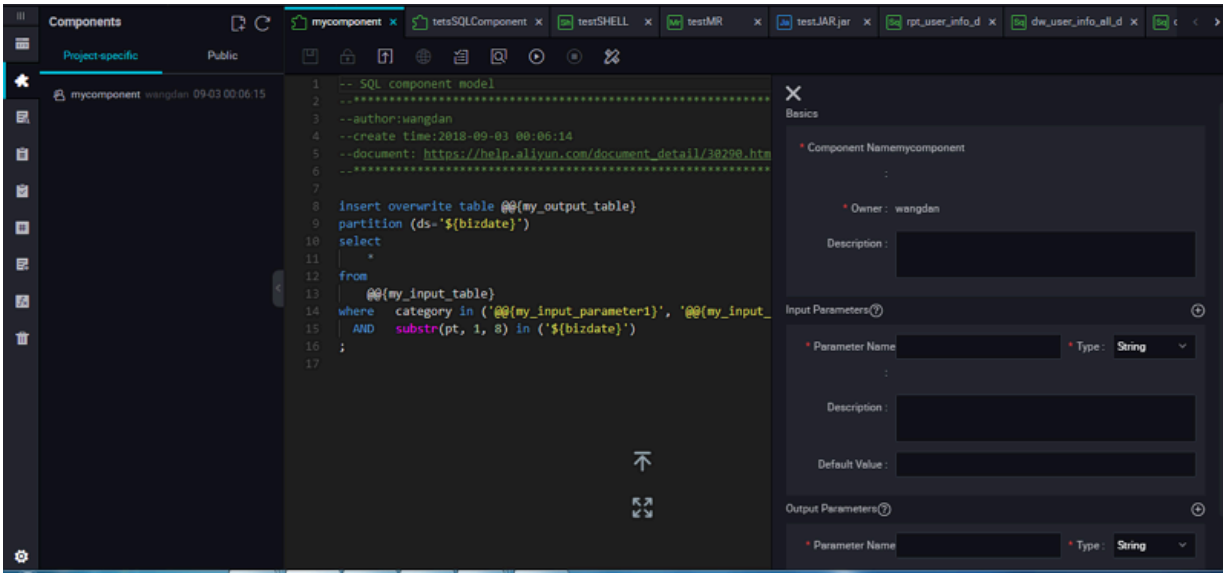
After the component developer releases a new version, the component users can choose whether to upgrade the used instance of the existing component to the latest used component version.

With the component version mechanism, developers can continuously upgrade components and component users can continuously enjoy the improved process execution efficiency and optimized business effects after upgrading the components.

For example, user A uses the v1.0 component developed by user B, and user B upgrades the component to V.2.0. User A can still use the v1.0 component after the upgrade, but will receive an upgrade reminder. After comparing the new code with the old code, user A finds that the business effects of the new version are better than that of the old version, and therefore can determine whether to upgrade to the latest version of the component.

You can easily upgrade an SQL component node based on the component template, by selecting Upgrade. After checking whether the SQL component node parameter settings are effective in the new version, and then make some adjustments based on the new version component instructions, and then submit and release the node similar to a common SQL component node.

Interface functions



The interface features are described below:

No.	Feature	Description
1	Save	Saves the current component settings.
2	Steal lock edit	Steals lock edit of the node if you are not the owner of the current component.
3	Submit	Submit the current component in the development environment.
4	Publish component	Publish a universal global component to the entire tenant, so that all users in the tenant can view and use the public component.
5	Resolve input and output parameters	Resolve the input and output parameters of the current code.
6	Precompilation	Edit the custom and component parameters of the current component.
7	Run	Run the component locally in the development environment.
8	Stop run	Stop a running component.
9	Format	Sort the current component code by keyword.
10	Parameter settings	View the component information, input parameter settings, and output parameter settings.
11	Version	View the submission and release records of the current component.

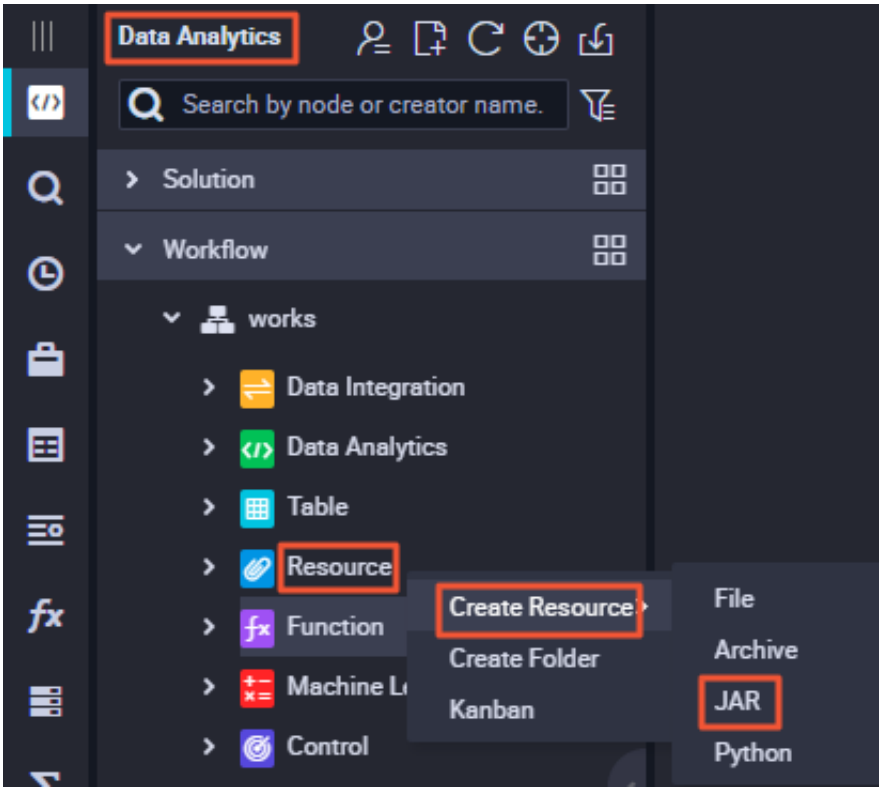
No.	Feature	Description
12	Reference records	View the usage record of the component.

1.5.6 ODPS Spark node

DataWorks supports the ODPS Spark node type. This topic describes how to create and configure an ODPS Spark node.

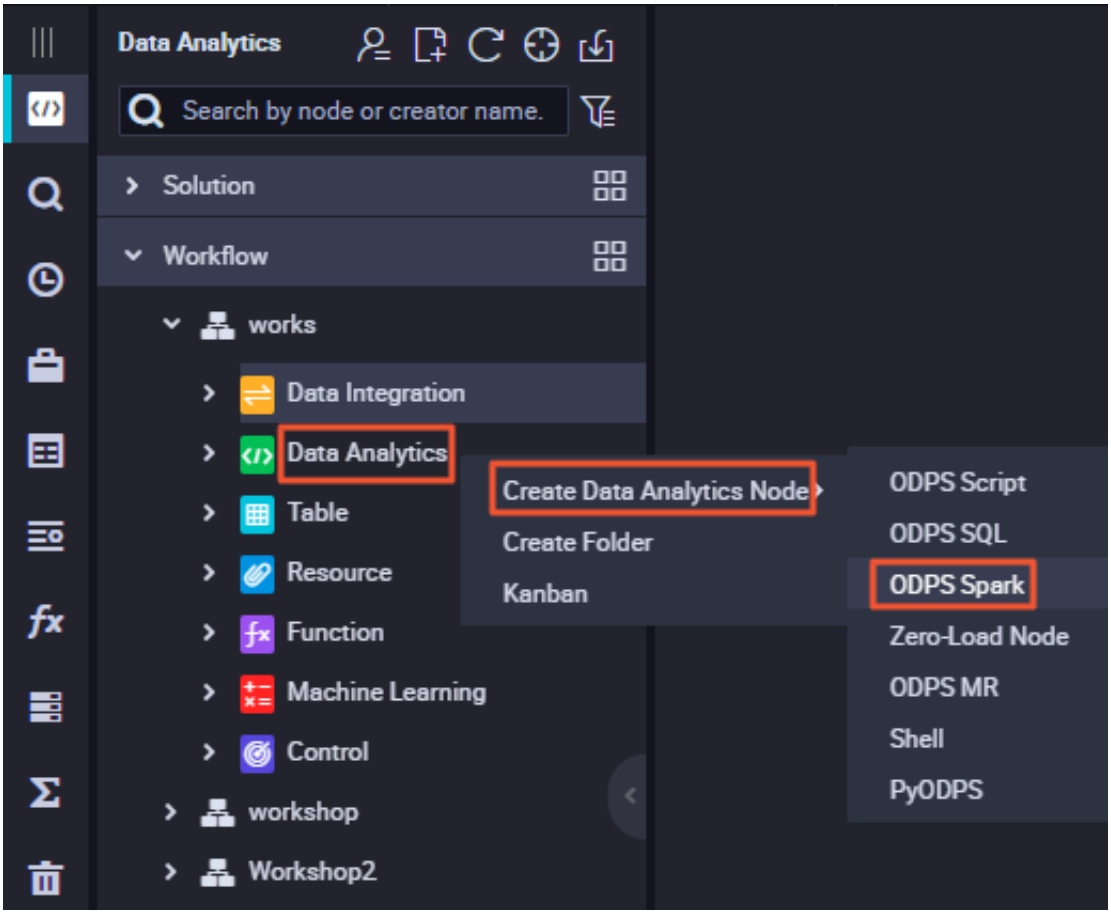
WordCount

1. In Data Analytics, right-click Workflow and select Create Workflow.
2. Right-click Resource, choose Create Resource > JAR, and upload the compiled JAR package.

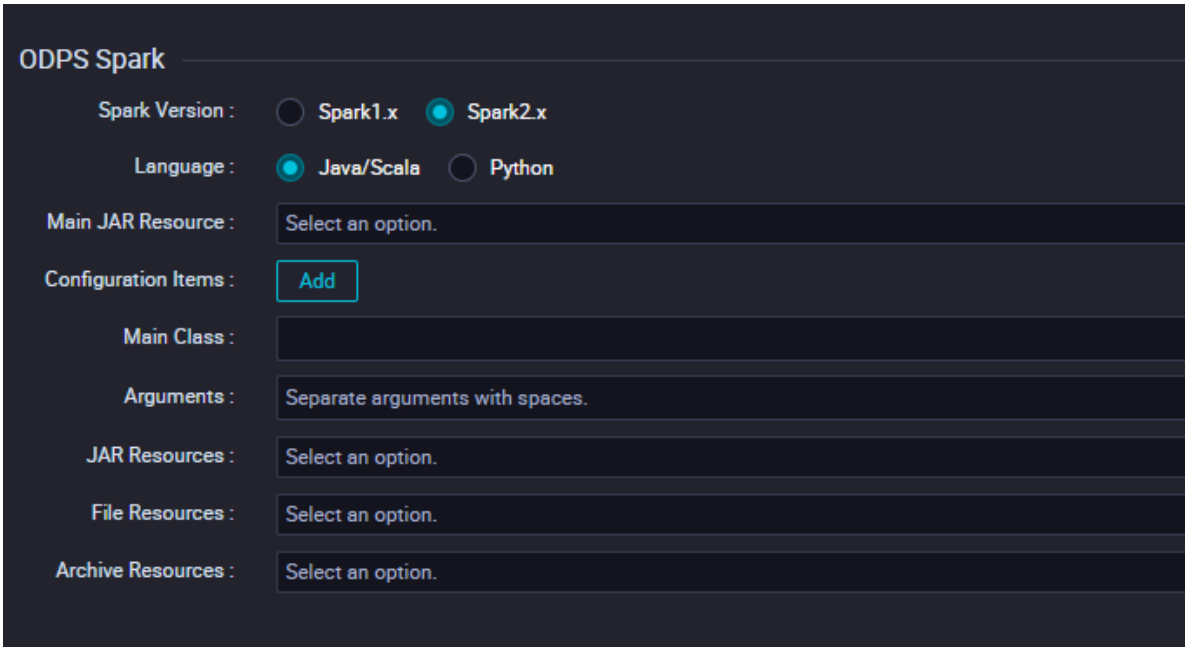


For more information about the WordCount sample code, see #unique_38.

3. Right-click Data Analytics under Workflow and choose Create Data Analytics Node > ODPS Spark to create an ODPS Spark node.



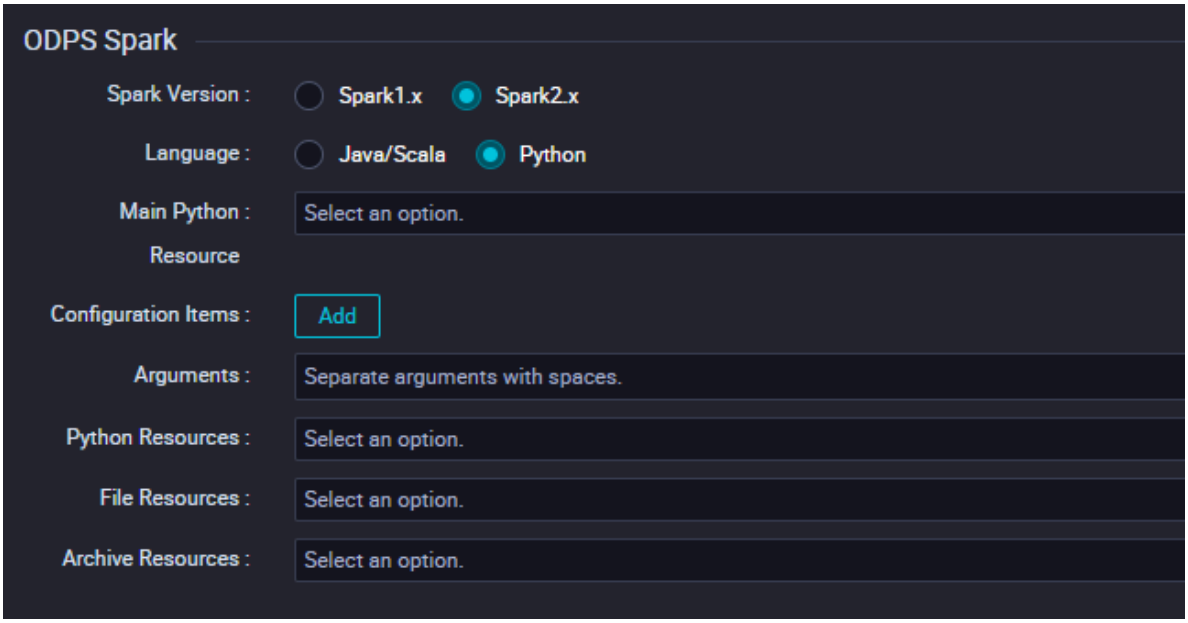
4. In the ODPS Spark dialog box that appears, configure the node information.



5. After the ODPS Spark node is configured, publish and run the node.

Python

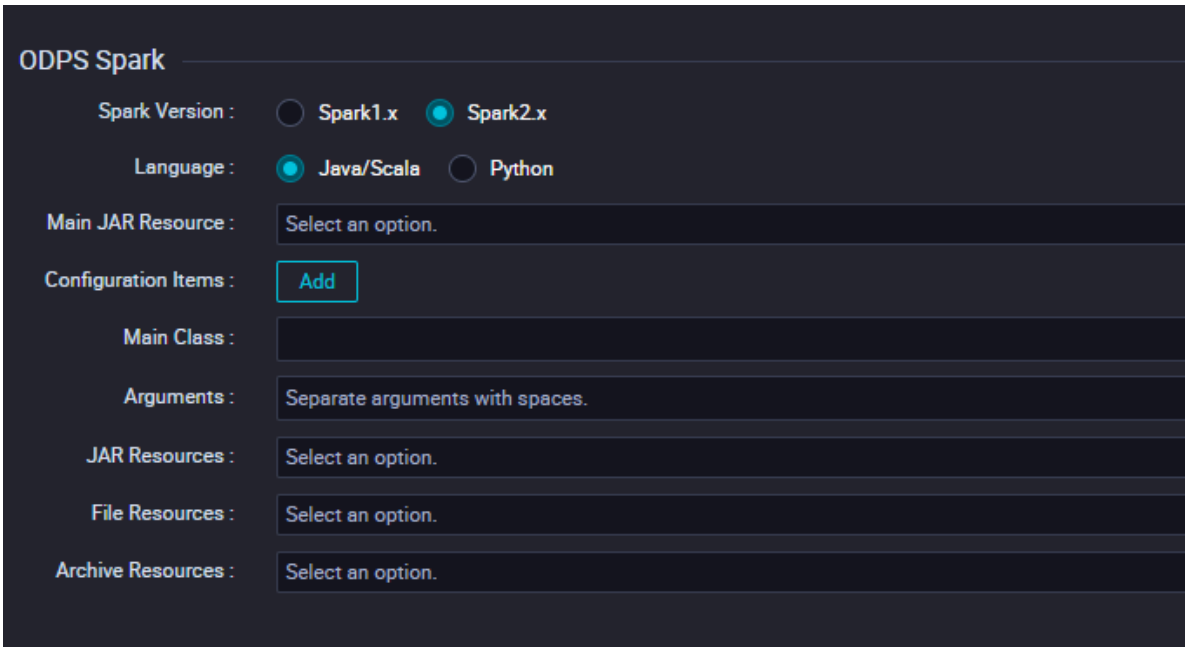
1. Prepare and upload the Python resource.
2. Create and configure an ODPS Spark node.



3. After the ODPS Spark node is configured, publish and run the node.

Lenet (BigDL)

1. Upload the JAR package and data (the mnist.zip file of the archive resource type).
2. Create and configure an ODPS Spark node.



3. After the ODPS Spark node is configured, publish and run the node.

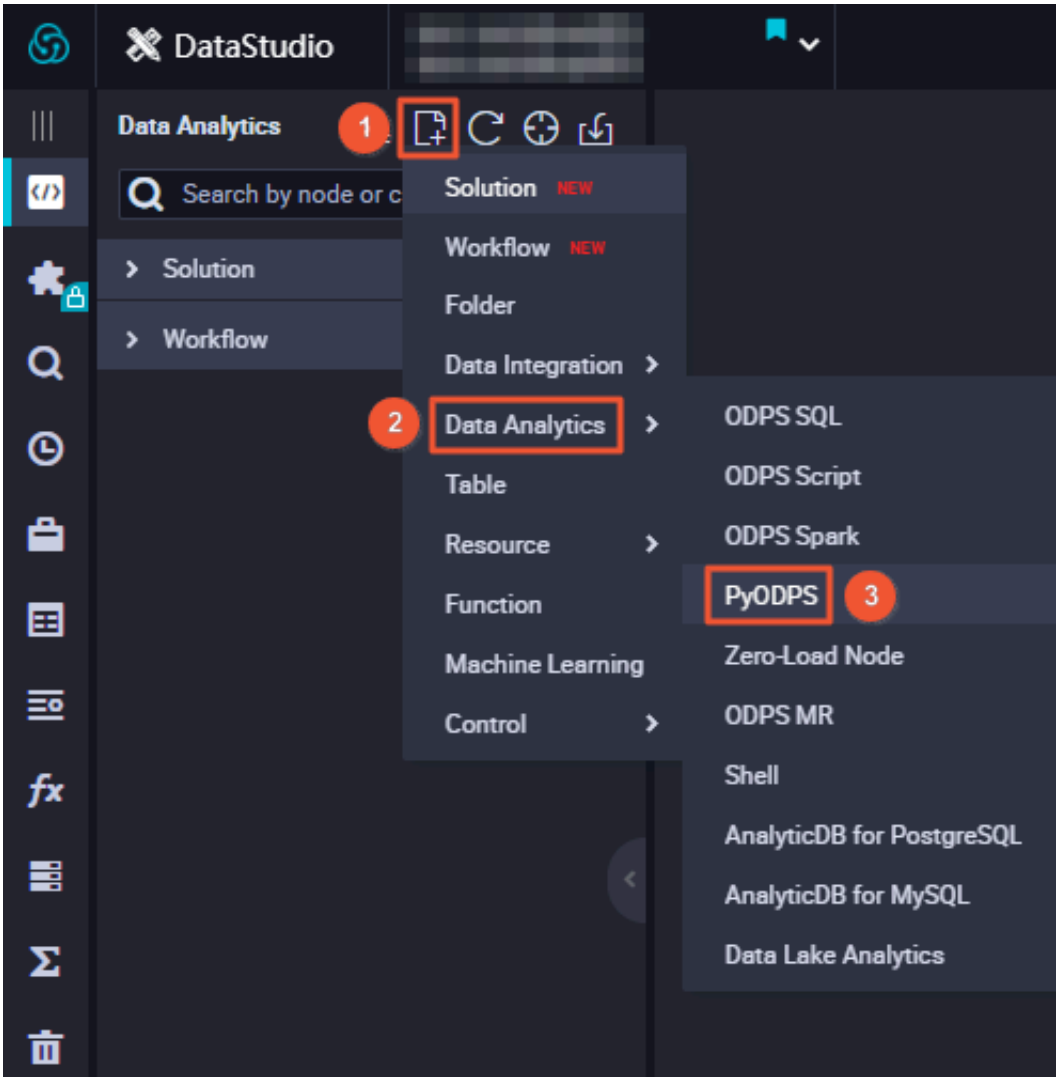
1.5.7 Virtual node

A virtual node is a control node that does not generate any data. Generally, it is used as the root node for the overall workflow node planning.

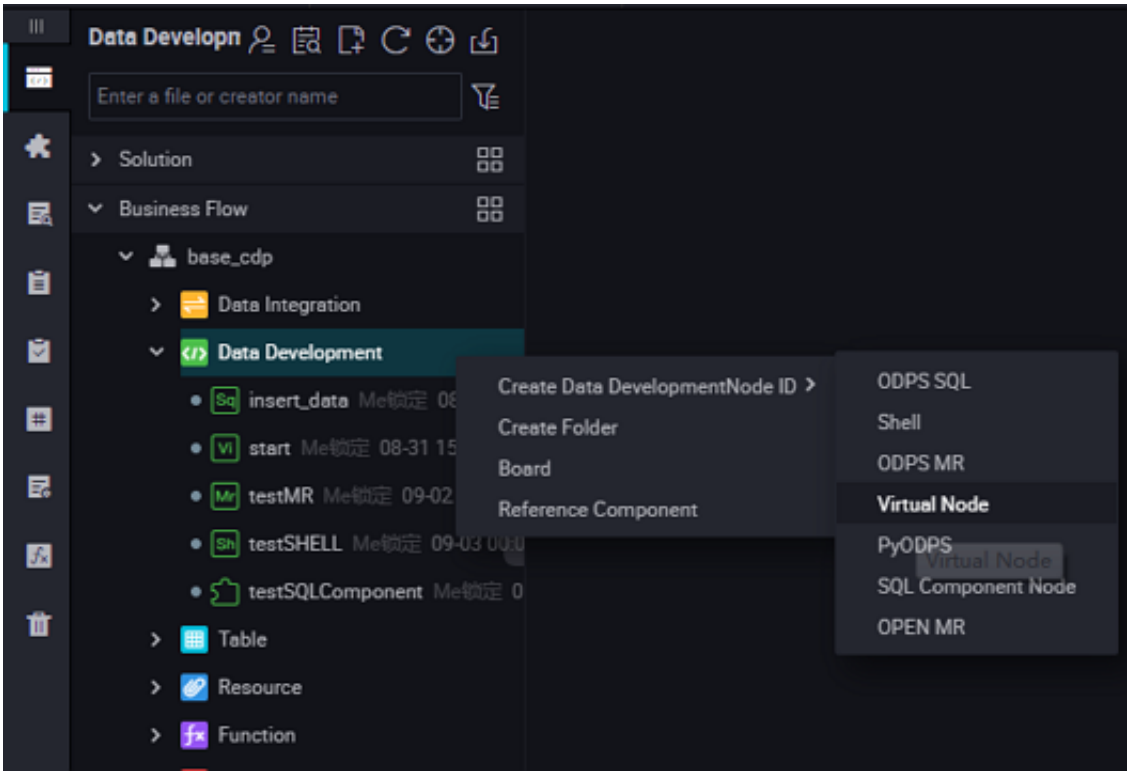
> Note:
> The final workflow output table contains multiple branch input tables. The virtual nodes are usually used if these input tables do not have any dependencies.

Create a virtual node task

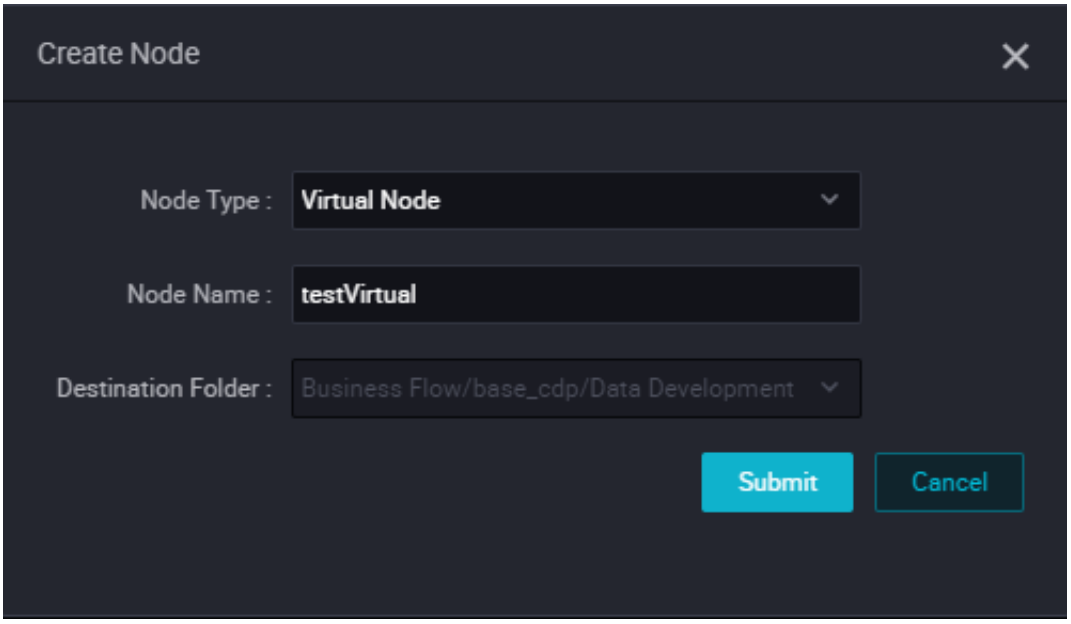
1. Right-click Business Flow under Data Development, and select Create Business Flow.



2. Right-click Data Development, and select Create Data Development Node > Virtual Node.



3. Set the node type to Virtual Node, and enter the node name. Select the target folder, and click Submit.



4. Edit the node code: You do not need to edit the virtual node code.

5. Node scheduling configuration.

   Click the Schedule on the right-side of the node task editing area to go to the Node Scheduling Configuration page. For more information about scheduling configuration, see Scheduling configuration.

6. Submit the node.

   After completing the configuration, click Save in the upper-left corner of the page or press Ctrl+S to submit (and unlock) the node to the development environment.

7. Publish a node task.

   For more information about the operation, see Publishmanagement.

8. Test in the production environment.

   For more information about the operation, see #unique_32.

1.5.8 ODPS MR node

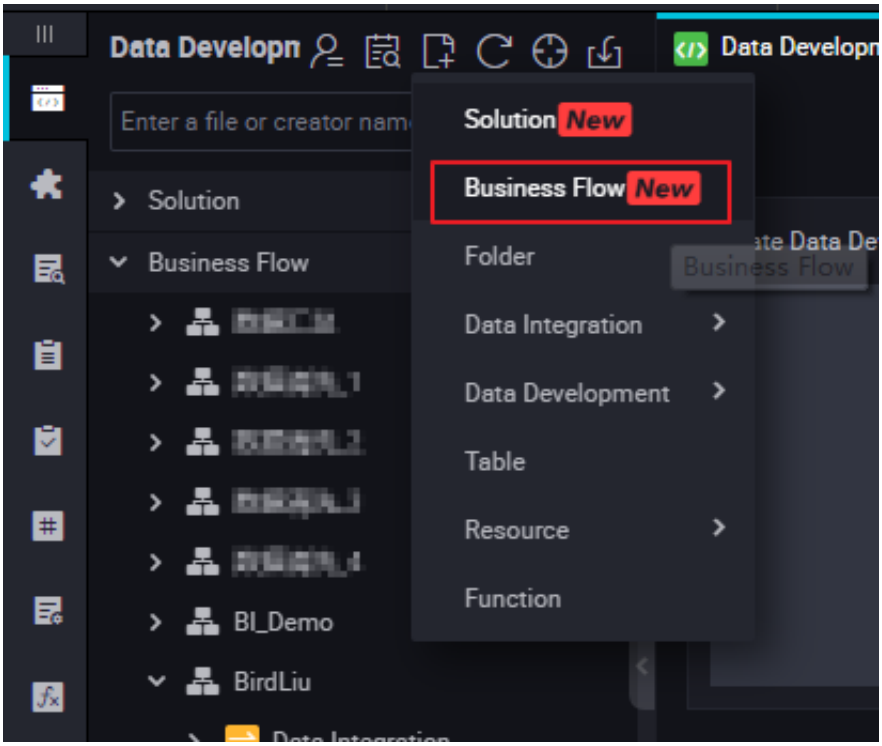
This topic describes the ODPS MR node functions. The MaxCompute supports MapReduce programming APIs. You can use the Java API provided by MapReduce to write MapReduce programs for processing data in MaxCompute. You can create ODPS MR nodes and use them in Task Scheduling.

For more information about how to edit and use the ODPS MR, see the examples in the MaxCompute documentation WordCount examples.

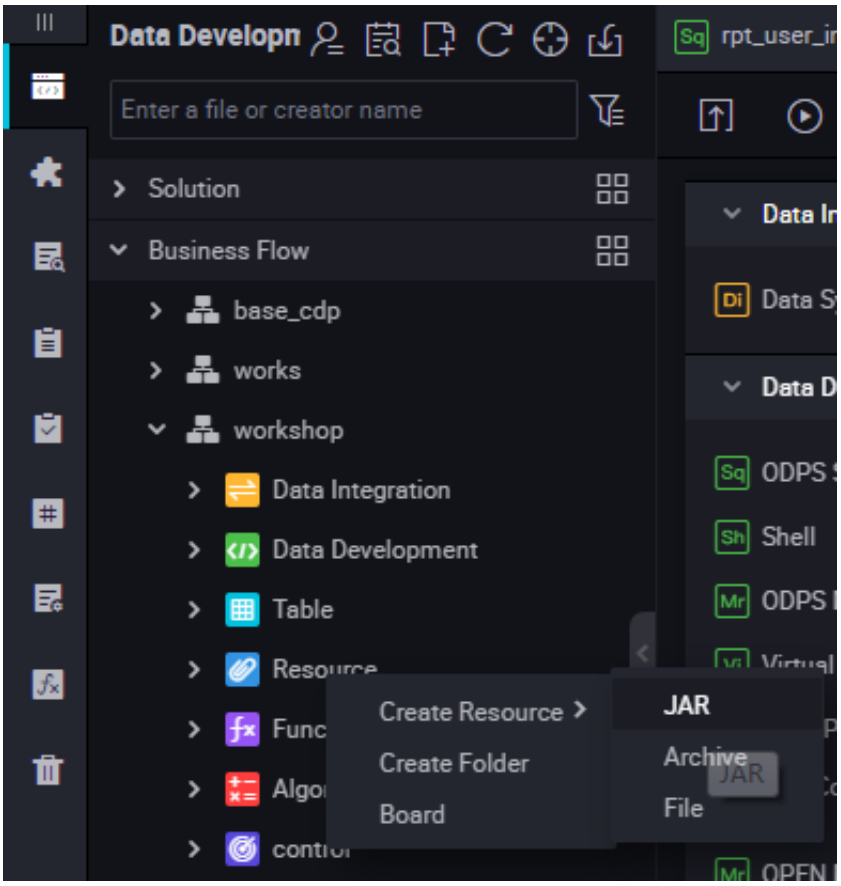
To use an ODPS MR node, you must upload and release the resource for usage, and then create the ODPS MR node.

Create a resource instance

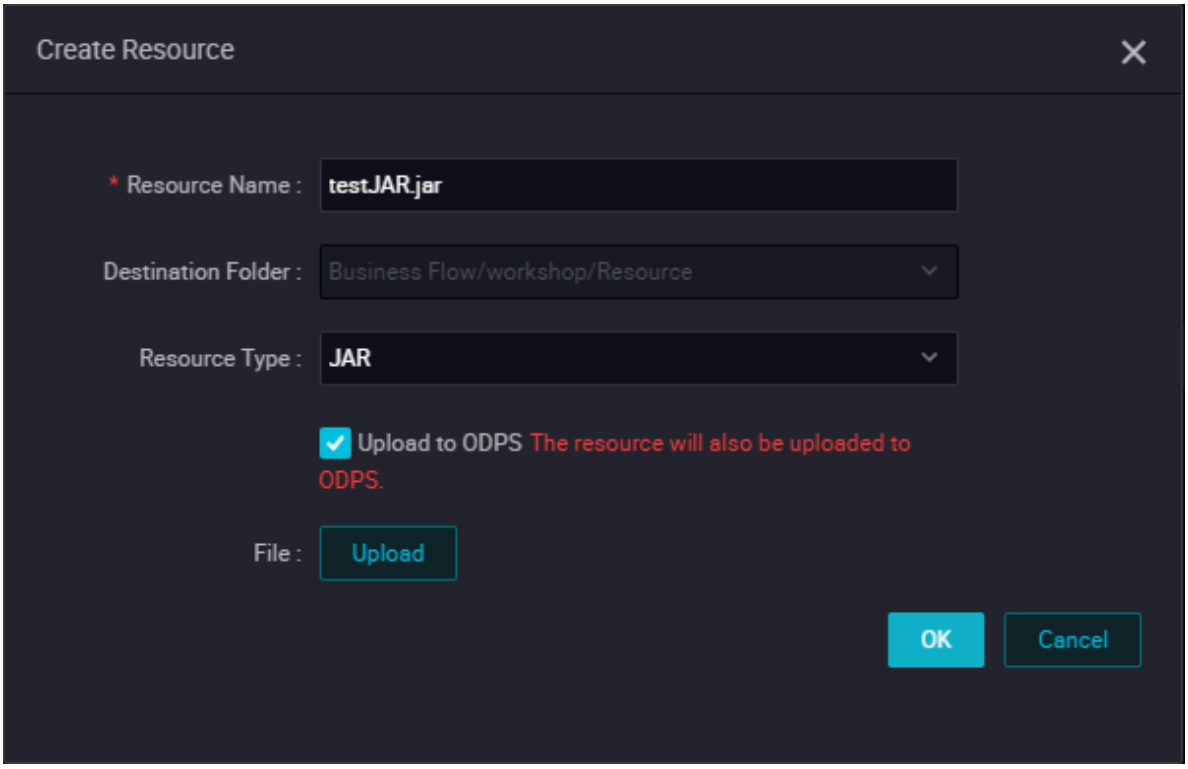
1. Right-click Business Flow under Data Development, and select Create Business Flow.



2. Right-click Resource, and select Create Resource > JAR.



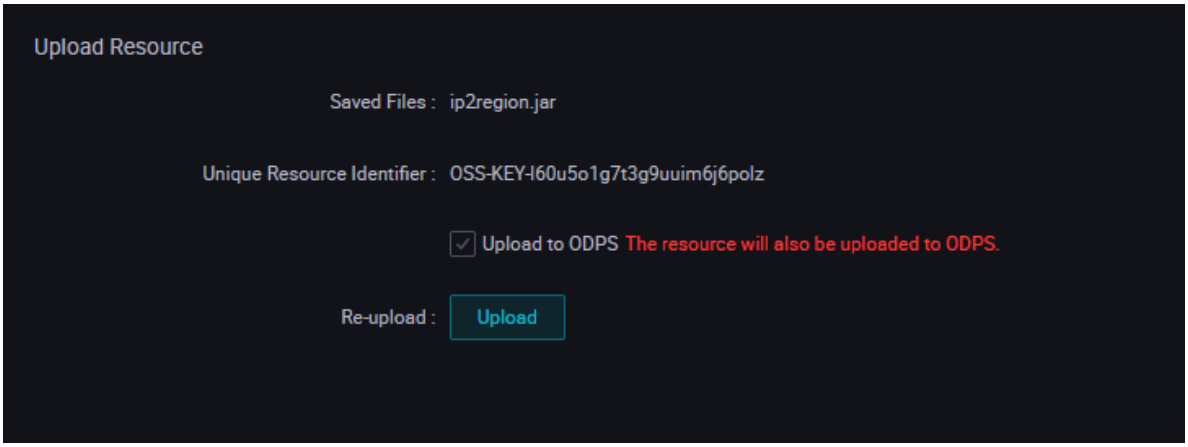
3. Enter the resource name in Create Resource according to the naming convention, and set the resource type to JAR, and then select a local JAR package.



Note:

- If this JAR package has been uploaded to the ODPS client, you must deselect Upload to ODPS. Otherwise, an error will be reported during the upload process.
- The resource name is not always the same as the uploaded file name.
- The resource name can be 1 to 128 characters in length, and include letters, numbers, underscores (_), and periods (.). It is case insensitive. The resource file extension is .jar if the resource is a JAR resource, and .py for a python resource.

4. Click Submit to submit the resource to the development scheduling server.

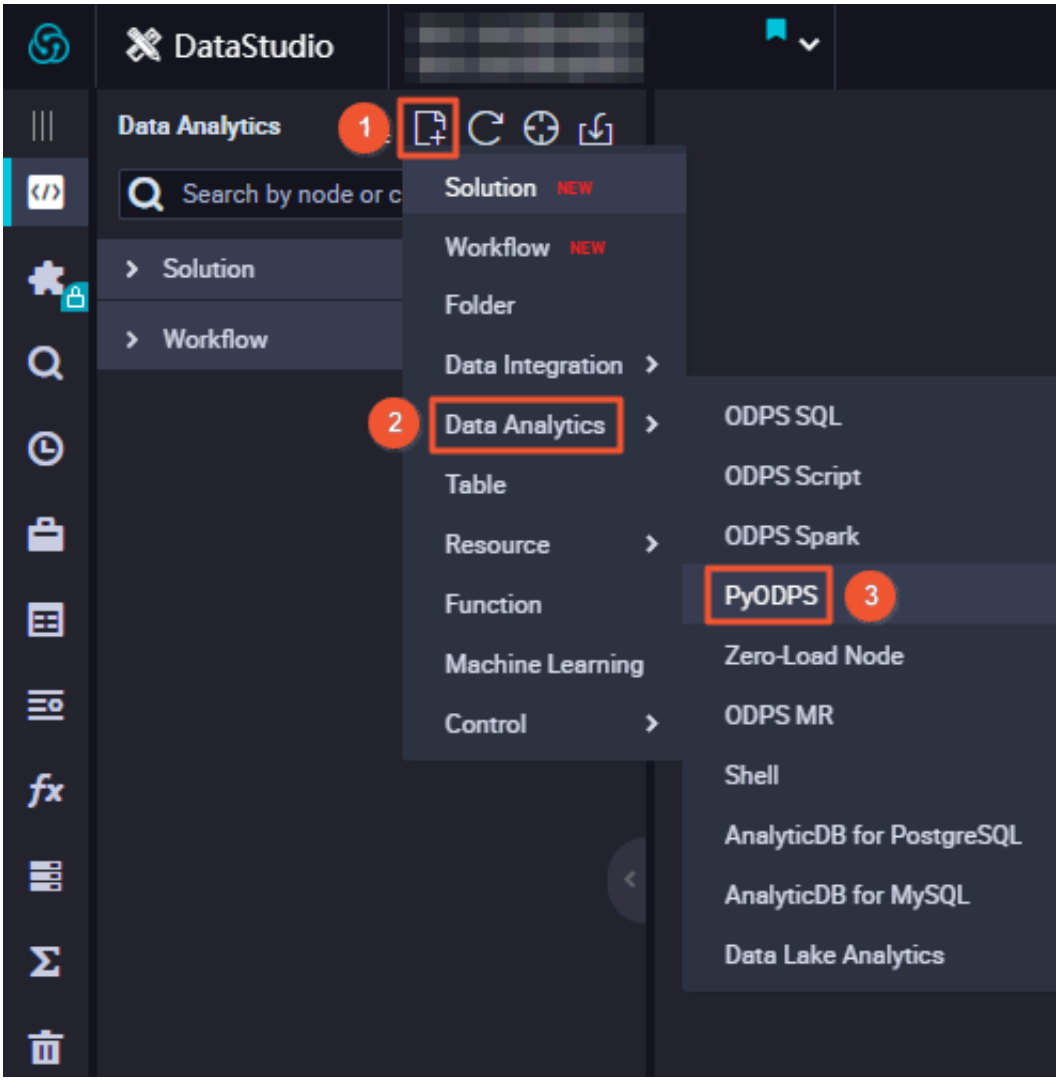


5. Publish a node task.

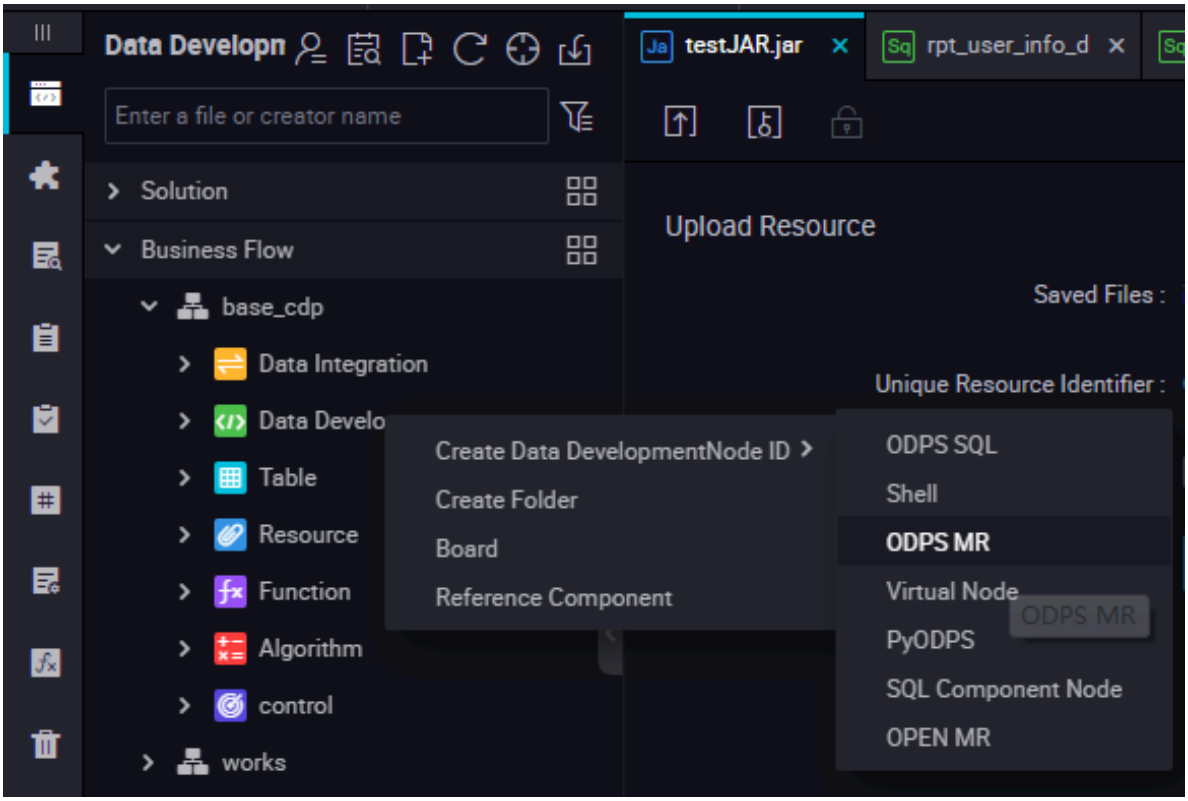
   For more information about the operation, see Release management.

Create an ODPS MR node

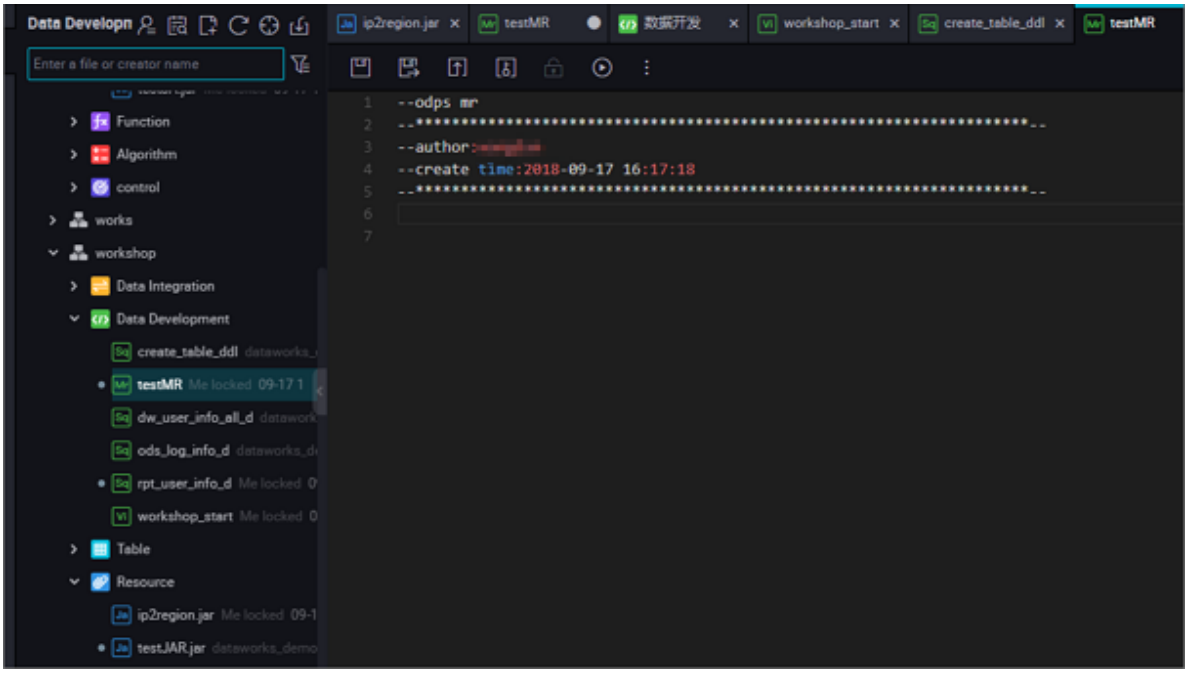
1. Right-click the Business Flow under Data Development, and select Create Business Flow.



2. Right-click Data Development, and select Create Data Development Node > ODPS MR.



3. Edit the node code. Double click the new ODPS MR node and enter the following interface:



The node code editing example as follows:

```
jar -resources base_test.jar -classpath ./base_test.jar
  com.taobao.edp.odps.brandnormalize.WordNormalizeW
ordAll
```

The code description as follows:

- The code `-resources base_test.jar` indicates the file name of the referenced JAR resource.
- The code `-classpath` is the JAR package path.
- The code `com.taobao.edp.odps.brandnormalize.WordNormalizeW ordAll` indicates the main class in the JAR package is called during execution. It must be consistent with the main class name in the JAR package.

When one MR calls multiple JAR resources, the classpath must be written as follows: `-classpath ./xxxx1.jar,./xxxx2.jar`, that is, two paths must be separated by a comma (,).

4. Node scheduling configuration.

   Click the Schedule on the right of the node task editing area to go to the Node Scheduling Configuration page. For more information about node scheduling configuration, see Scheduling configuration.

5. Submit the node.

   After completing the configuration, click Save in the upper-left corner of the page or press Ctrl+S to submit (and unlock) the node in the development environment.

6. Publish a node task.

   For more information about the operation, see Release management.

7. Test in the production environment.

   For more information about the operation, see #unique_32.

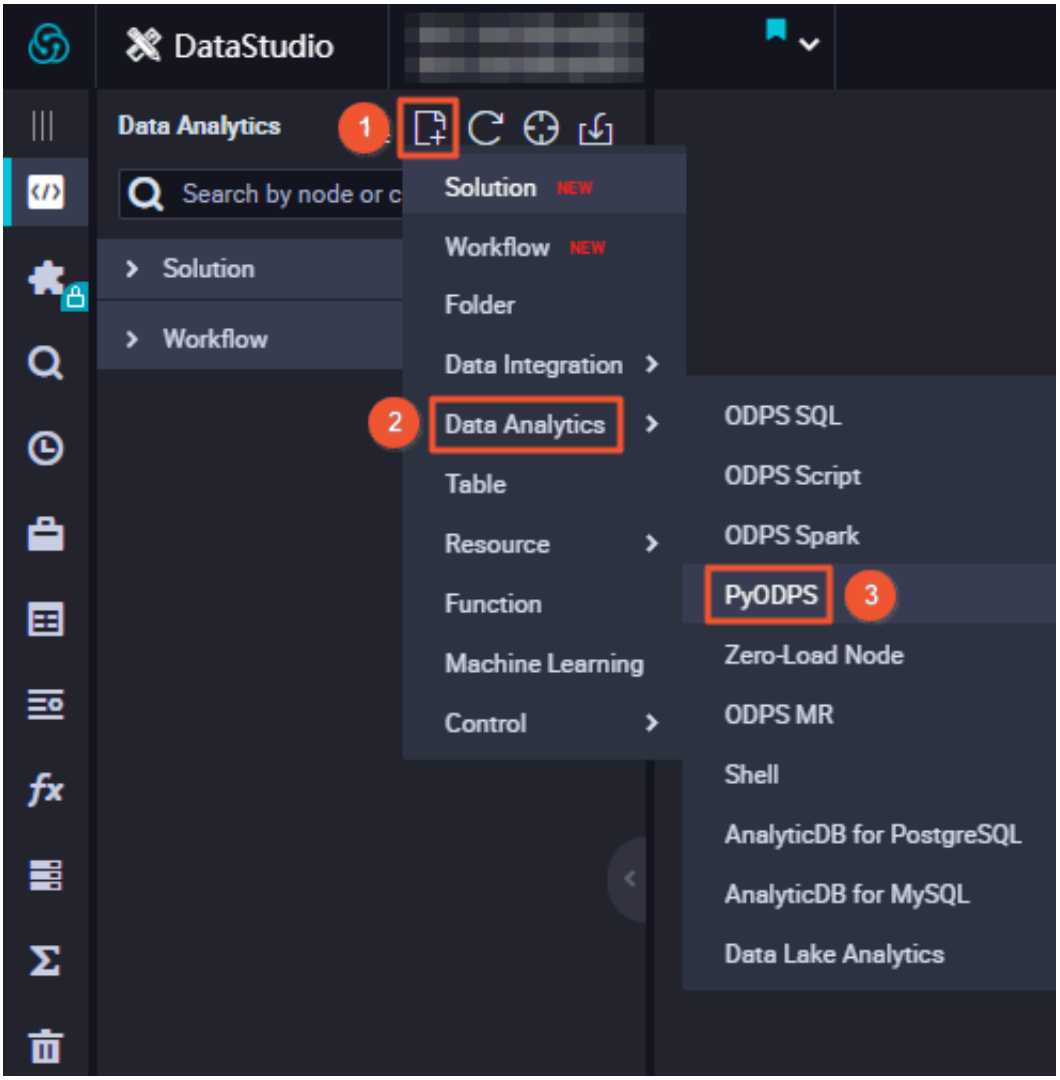
1.5.9 SHELL node

This topic describes the SHELL node. The SHELL node supports standard SHELL syntax but not the interactive syntax. The SHELL task can run on the default resource

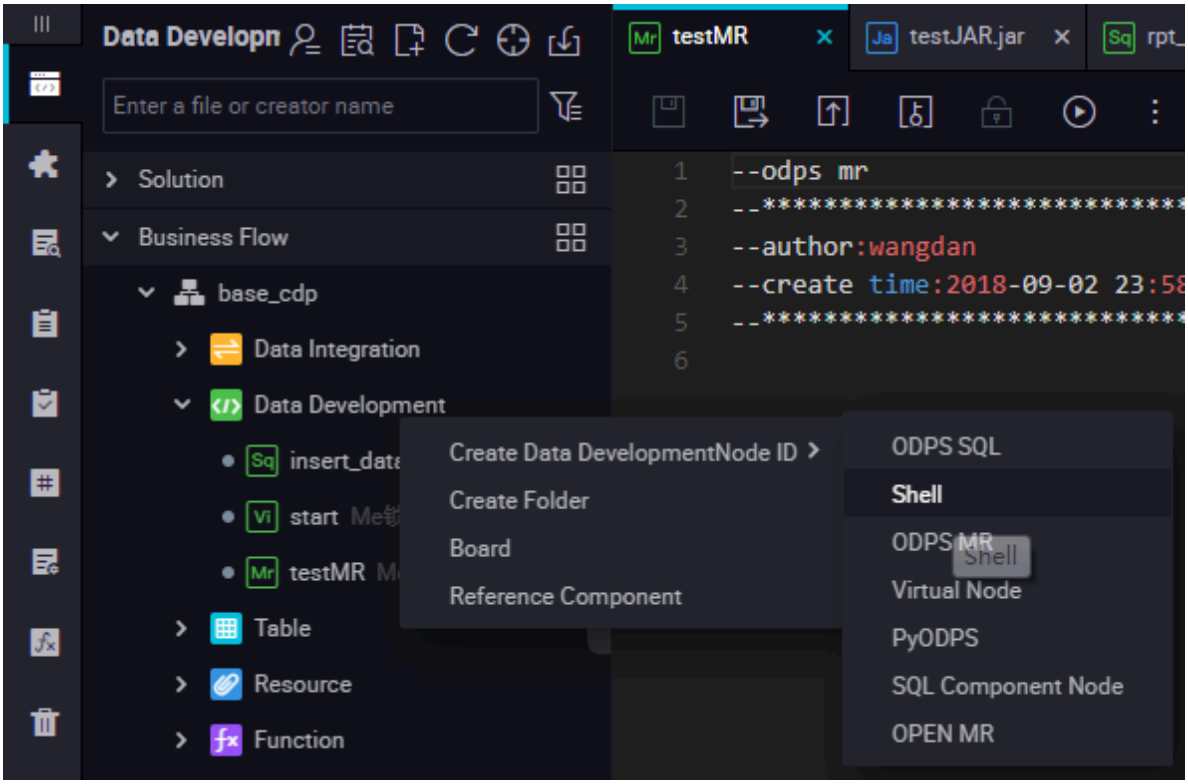
group. If you want to access an IP address or a domain name, add the IP address or domain name to the whitelist by choosing Project Configuration.

Procedure

1. Right-click Business Flow under Data Development, and select Create Business Flow.

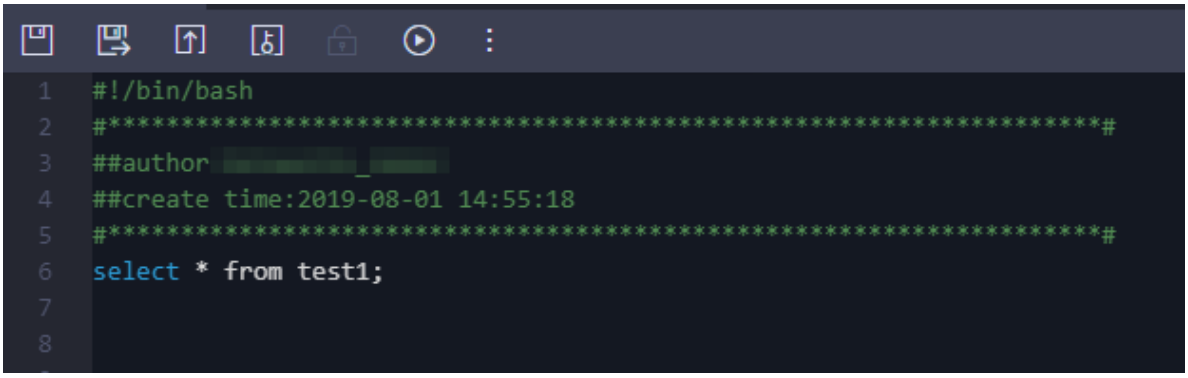


2. Right-click Data Development, and select Create Data Development Node > SHELL.



3. Set the node type to SHELL, and enter the node name. Select the target folder, and then click Submit.
4. Edit the node code.

   Go to the SHELL node code editing page and edit the code.



   If you want to call the System Scheduling Parameters in a SHELL statement, then compile the SHELL statement as follows:

```
echo  "$ 1  $ 2  $ 3 "
```

   Note:

Separate multiple parameters by spaces, for example: Parameter 1 Parameter 2... For more information about the usage of system scheduling parameters, see #unique_42.

5. Schedule node configuration.

   Click the Scheduling Configuration on the right of the node task editing area to go to the Node Scheduling Configuration page. For more information about Node Scheduling Configuration, see Scheduling configuration.

6. Submit the node.

   After completing the configuration, click Save in the upper-left corner of the page or press Ctrl+S to submit (and unlock) the node to the development environment.

7. Release a node task.

   For more information about the operation, see Release management.

8. Test the production environment.

   For more information about the production environment, see #unique_32.

Use cases

Connect to a database with SHELL

- If the database is built on Alibaba Cloud and the region is China (Shanghai), you must open the database with the following whitelisted IP addresses to connect to the database.

  10.152.69.0/24,10.153.136.0/24,10.143.32.0/24,120.27.160.26,10.46.67.156,120.27.160.81,10.46.64.81,121.43.110.160,10.117.39.238,121.43.112.137,10.117.28.203,118.178.84.74,10.27.63.41,118.178.56.228,10.27.63.60,118.178.59.233,10.27.63.38,118.178.142.154,10.27.63.15,100.64.0.0/8

  Note:
  If the database is built on Alibaba Cloud, but the region is not China (Shanghai). We recommend that you use the Internet or buy an ECS instance in the same database region, as the scheduling resource to run the SHELL task on a custom resource group.

- If the database is built locally, we recommend that you use the Internet connection and open the database in the preceding whitelisted IP addresses.

  Note:
  If you are using a Custom Resource Group to run the SHELL task, you must add the IP addresses of machines in the Custom Resource Group to the preceding whitelist.

1.5.10 PyODPS node

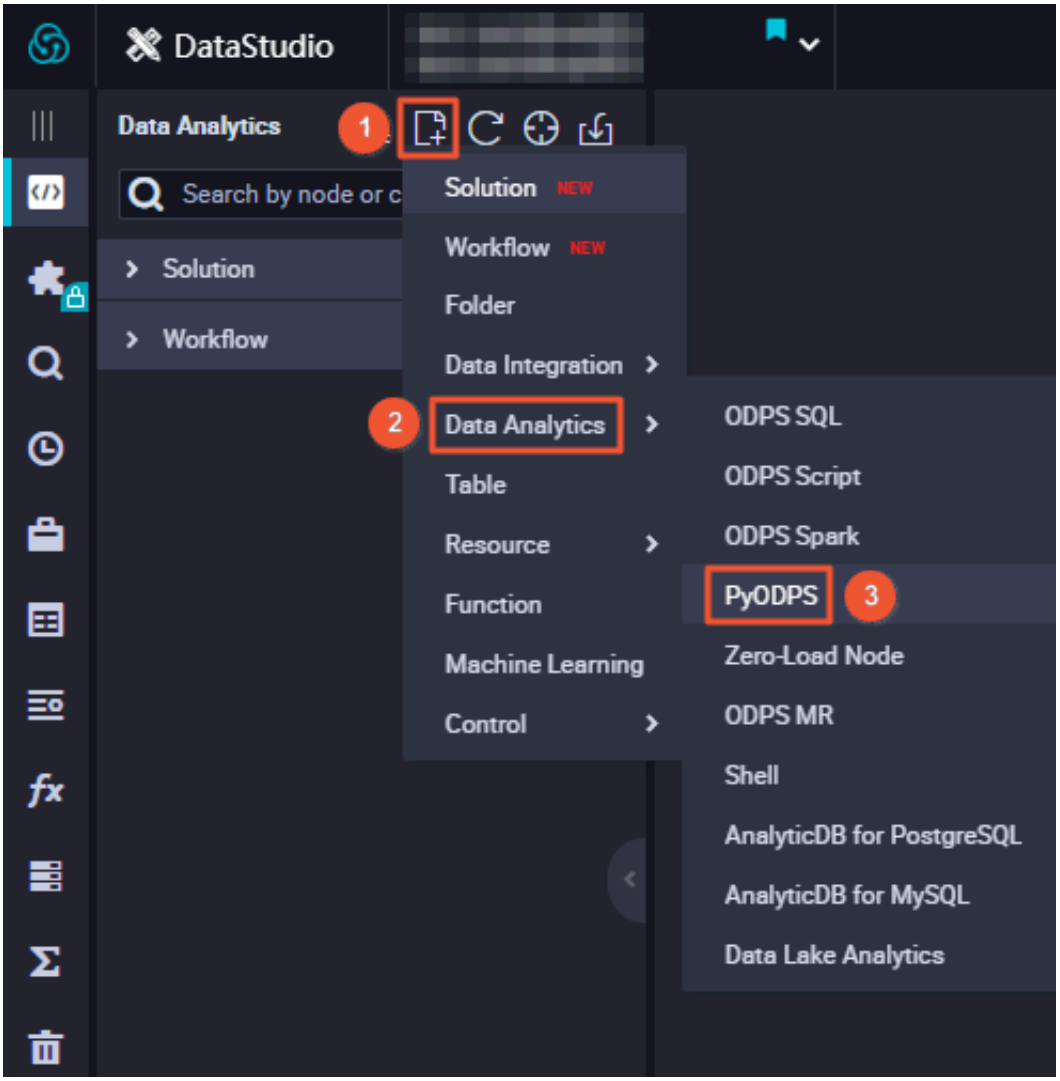
This topic describes the PyODPS node functions. The PyODPS node type in DataWorks can be integrated with the Python SDK of MaxCompute. You can edit the Python code to operate MaxCompute on a PyODPS node of DataWorks.

The Python SDK provided in MaxCompute can be used to operate MaxCompute.

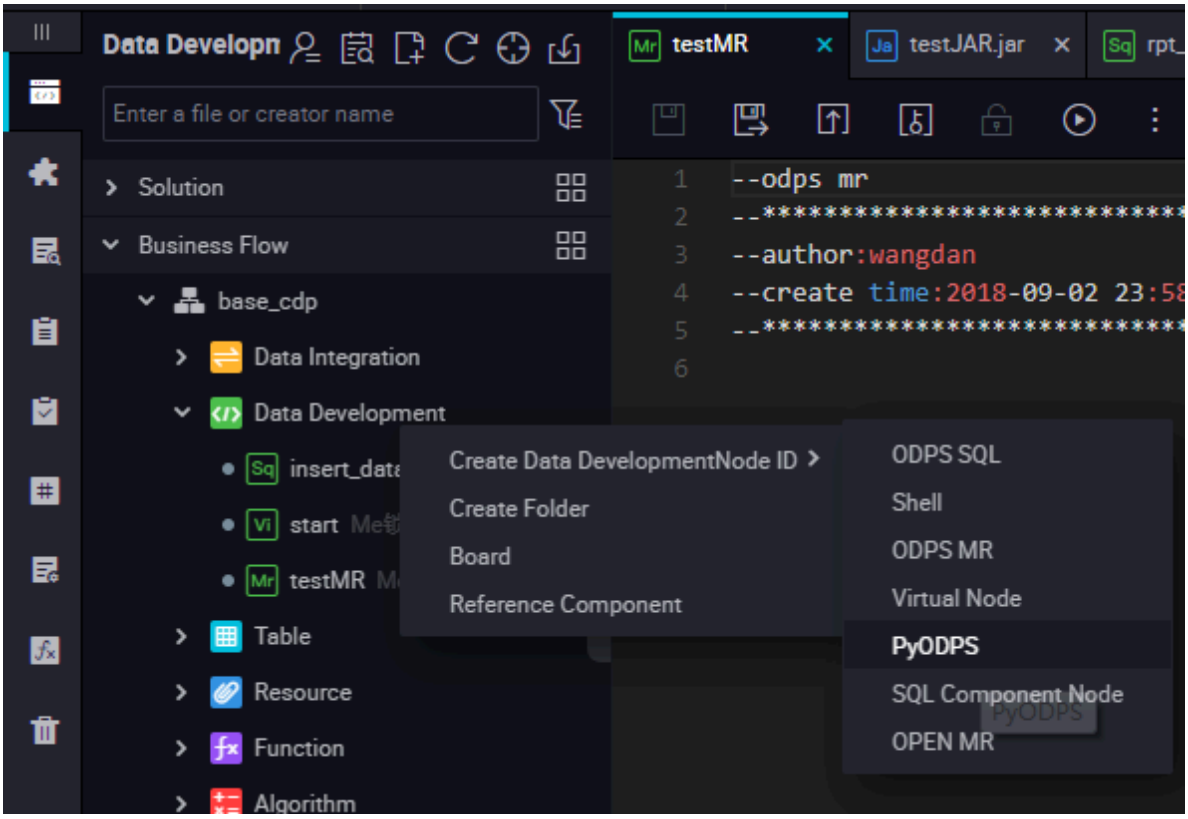
Note:
The Python 2.7 is used in the underlying layer. The data size of the PyODPS node process cannot exceed 50 MB, while the memory occupied cannot exceed 1 GB.

Create a PyODPS node

1. Right-click the Business Flow under Data Development, and select Create Business Flow.



2. Right-click Data Development, and select Create Data Development Node > PyODPS.



3. Edit the PyODPS node.

   a. MaxCompute portal

      On DataWorks, the PyODPS node contains a global variable odps or o, which is the MaxCompute entry. You do not need to manually define a MaxCompute entry.

```
print ( odps . exist_tabl  e (' PyODPS_iri  s '))
```

   b. Run the SQL statements

      PyODPS supports MaxCompute SQL query and can read the execution result. The return value of the execute_sql or run_sql method is the running instance.

      Note:
      Not all commands that can be executed on the MaxCompute console are SQL statements accepted by MaxCompute. You need to use other methods to call non-DDL/DML statements. For example, use the run_security_query method to

call the GRANT or REVOKE statements, and use the run_xflow or execute_xflow method to call PAI commands.

```
o . execute_sq  l (' select  *  from   dual ') #  Run   the
  SQL   statements   in   synchronou  s   mode .  Blocking
continues   until   execution   of   the   SQL   statement   is
  completed .
instance  =  o . runsql (' select  *  from   dual ') #  Run
the   SQL   statements   in   asynchrono  us   mode .
print ( instance . getlogview  _address ()) #  Obtain   the
logview   address .
instance . waitforsuc  cess () #  Blocking   continues   until
execution   of   the   SQL   statement   is   completed .
```

c. Configure the runtime parameters

The runtime parameters must be set sometimes. You can set the hints parameter with the dict parameter type.

```
o . execute_sq  l (' select  *  from   PyODPS_iri  s ',  hints
={' odps . sql . mapper . split . size ':  16 })
```

After you add sql.settings to the global configuration, the related runtime parameters are added upon each running.python.

```
from   odps   import   options
options . sql . settings  = {' odps . sql . mapper . split . size
':  16 }
o . execute_sq  l (' select  *  from   PyODPS_iri  s ')  # "
hints "  is   added   based   on   the   global   configurat  ion
.
```

d. Read the SQL statement execution results

The instance that runs the SQL statement can perform the open_reader operation. In this case, the structured data is returned as the SQL statement execution result.

```
with   o . execute_sq  l (' select  *  from   dual ').
open_reade  r ()  as   reader :
for   record   in   reader :  #  Process   each   record .
```

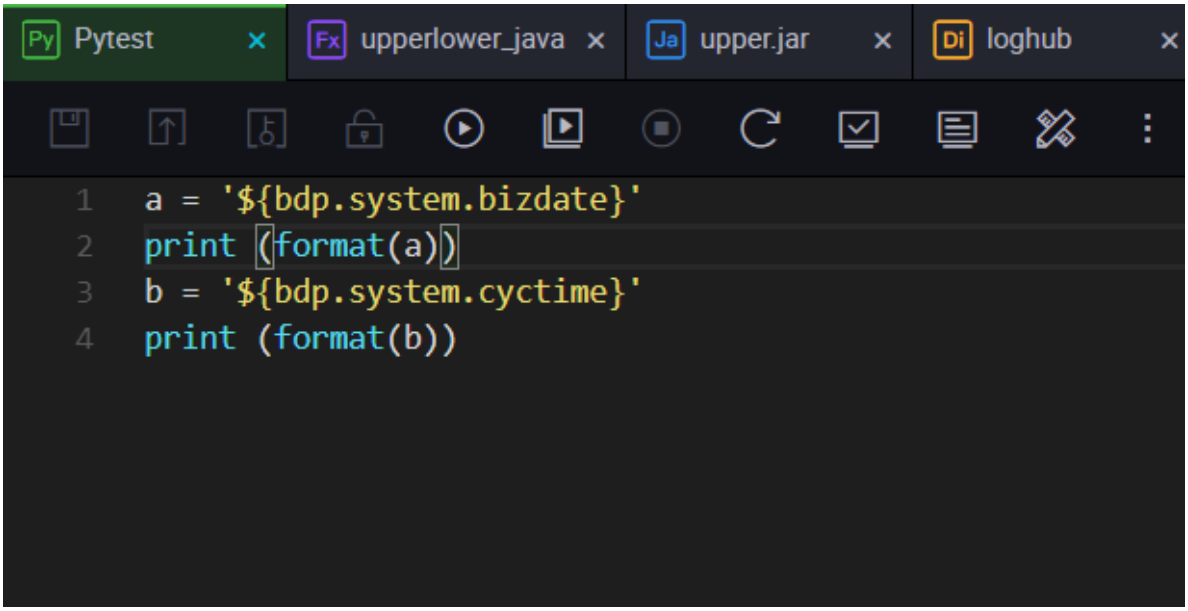
In another case, desc may be executed in an SQL statement. In this case, the original SQL statement execution result is obtained through the reader.raw attribute.

```
with   o . execute_sq  l (' desc   dual '). open_reade  r ()  as
  reader :
print ( reader . raw )
```

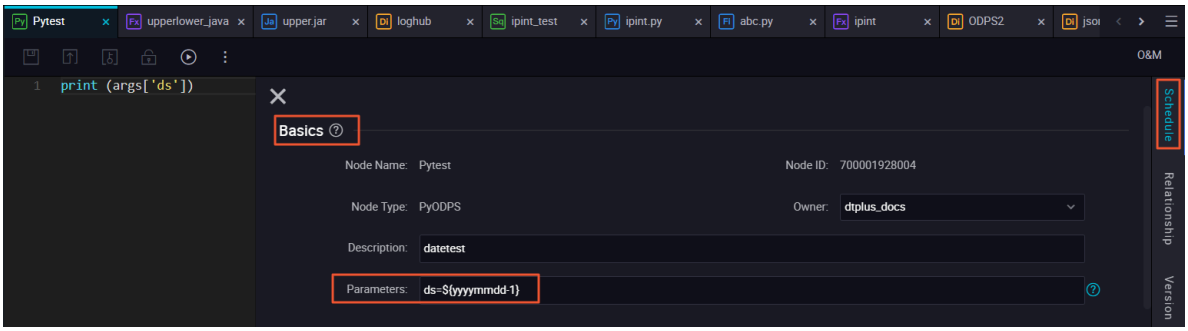
 Note:

The user-defined scheduling parameters are used in data development. If a PyODPS node is triggered on the page, and the time must be specified. The PyODPS node time cannot be directly replaced by an SQL node.

You can configure system parameters as following:



You can configure user-defined parameters as following.



4. Node scheduling configuration.

   Click the Schedule on the right of the node task editing area to go to the Node Scheduling Configuration page. For more information, see Scheduling configuration.

5. Submit the node.

   After completing the configuration, click Save in the upper-left corner of the page or press Ctrl+S to submit (and unlock) the node in the development environment.

6. Publish a node task.

   For more information about the operation, see Release management.

7. Test in the production environment.

   For more information about the operation, see #unique_32.

1.5.11 for-each node

This topic describes how to use a for-each node to repeat a loop twice and display the loop count.

Create a workflow

1. On the DataStudio page, click Data Analytics in the left-side navigation pane. Move the pointer over the Create icon and choose Control > for-each.
2. In the Create Node dialog box that appears, set the parameters and click Commit.
3. In the created workflow, create an assignment node as the parent node of the for-each node.

   The assignment node is a SHELL node. The sample code for the node is as follows:

   ```
   echo  ' this   is   name , ok ';
   ```

   The outputs parameter is the default output parameter of the assignment node.

Edit the for-each node

Note:

- The start and end nodes of the for-each node have fixed logic and cannot be edited .
- After modifying the code for the SHELL node of the for-each node, save the modification. You are not prompted to save the modification when submitting the node. If you do not save the modification, the latest code cannot be updated in time.

The code for the SHELL node is as follows:

```
echo  ${ dag . loopTimes } ---- Displays   the   loop   count .
```

A for-each node supports the following environment variables:

- ${dag.foreach.current}: the current data row.
- ${dag.loopDataArray}: the input dataset.
- ${dag.offset}: the offset of the loop count to 1.

- ${dag.loopTimes}: the loop count, whose value equals to the value of ${dag.offset} plus 1.

```
// Compare the code of the SHELL node with that of
   a common for loop .
 data =[] // It is equivalent to ${ dag . loopDataAr ray }.
// i is equivalent to ${ dag . offset }.
 for ( int i = 0 ; i < data . length ; i ++) {
   print ( data [ i ]); // data [ i ] is equivalent to ${ dag
 . foreach . current }.
}
```

The ${dag.loopDataArray} parameter is the default input parameter of the for-each node. Set this parameter to the value of the outputs parameter of the parent node. If you do not set this parameter, an error occurs when you submit the node.

Click the Submit icon. On the O&M page that appears, check the running result.

1.5.12 do-while node

You can define mutually dependent nodes, including a loop decision node named "end", on a do-while node. DataWorks repeatedly runs the nodes and exits the loop only when the end node returns False.

 Note:
A loop can be repeated for a maximum of 128 times. If the loop count exceeds this limit, an error occurs.

The do-while node supports the MaxCompute SQL, SHELL, and Python languages. If you use MaxCompute SQL, you can use a case statement to evaluate whether the specified condition for exiting the loop is met. The following figure shows the sample code for the end node.

Simple example

This section describes how to use a do-while node to repeat a loop five times and display the loop count each time the loop runs.

1. On the DataStudio page, click Data Analytics in the left-side navigation pane. Move the pointer over the Create icon and choose Control > do-while.
2. In the Create Node dialog box that appears, set the parameters and click Commit.

3. Double-click the created do-while node and define the loop body.

   The do-while node consists of the start, sql, and end nodes.

   - The start node marks the startup of a loop and does not have any business effect.
   - DataWorks provides the sql node as a sample business processing node. You need to replace the sql node with your own business processing node, for example, a SHELL node named "Display loop count." The following figure shows the sample code for the SHELL node.
   - The end node marks the end of a loop and determines whether to start the loop again. In this example, it defines the condition for exiting the loop for the do-while node.

     The end node only assigns values True and False, indicating whether to start a loop again or exit the loop. The following figure shows the sample code for the end node.

     The ${dag.loopTimes} variable is used in both the "Display loop count" node and the end node. It is a reserved variable of the system. It indicates the loop count and increments from 1. The internal nodes of the do-while node can directly reference this variable.

     The value of the ${dag.loopTimes} variable is compared with 5 in the code, limiting the total number of times the loop runs. The value is 1 for the first run, 2 for the second run, and so on. When the loop runs for the fifth time, the value is 5. In this case, the conditional statement ${dag.loopTimes}<5 is False, and the do-while node exits the loop.

4. Run the do-while node.

   You can configure the scheduling settings for the do-while node as needed and submit it to O&M for running.

   - do-while node: The do-while node is displayed as a whole node in O&M. To view the loop details about the do-while node, right-click the node and select View Internal Nodes.
   - Internal loop body: This view is divided into three parts.
     - The left pane of the view lists the rerun history of the do-while node. A record is generated for each run of the whole do-while instance.
     - The middle pane of the view shows a loop record list. Each record correspond s to each run of the do-while node. The running status of the node for each run is also displayed.
     - The right pane of the view shows the details about the do-while node each time the loop runs. You can click a record in the loop record list to view the running status of the corresponding instance.
5. Check the running result.

   Access the internal loop body. In the loop record list, click the record correspond ing to the third run. The loop count is 3 in the run logs.

   You can also view the run logs of the end node that are generated when the loop runs for the third time and for fifth time, respectively.

   The conditional statement 3<5 is True when the loop runs for the third time, while the conditional statement 5<5 is False when the loop runs for the fifth time. Therefore, the do-while node exits the loop after the fifth run.

Based on the preceding simple example, the do-while node works in the following process:

1. Run from the start node.
2. Run nodes in sequence based on the defined node dependencies.
3. Define the condition for exiting a loop for the end node.
4. Run the conditional statement of the end node after the loop ends for the first time.
5. Record the loop count as 1 and start the loop again if the conditional statement returns True in the run logs of the end node.

6. Exit the loop if the conditional statement returns False in the run logs of the end node.

Complex example

Besides the preceding simple scenarios, do-while nodes can also be used in complex scenarios where each row of data is processed in sequence by using a loop. Before processing data in such scenarios, make sure that:

- You have deployed a parent node that can export queried data to the do-while node. You can use an assignment node to meet this condition.
- The do-while node can obtain the output of the parent node. You can configure the context and dependencies to meet this condition.
- The internal nodes of the do-while node can reference each row of data. In this example, the existing node context is enhanced and the system variable ${dag.offset} is assigned to help you reference the context of the do-while node.

This section describes how to use the do-while node to respectively display records 0 and 1 in two rows of the tb_dataset table each time the loop runs.

1. On the DataStudio page, click Data Analytics in the left-side navigation pane. Move the pointer over the Create icon and choose Control > do-while.
2. In the Create Node dialog box that appears, set the parameters and click Commit.
3. Double-click the created do-while node and define the loop body.
   a. Create a parent node named "Initialize dataset" for the do-while node. The parent node generates a test dataset.
   b. Click Schedule in the upper-right corner to configure a dedicated context for the do-while node. Set Parameter Name to input and Value Source to the output of the parent node.
   c. Type the code for the business processing node named "Print each data row."
      - `${ dag . offset }`: a reserved variable of DataWorks. This variable indicates the offset of the loop count to 1. The offset is 0 for the first run, 1 for the second run, and so on. The offset equals to the loop count minus 1.
      - `${ dag . input }`: the context that you configure for the do-while node. As mentioned above, the do-while node is configured with the input parameter,

with Value Source set to the output of the parent node named "Initialize dataset."

The internal nodes of the do-while node can directly use ${dag.${ctxKey}} to reference the context. In this example, ${ctxKey} is set to input. Therefore, you can use ${dag.input} to reference the context.

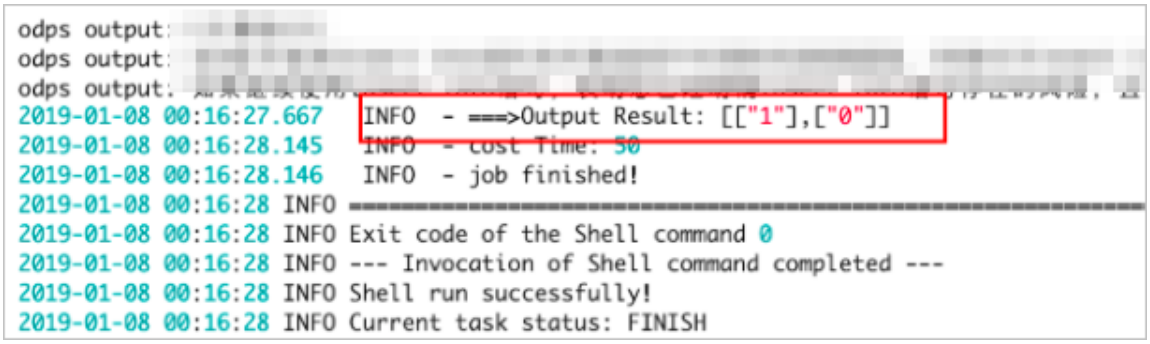
- `${ dag . input [${ dag . offset }]}`: The node "Initialize dataset" exports a table. DataWorks can obtain a row of data in the table based on the specified offset. The value of ${dag.offset} increments from 0. Therefore, the displayed results are ${dag.input[0]}, ${dag.input[1]}, and so on until all data in the dataset is displayed.

d. Define the condition for exiting the loop for the end node. As shown in the following figure, the values of ${dag.loopTimes} and ${dag.input.length} are compared. If the value of ${dag.loopTimes} is smaller than that of ${dag.input.length}, the end node returns True and the do-while node continues the loop. Otherwise, the end node returns False and the do-while node exits the loop.

 Note:
The system automatically sets the ${dag.input.length} variable to the number of rows in the array specified by the input parameter based on the context configured for the do-while node.

4. Run the nodes and view the running result.

   · The node "Initialize dataset" generates data rows 0 and 1.



   · The following figures show the running result of the node "Print each data row."

Figure 1-1: Display the first row of data

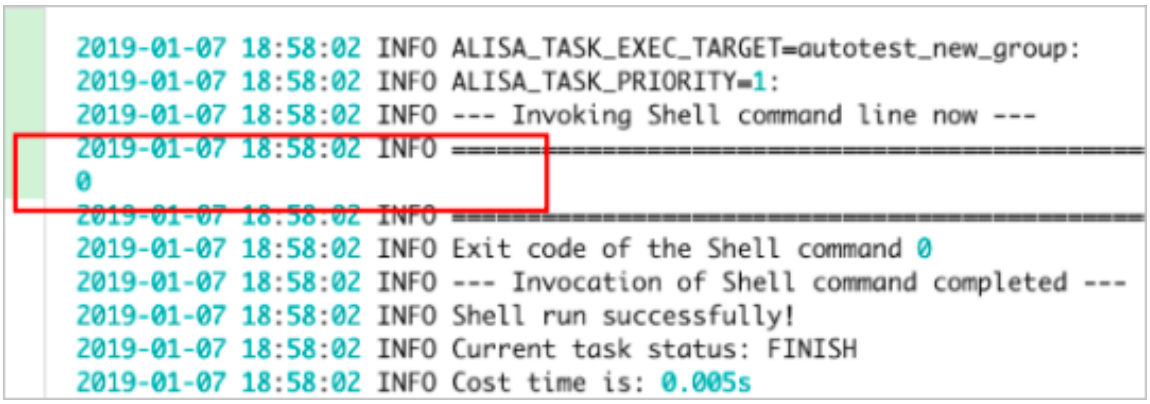
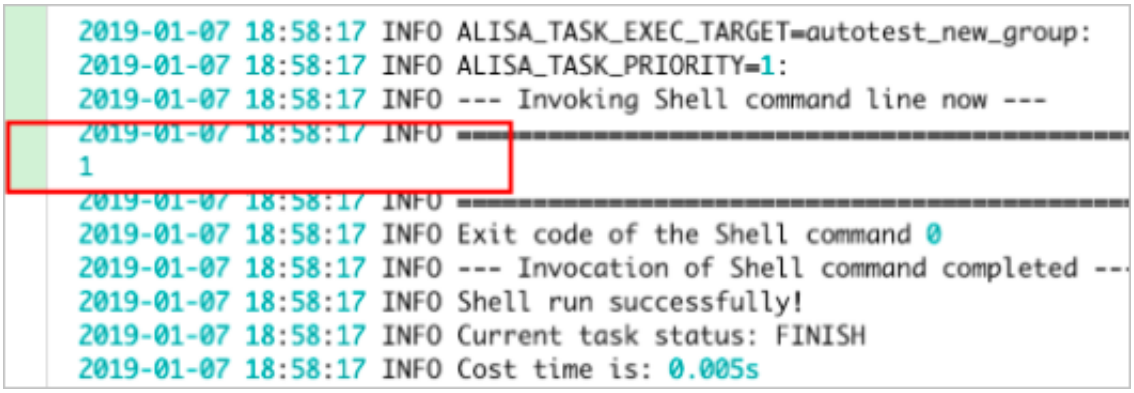


Figure 1-2: Display the second row of data



- The following figures show the running result of the end node.

  Figure 1-3: Run logs generated when the loop runs for the first time

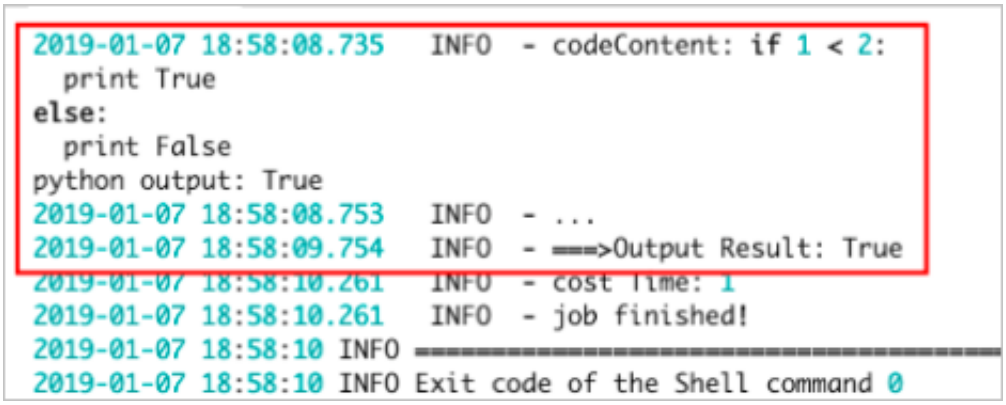
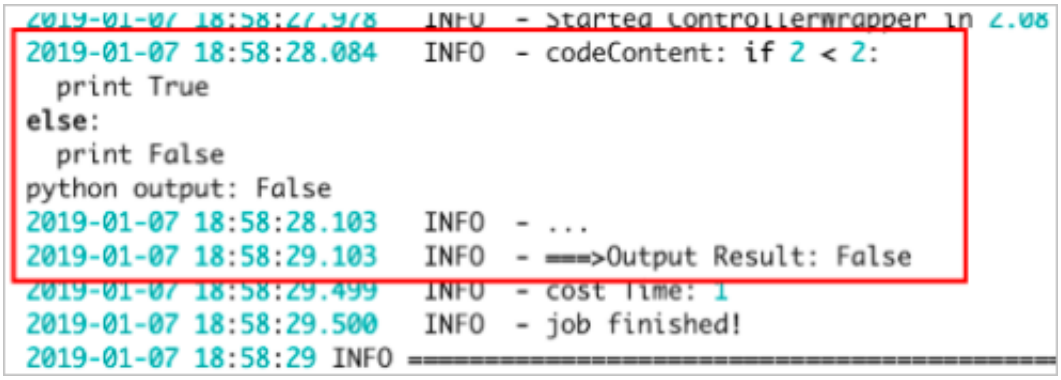


  Figure 1-4: Run logs generated when the loop runs for the second time



  As shown in the preceding figures, the loop count is smaller than the number of the rows when the loop runs for the first time. Therefore, the end node returns True and the loop continues. The loop count equals to the number of the rows when the loop runs for the second time. Therefore, the end node returns False and the loop stops.

Summary

- Compared with the while, foreach, and do...while statements, a do-while node:
  - Contains a loop body that runs a loop before evaluating the conditional statement, providing the same function as the do...while statement. A do-while node can also use the system variable ${dag.offset} and the node context to implement the function of the foreach statement.
  - Cannot achieve the function of the while statement because a do-while node runs a loop before evaluating the conditional statement.

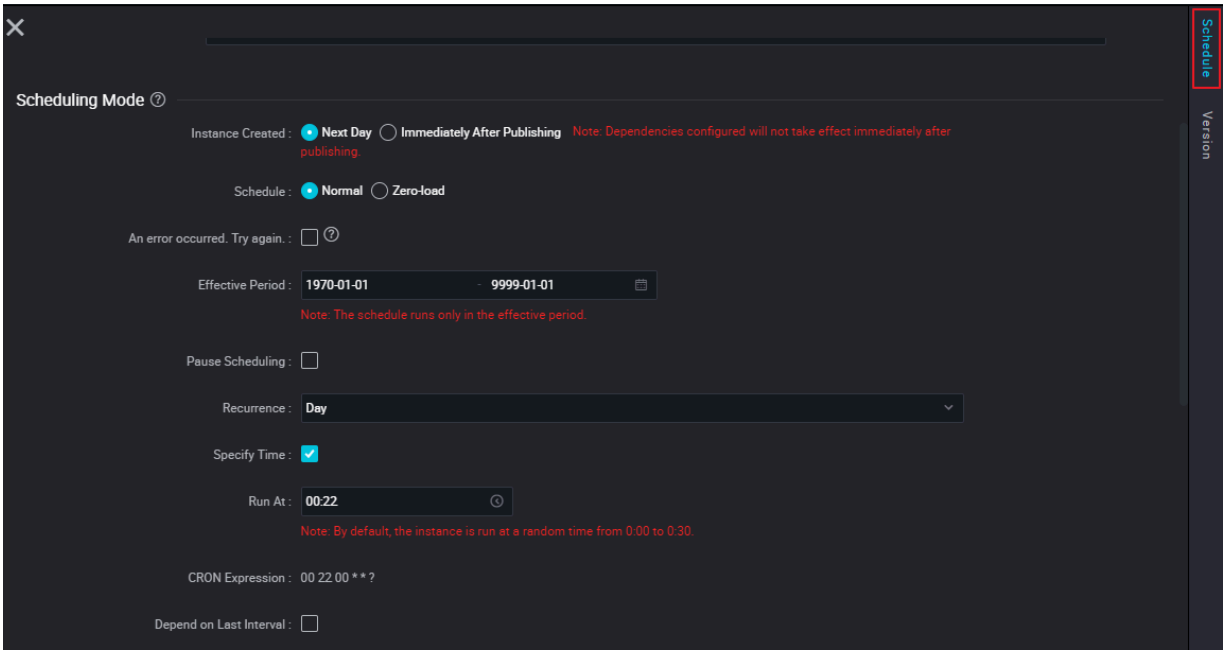
- The do-while node works in the following process:
  1. Run nodes in the loop body starting from the start node based on node dependencies.
  2. Run the code defined for the end node.
     - Run the loop again if the end node returns True.
     - Stop the loop if the end node returns False.
- Method to use the context: The internal nodes of the do-while node can use ${dag.${ctxKey}} to reference the context defined for the do-while node.
- System parameters: DataWorks automatically issues the following system variables for the internal nodes of the do-while node:
  - ${dag.loopTimes}: the loop count, starting from 1.
  - ${dag.offset}: the offset of the loop count to 1, starting from 0.

1.5.13 Cross-tenant nodes

This topic describes cross-tenant nodes that are typically used to associate nodes from different tenants. The cross-tenant nodes are divided into sender and recipient nodes.

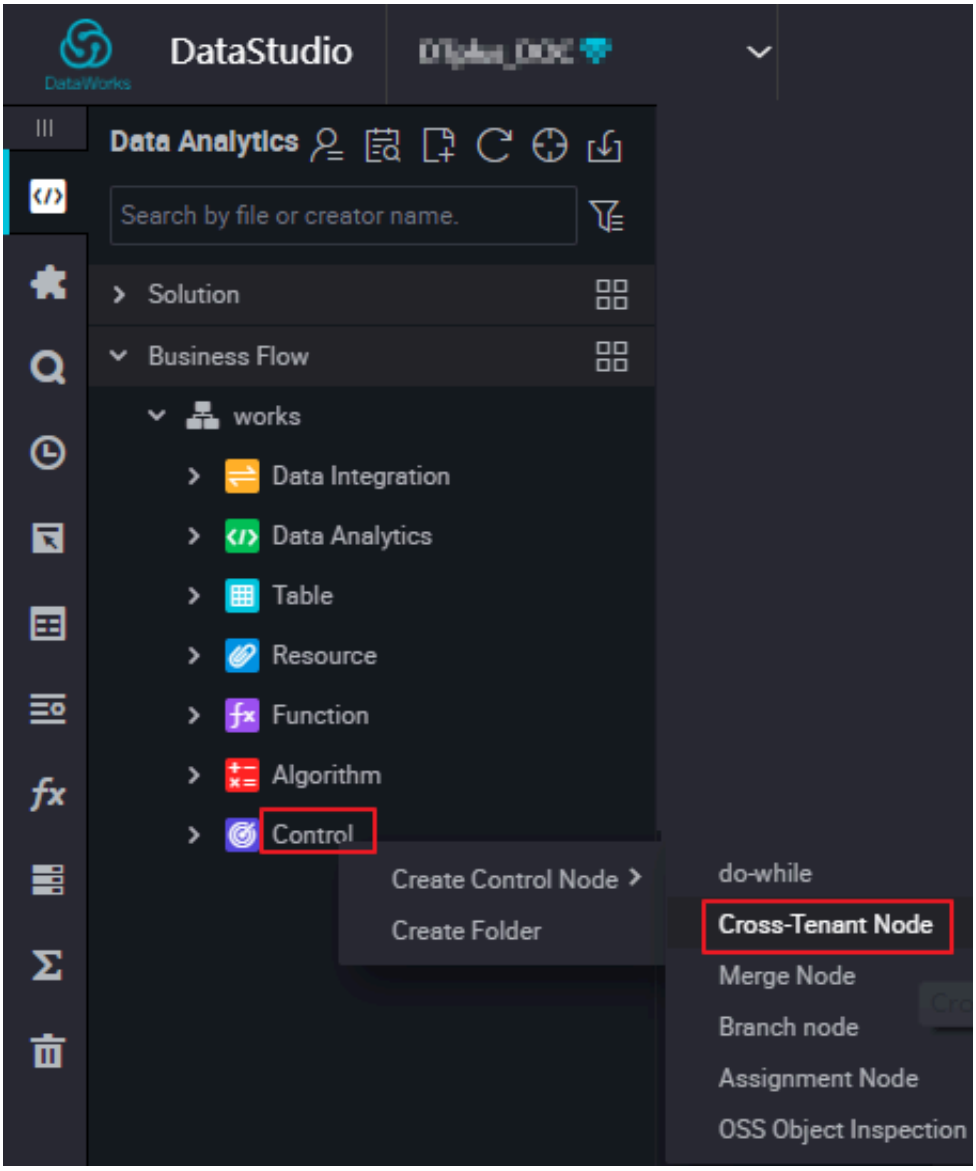
Prerequisites

A sender node and the recipient node must use the same Cron expression. You can choose Schedule > Scheduling Mode to view the Cron expression, as shown in the following figure.



Create a node

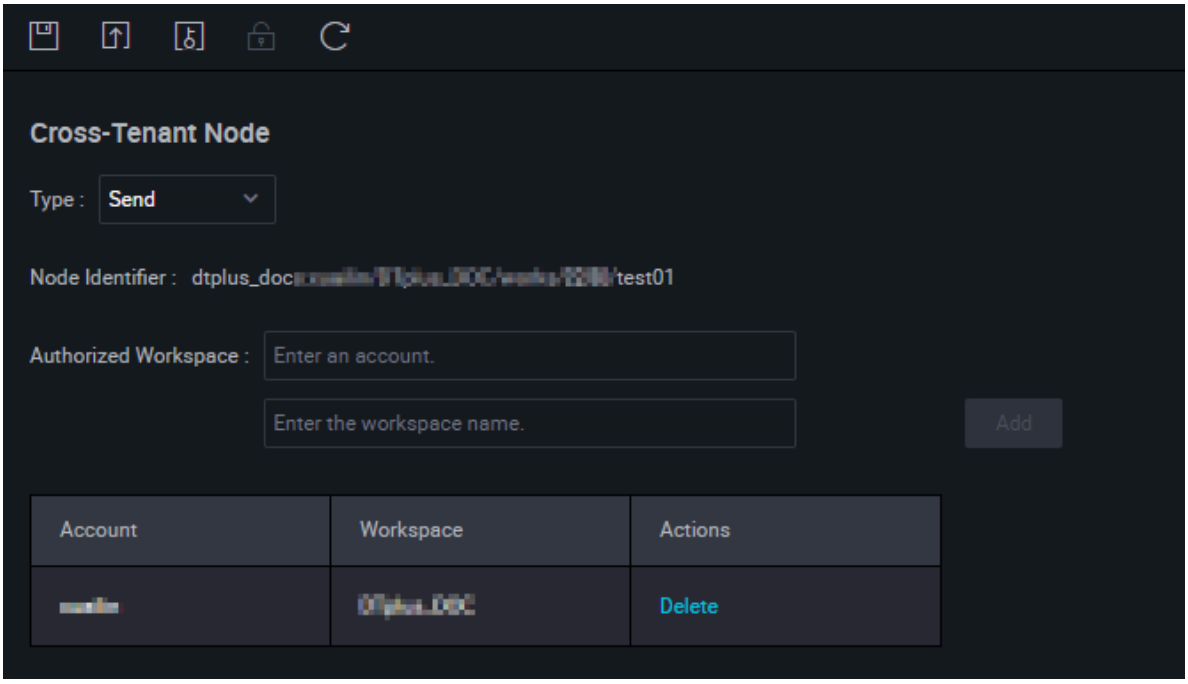
1. On the Data Studio page, right-click Control, and choose Create Control Node > Cross-Tenant Node.



Enter a name in the dialog box and click Submit.

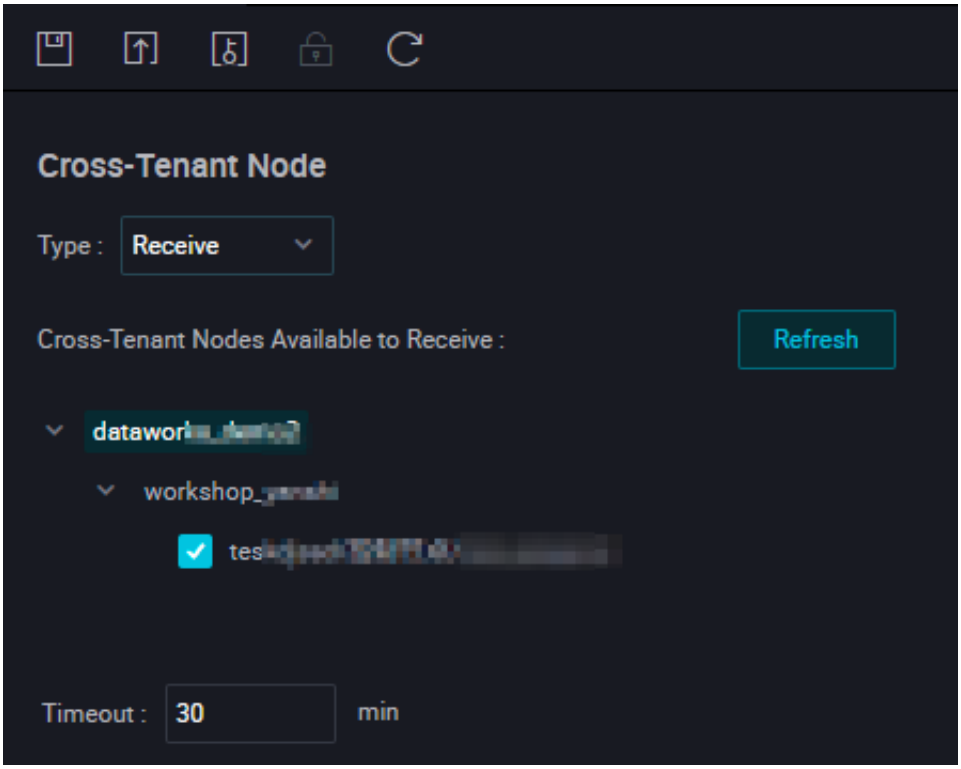
2. Complete the node configuration. Set the node type to Sendor Receive. Authorize a target workspace and a target Alibaba Cloud account. This example sets the node type to Send. Therefore, you need to enter the workspace and account that is

authorized by the recipient node. Save and submit the node after completing the node configuration.



Follow the same procedure to create a control node under the recipient's account and workspace. Set the node type to Receive. Afterward, the information about

the available sender nodes will appear. You must also set the timeout timer. The timeout timer restarts when the recipient node starts running.



The sender node sends a message to the message center, and starts running the message after it is successfully delivered. The recipient node continuously pulls messages from the message center. The recipient node starts running when it successfully pulls a message within the timeout period.

If the recipient node will not be created when it does not receive a message within the timeout period. The timeout of a message can be set to a maximum of 24 hours.

Example:

On October 8, 2018, a periodically created instance was successfully run and a message was sent to the message center. The recipient node is displayed, if you create a retroactive instance for the recipient node with the business date set to October 7, 2018.

1.5.14 Merge node

This topic describes the merge node concept, and how to create a merge node and define the merging logic. It also shows you the scheduling configuration and operation details of the merge node through a practical case.

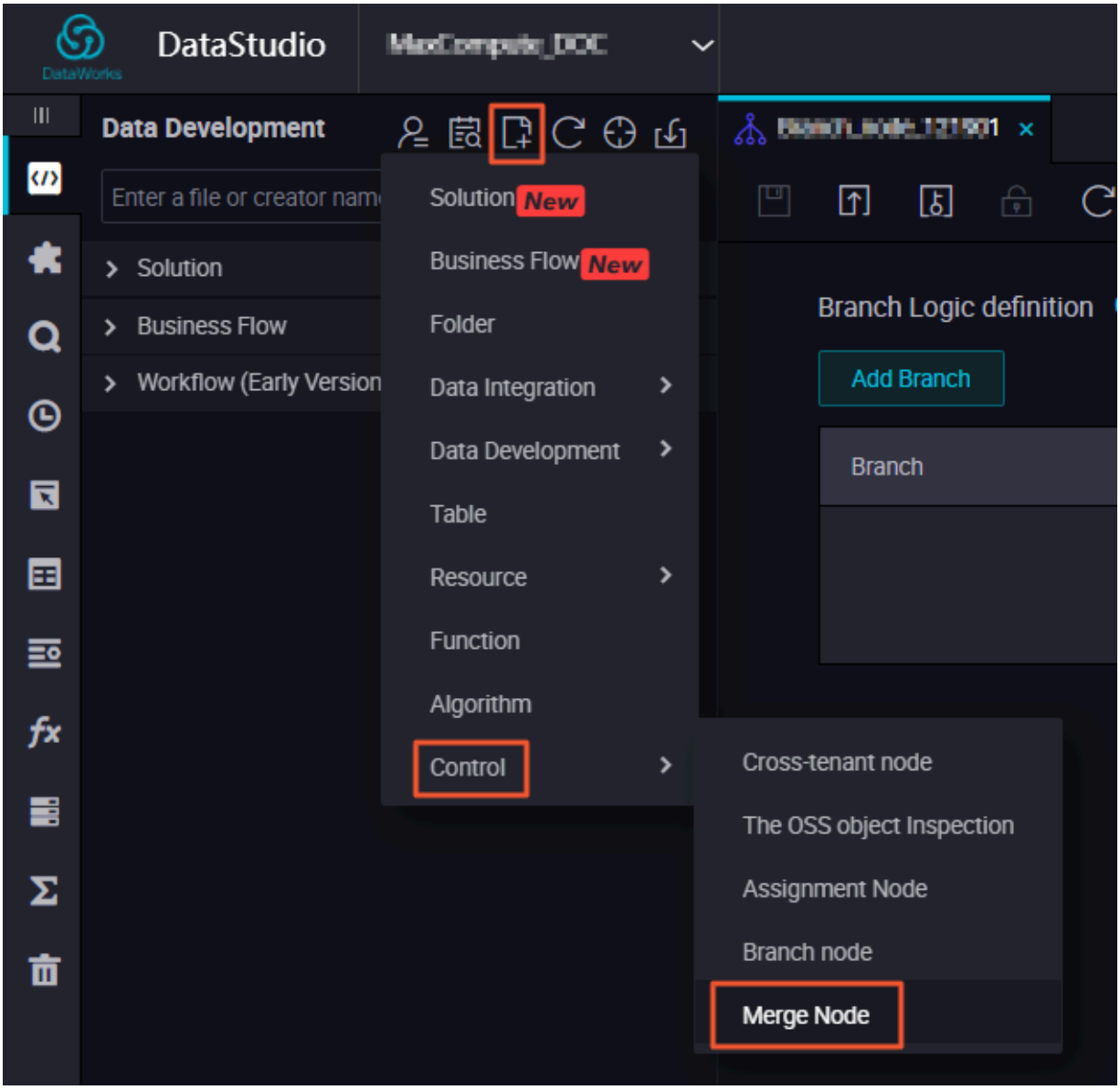
Concept

- The merge node is a type of logical control family nodes in DataStudio.
- The merge node can merge the running states of upstream nodes, and aims to solve the issues of dependency mounting and running trigger of downstream nodes of branch nodes.
- The current logical definition of merge node does not support selecting nodes that are in the running state, but supports merging multiple downstream nodes of the branch nodes, so that more downstream nodes can be mounted to the merge node as a dependency.

For example, the branch node C defines two logically exclusive branches C1 and C2. Different branches use different logic to write to the same MaxCompute table. If the downstream node B depends on the output of this MaxCompute table, and must use the merge node J to merge branches first. Then add merge node J to the upstream dependency of B. If B is mounted directly under C1 and C2, at any given time one of the branch nodes will fail to run because it does not meet branch conditions. B cannot be triggered by the schedule to run.

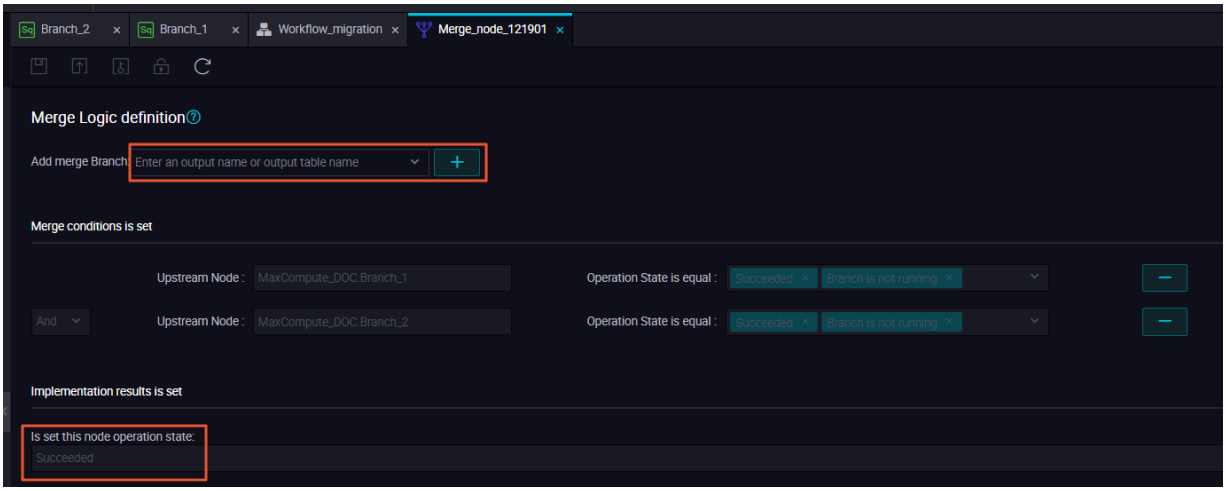
Create a merge node

Merge Node is located in the Control class directory of the new node menu, as shown in the following figure.

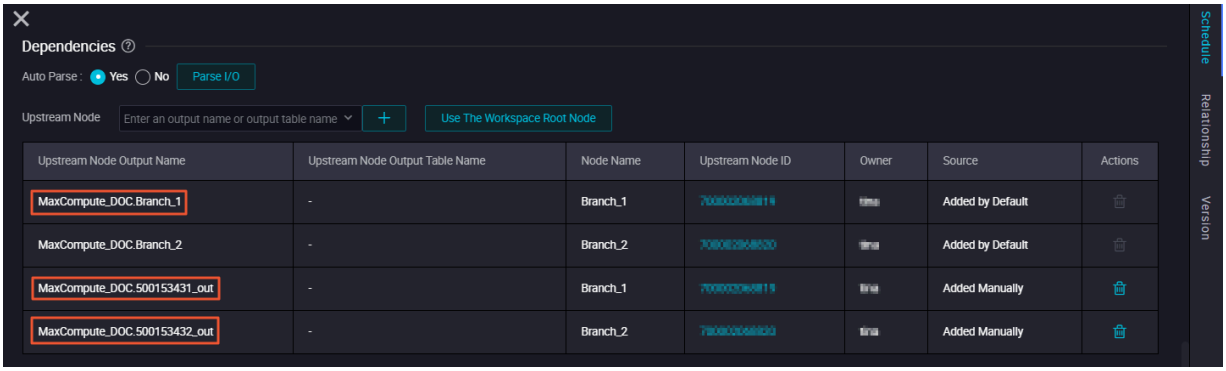


Define the merge logic

To add a merge branch, click Add. You can enter the output name or the parent node output table name, and view records under the merge condition,. The execution results will display the running status. Currently, there are only two running states: Successful, Branch Not Running, as shown in the following figure.

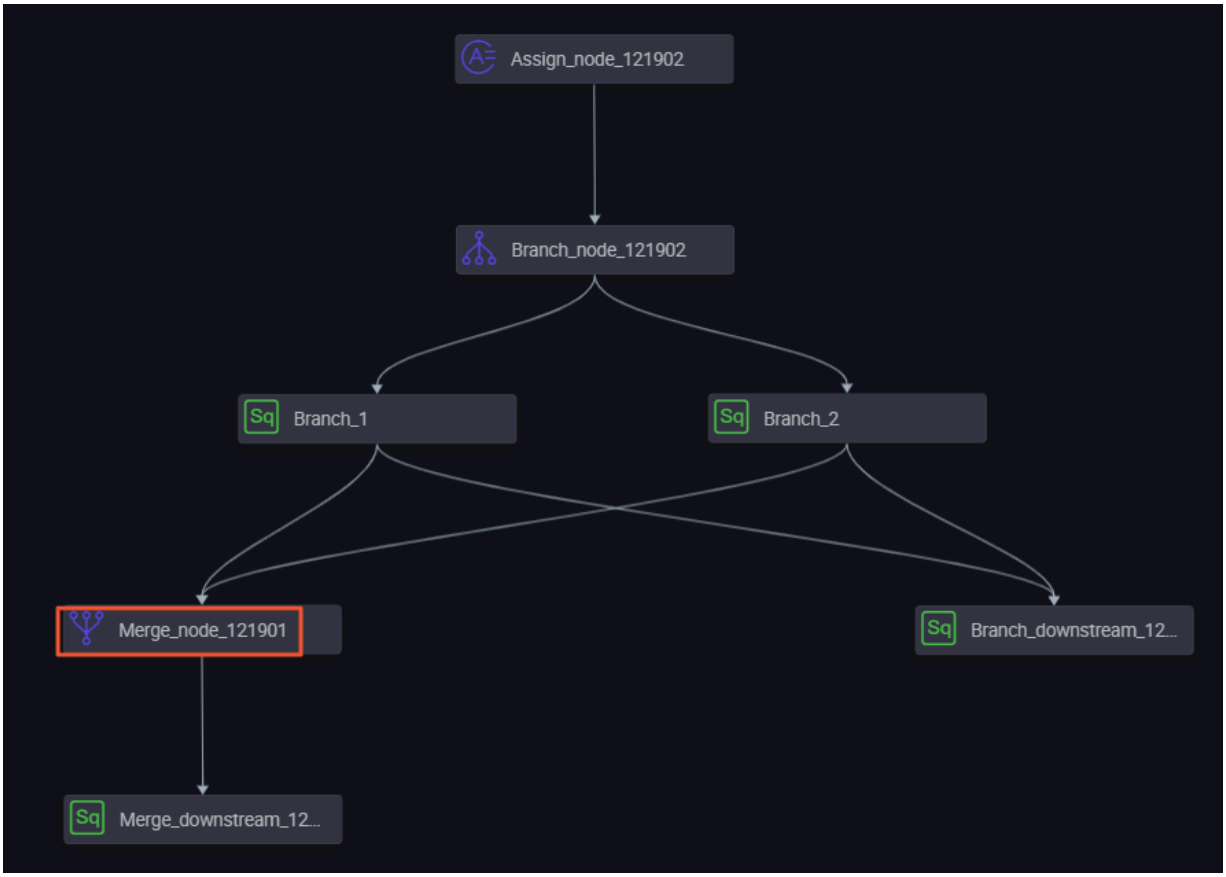


The scheduling attribute of the merge node is shown in the following figure.

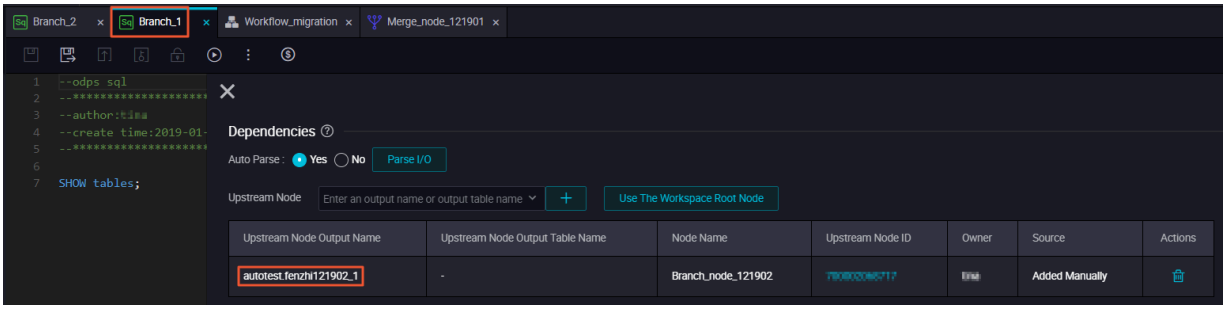


An example of the merge node

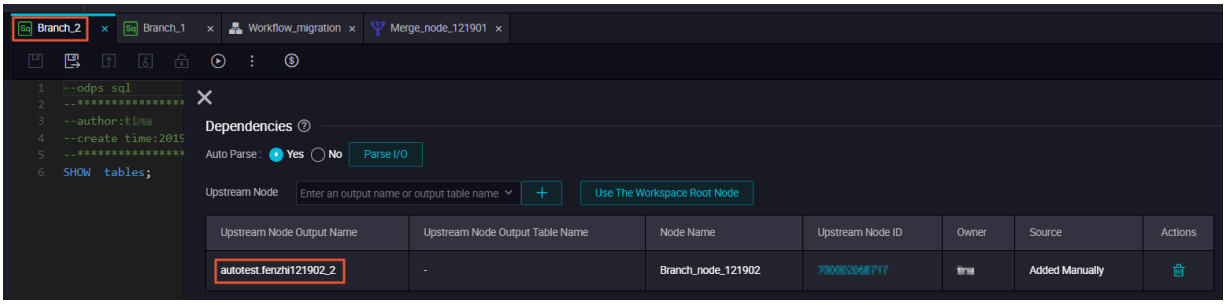
In the downstream node, you can define the branch direction under different conditions by selecting the corresponding branch node output after adding the branch node as the upstream node. For example, in the business process shown in the figure below,Branch_1 and Branch_2 are both downstream nodes of the branch node.



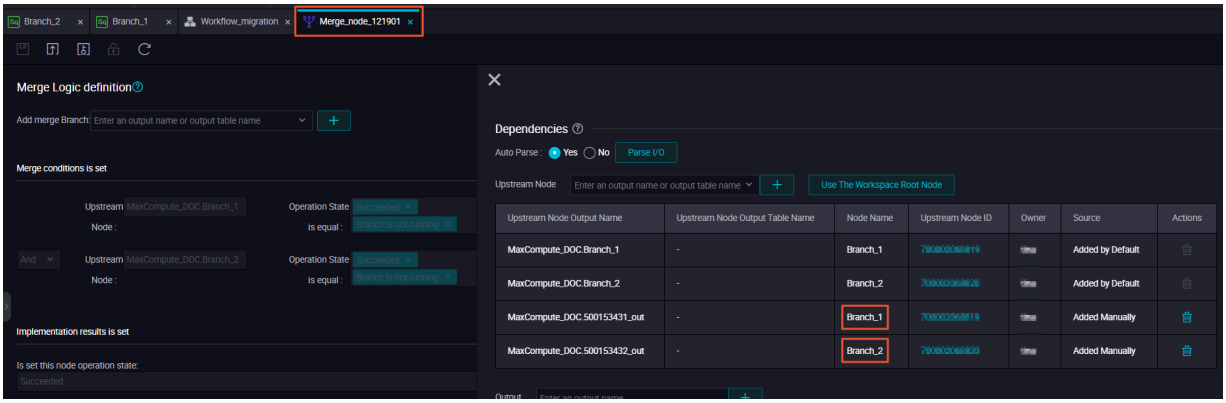
Branch_1 depends on the output of 'autotest.fenzhi121902_1', as shown in the following figure.



Branch_2 depends on the output of 'autotest.fenzhi121902_2', as shown in the following figure.



The scheduling attribute of the merge node is shown in the following figure.



Run the task

When the branch meets the specified condition, select the downstream node of the branch to run. You can view the run details in the Running Log.

When the branch does not meet the condition and does not select the downstream node of the branch to run. You can view the node that is set to 'skip' in the Running Log.

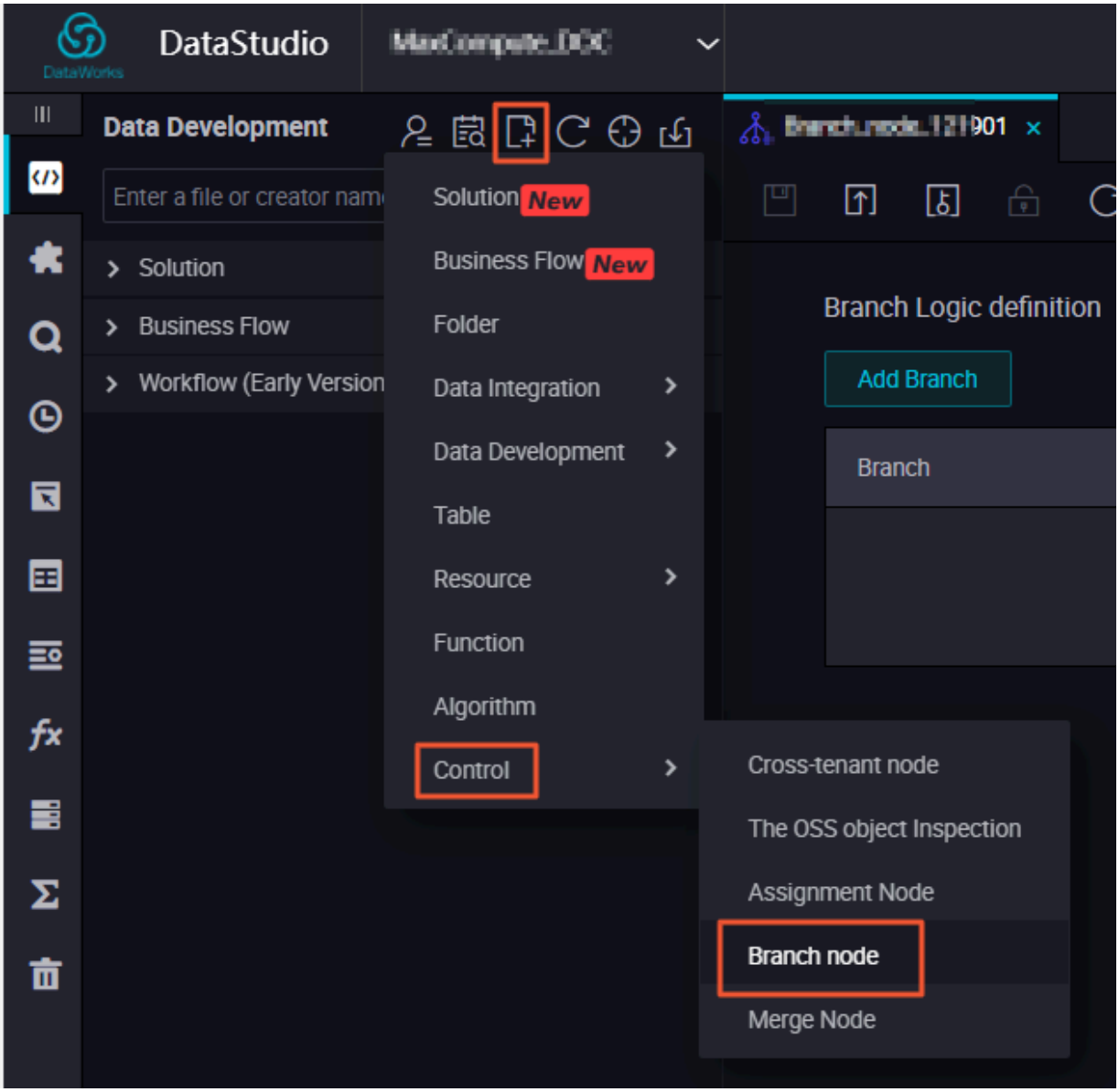
The downstream node of the merge node is running normally.

1.5.15 Branch node

The branch node is a logical control family nodes provided in DataStudio. The branch node can define the Branch Logic and the direction of downstream branches under Different Logical Conditions.

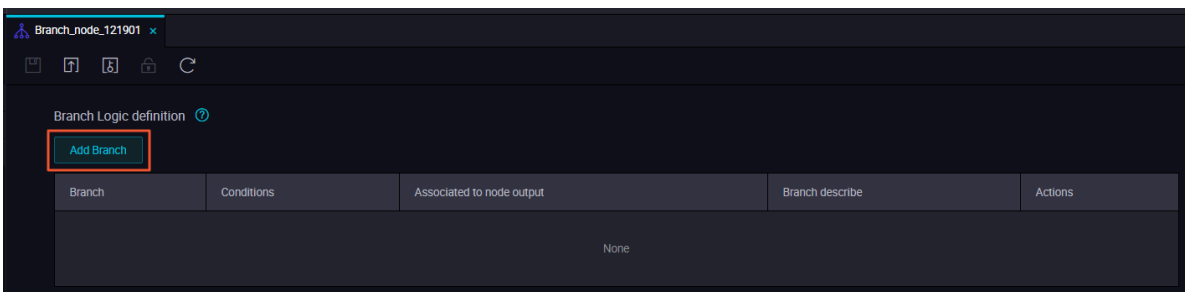
Create a branch node

The branch node is located in the Control class directory of the new node menu, as shown in the following figure.

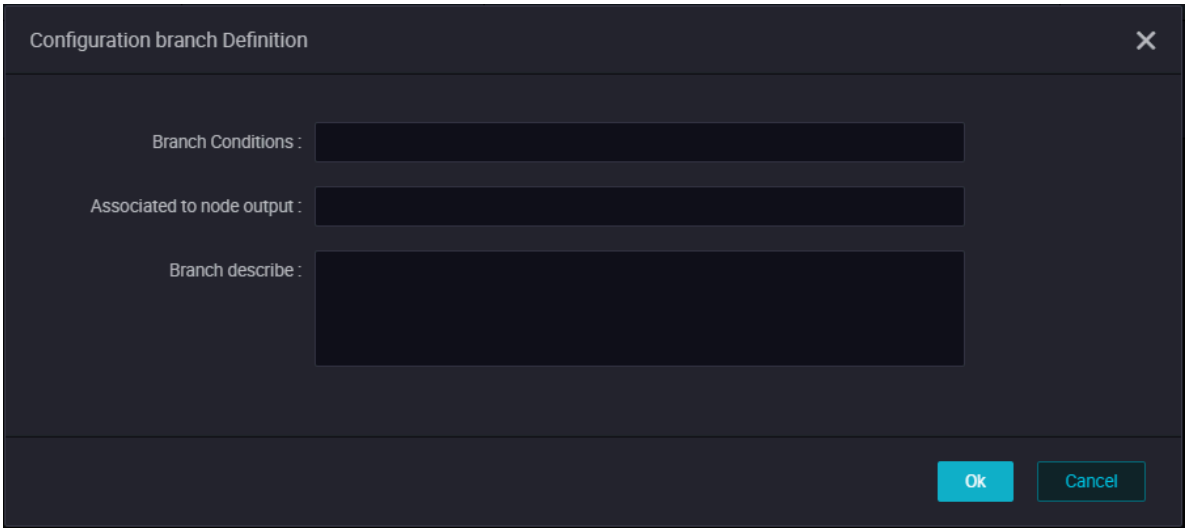


Define the branch logic

1. After creating the branch node, go to the Branch Logic Definition page, as shown in the following figure.



2. In the Branch Logic Definition page, you can use Add Branch button to define the Branch Conditions, Associated to Node Output, and the Branch Describe, as shown in the following figure.



The parameters are as follows:

- Branch conditions
  - The branch condition only supports defining logical judgment condition according to Python comparison operators.
  - If the value of the running state expression is true, it means the correspond ing branching condition is satisfied. Otherwise, the branching condition is unsatisfactory.
  - If a parsing error of the running state expression occurs, the running state of the whole branch node is set to failure.
  - The branching conditions supports using global variables, and parameters defined in the node context, such as `${Input}` in the figure. This can be a node input parameter defined in the branching node.
- Associated to node output
  - The node output is used to mount dependencies for the downstream node of the branch node.
  - When the branch does not meet conditions, the downstream node mounted on the associated node output is selected for running. This also refers to the status of other upstream nodes that the node depends on.

- When the branch does not meet the condition, the downstream node mounted on the associated node output will not run. The downstream node is placed in a not running state because it does not meet the branch condition.

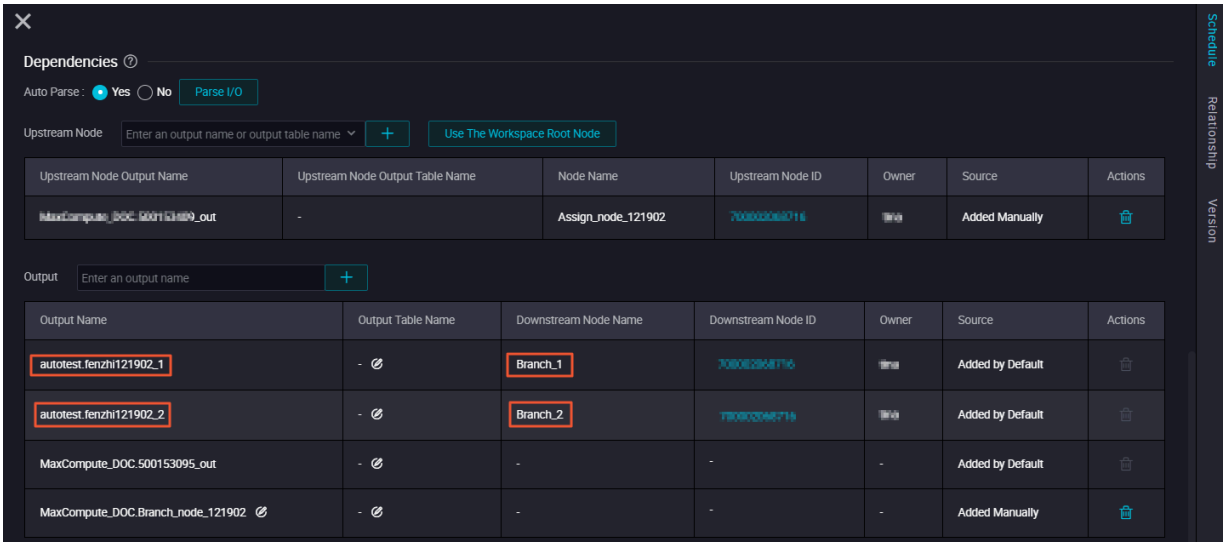
· Branch description: The description of the branch definition.

Define two branches as follows: `${Input}==1` and `${Input}>2`, as shown in the following figure.

- Edit: Click the Edit button to modify the setting branches. The related dependencies will also be updated.
- Delete: Click the Delete button to delete the setting branches. The related dependencies will also be updated.

Scheduling configuration

After defining the branch condition, the output name is automatically added to the node Output of the Schedule, and the downstream node can depend on the output name mount. As shown in the following figure:

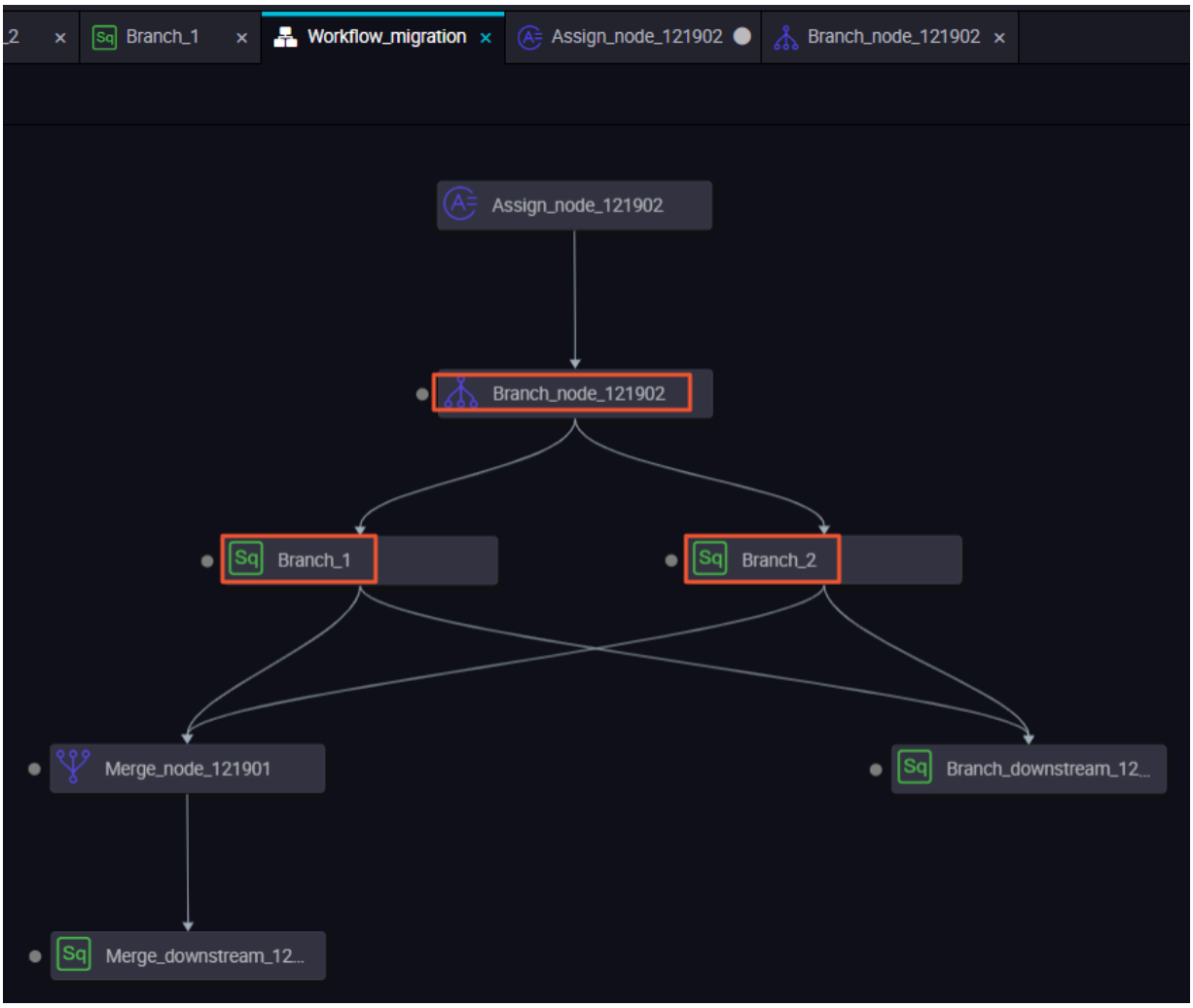


Note:

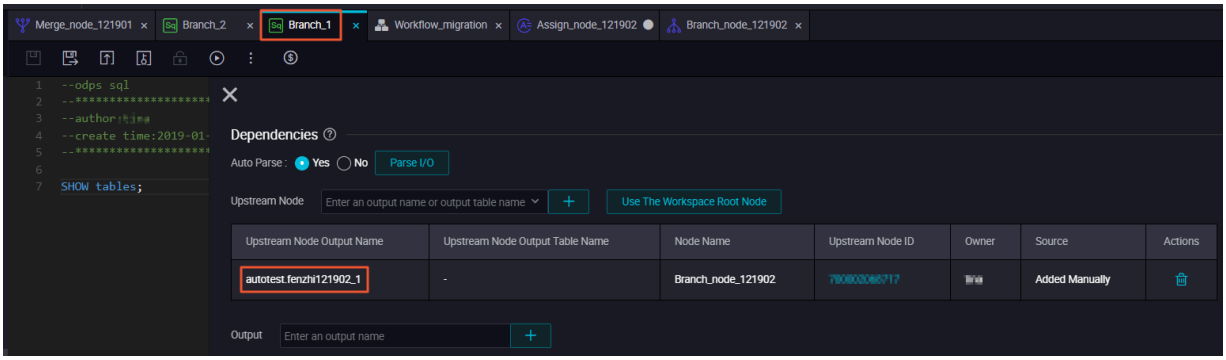
You need to enter output records and context dependencies established by the connection manually if there are no output records in the scheduling configuration for context dependencies.

Output case - downstream node mounted to a branch node

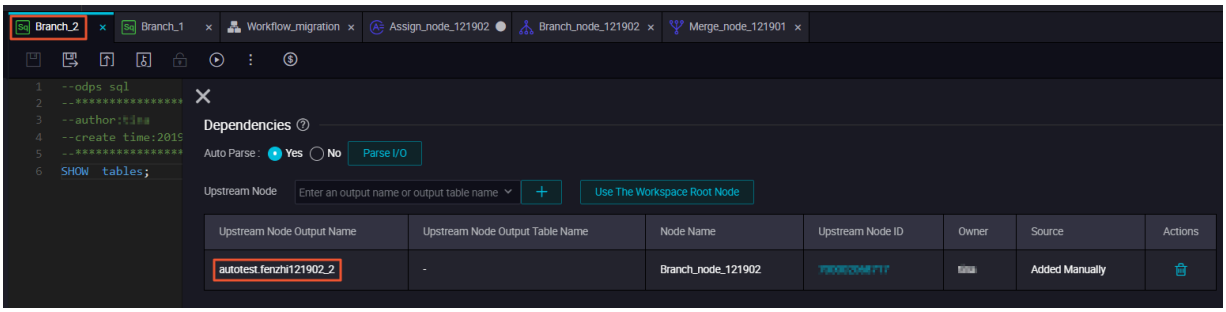
You can define the branch direction under different conditions by selecting the corresponding branch node output in the downstream node, after adding the branch node as the upstream node. For example, in the business process shown in the figure below,Branch_1 and Branch_2 are both downstream nodes of the branch node.



Branch_1 depends on the output of 'autotest.fenzhi121902_1', as shown in the following figure.



Branch_2 depends on the output of 'autotest.fenzhi121902_2', as shown in the following figure.



Submit scheduling operation

Submit the dispatch to the operation center to run, and the branch node satisfies the condition that is dependent on 'autotest.fenzhi121902_1' .Therefore, the print result of the log is as follows.

- When the branch meets the condition, select the downstream node of the branch to run. You can see the details of the run in Running Log.
- When the branch does not meet the condition, do not select the downstream node of the branch run. You can view the node set to 'skip' in the Running Log.

Addition: supported Python comparison operators

In the following table, we assume that variable a is 10 and variable b is 20.

Comparison operators	Description	Example
==	Equal - Compares objects for equality.	(a==b) returns 'false'
!=	Not equal - Compares whether two objects are not equal.	(a!=b) returns 'true'
<>	Not equal - Compares whether two objects are not equal.	(a<>b) returns 'true'. This operator is similar to '!='.

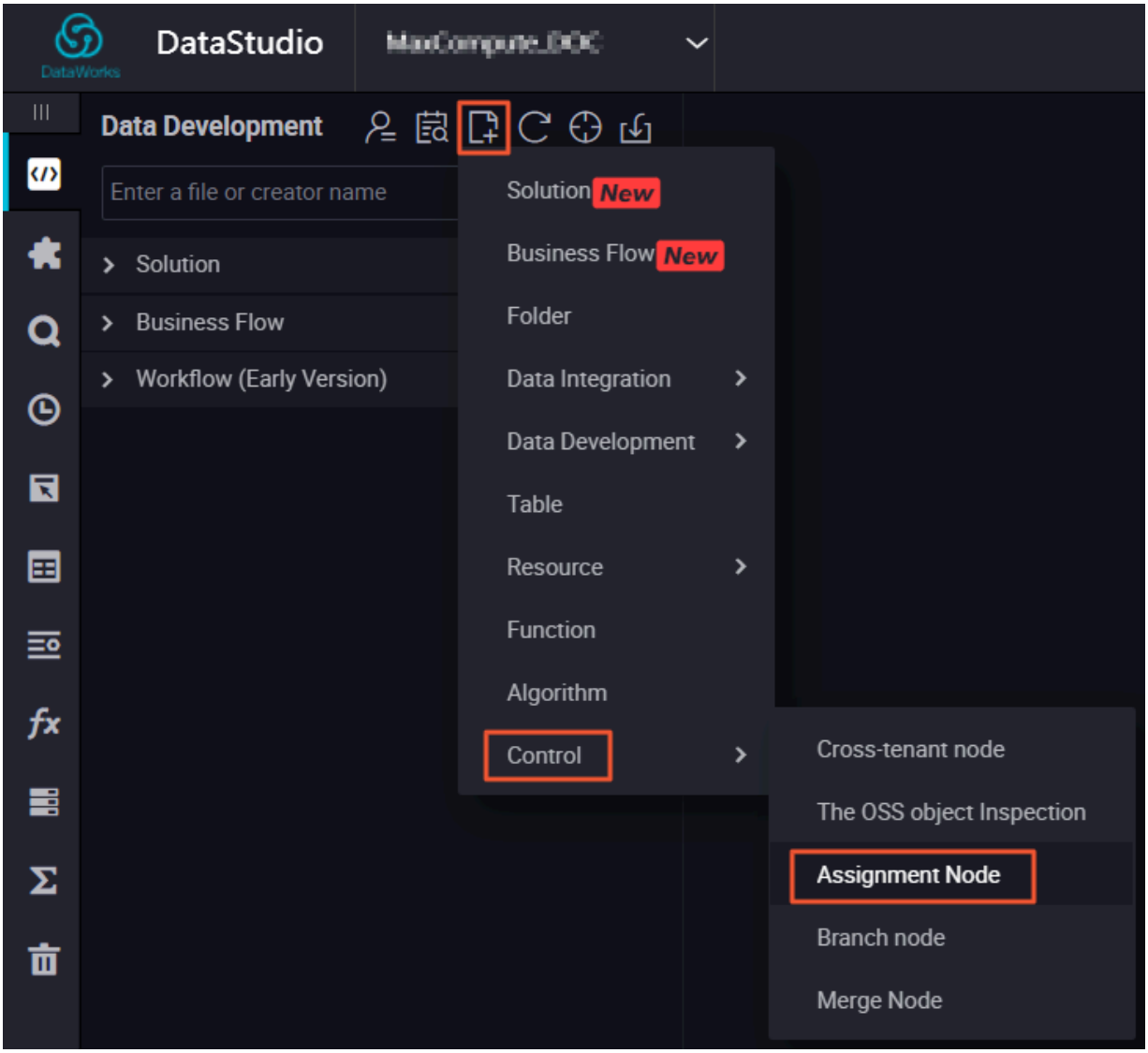
Comparison operators	Description	Example
>	Greater than - Returns whether x is greater than y.	(a>b) returns 'false'
<	Less than - Returns whether x is less than y. All comparison operators return 1 for true, and 0 for false. This is equivalent to the special variables True and False, respectively.	(a<b) returns 'true'
>=	Greater than or equal to - Returns whether x is greater than or equal to y.	(a>=b) returns 'false'
<=	Less than or equal to - Returns whether x is less than or equal to y.	(a<=b) returns 'true'

1.5.16 Assignment node

This topic describes the functions of the Assignment Node. The Assignment Node is a special node type that supports the assignment of output parameters by writing code in the node. The Assignment Node transfers the integrated node context to downstream nodes for reference, which in turn is used as values.

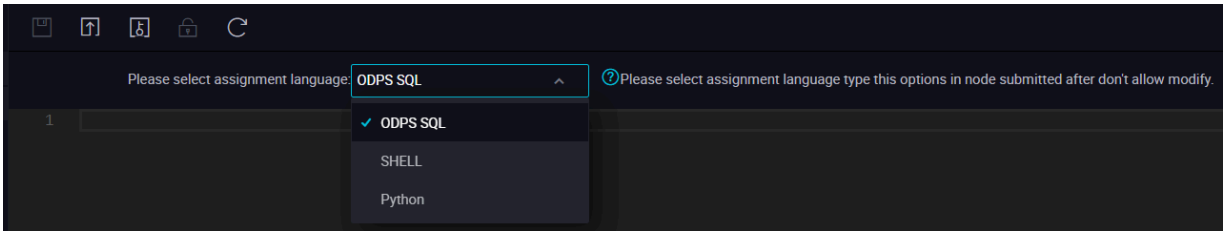
Create an assignment node

Go to Control and click the Assignment Node that is located in the class directory of the new node menu, as shown in the following figure.



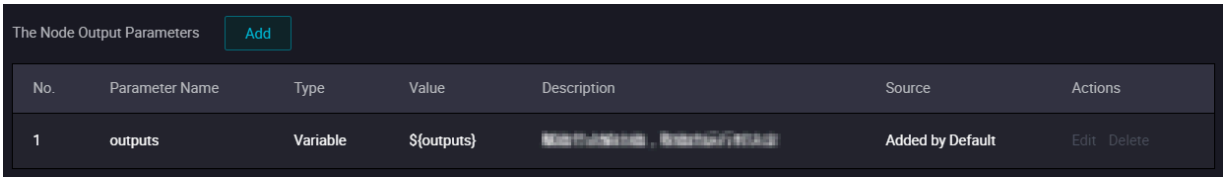
Write the logic value of the assignment node

The assignment node has a fixed output parameter that names outputs in the Node Context. It supports the usage of MaxCompute, Shell, and Python to write code to assign parameters, whose values are the operation and calculation results of the node code. Only one language can be selected for a single assignment node.



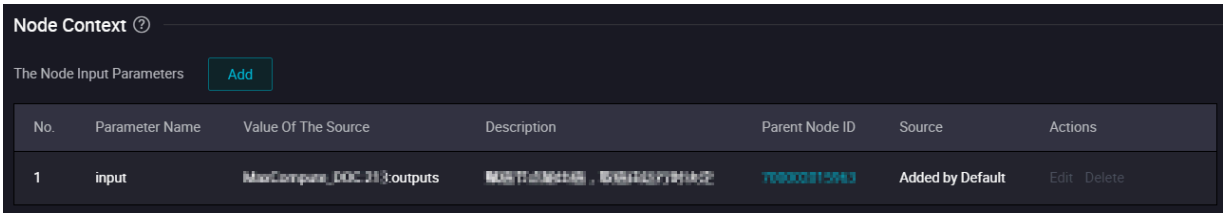
Note:

- The value of the output parameter takes only the output from the last line of code as follows:
  - The output of the SELECT statement on the last line of MaxCompute SQL.
  - The data from the ECHO statement on the last line of shell.
  - The output of the PRINT statement on the last line of Python.
- The maximum transfer value of the output parameter is 2M. If the assignment statement output value exceeds this limit, the assignment node will fail to run.



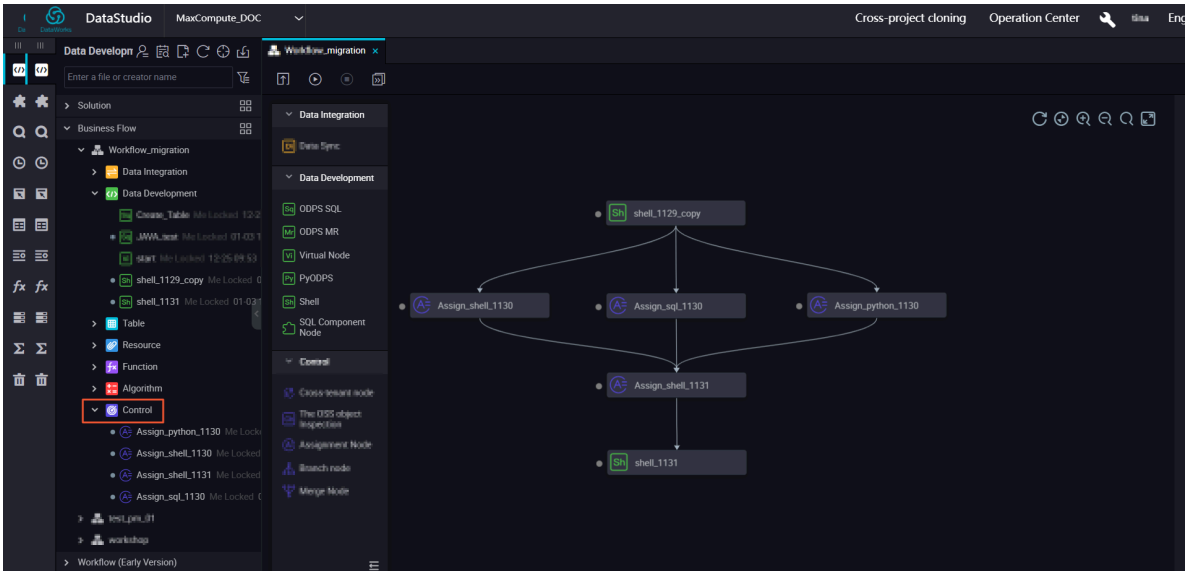
Use the assignment node output on the downstream node

Add an Assignment Node as an upstream dependency in the downstream node, and define the Assignment Node output as an input parameter for the node through node context. Then reference the node in code to obtain the specific values of the upstream assignment node output parameters. For more information, seeNode context.

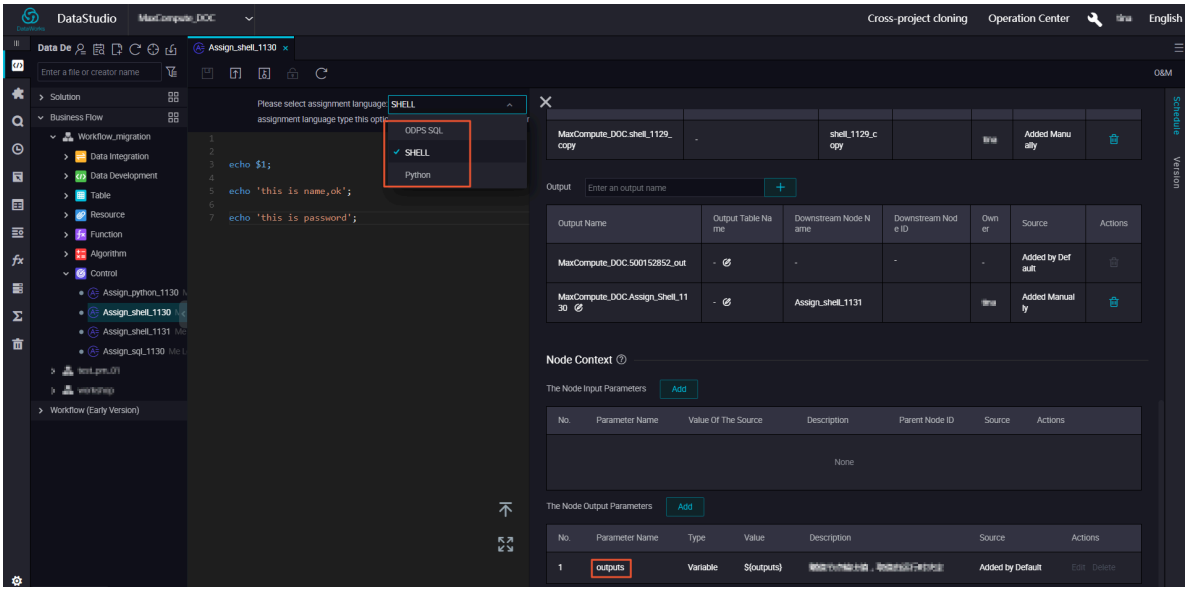


An example of assignment node

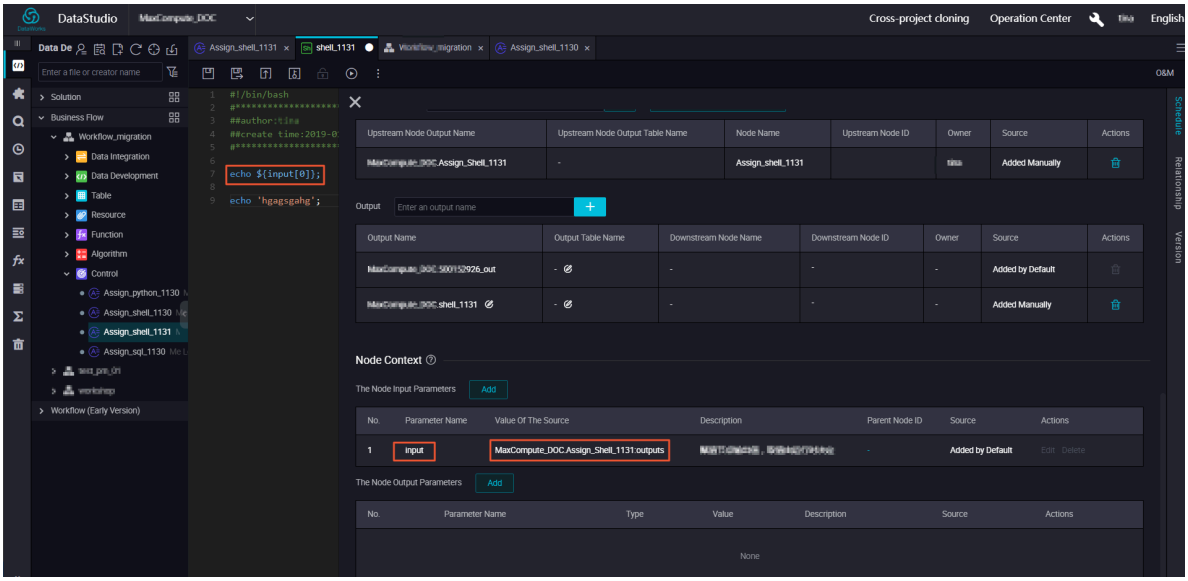
1. Create the business flow, and then create the following nodes as shown in the figure, respectively.



2. By default, the system will display an Outputs parameter when the assignment node is configured. After the task is run, you can find the relevant parameter results in the related Operation Center > Properties > Context page.



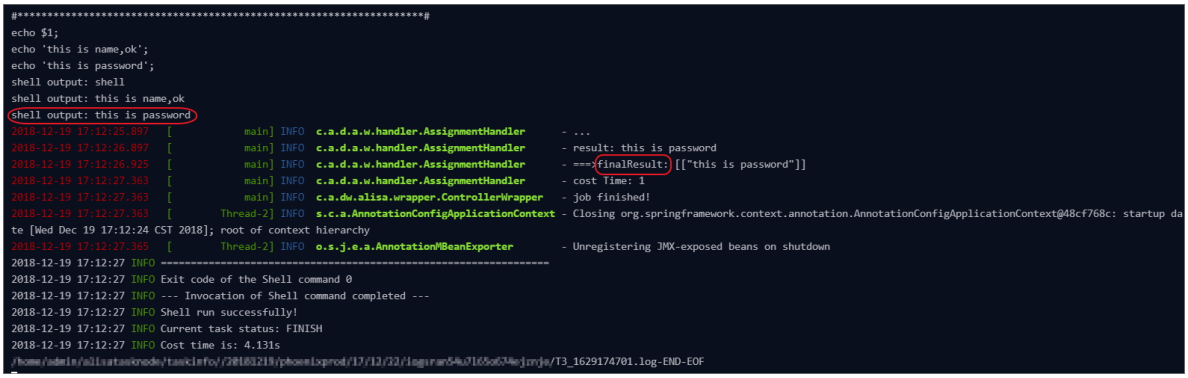
3. The upstream Outputs parameter is used as the downstream input parameter, as shown in the figure below.



Run the assignment node task

> **Note:**
> Typically, you can supplement data running in the above configuration parameters in O&M. The above configuration parameters can be validated through patch data operation, but the test operation parameters cannot be validated.

1. When the task is configured and scheduled, a run instance is generally generated the next day. The following figure is an example of running supplementary data.
2. You can view the context input and output parameters, and click the next link to view the input or output results during runtime.
3. In the Running Log, you can view the final code output through 'finalResult'.

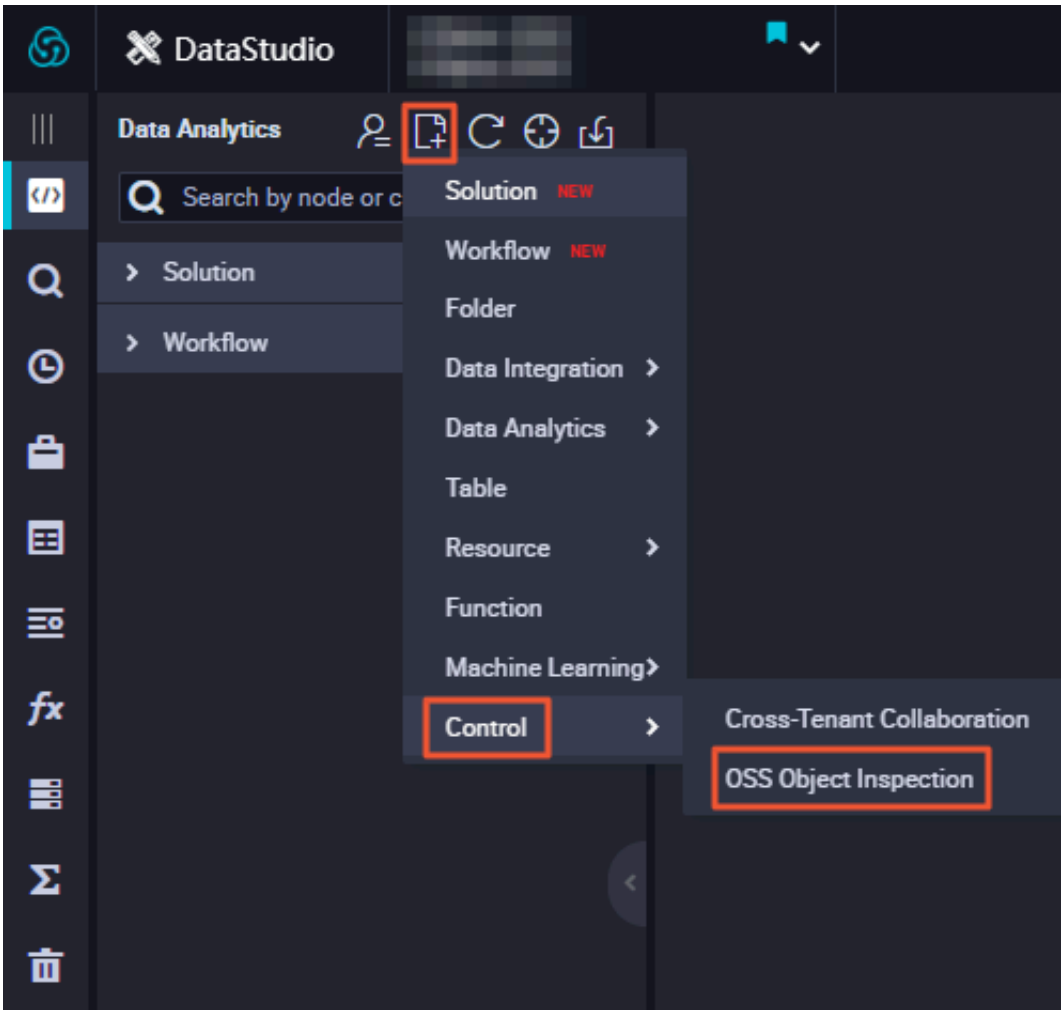


1.5.17 OSS object inspection

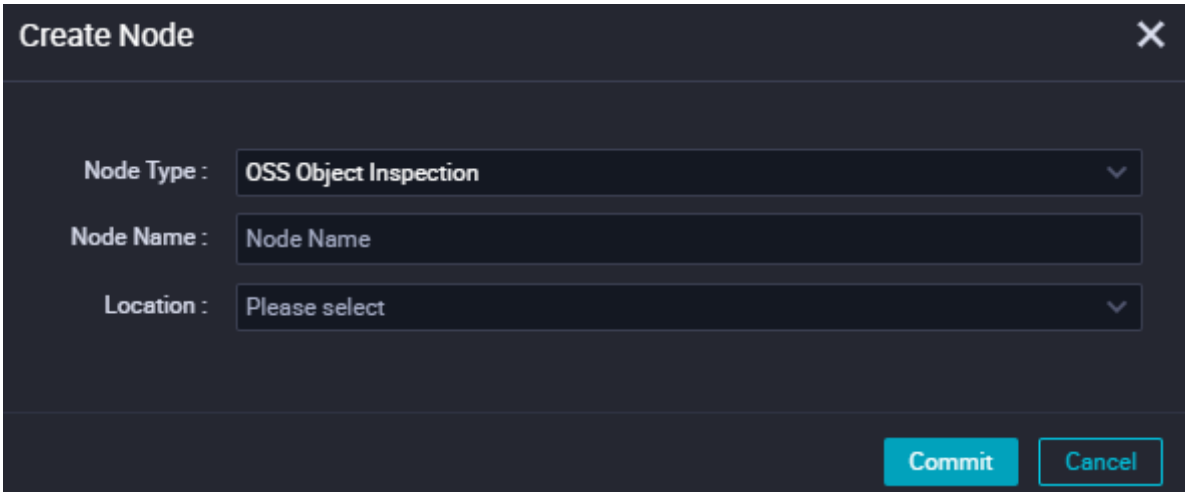
You can use the Object Storage Service (OSS) object inspection feature to monitor OSS objects if child tasks depend on that OSS objects are stored to OSS. For example, you can start a task for synchronizing OSS data files to DataWorks only after the OSS data files are generated. In this case, you can use the OSS object inspection feature to monitor the OSS data files.

Inspection objects: OSS objects of all tenants.

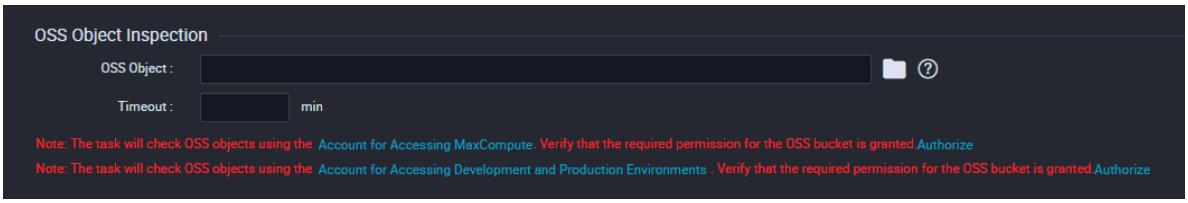
1. In Data Analytics, click the Create icon and choose Control > OSS Object Inspection.



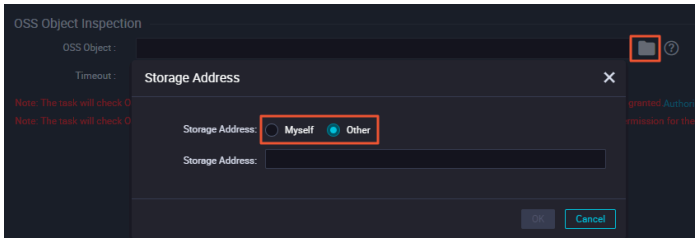
2. In the Create Node dialog box that appears, set the parameters and click Commit.



3. On the OSS Object Inspection page that appears, set the parameters.



No.	Parameter	Description
1	OSS Object	The storage path of the OSS object. You can add a scheduling parameter to the storage path. For more information, see #unique_42.
2	Timeout	The timeout period. During the timeout period, DataWorks checks whether the OSS object exists in OSS every five seconds. If the OSS object is not detected before the timeout period ends, the OSS object inspection task fails.

No.	Parameter	Description
3	Storage Address	The storage space of the OSS object. Valid values: · Myself: detects the OSS object in the storage space of the current tenant. · Other: detects the OSS object in the storage space of another tenant. 

Note:

· When the OSS object inspection task is running, it monitors the OSS object through MaxCompute. Make sure that MaxCompute has the required permissions on the OSS bucket. For more information, see #unique_52.
· In the development or production environment, the task monitors the OSS object through the access identity of the development or production environment. Make sure that the access identity has the required permissions on the OSS bucket.

4. Grant MaxCompute the permission to access OSS in the Resource Access Management (RAM) console.

   MaxCompute uses RAM and Security Token Service (STS) of Alibaba Cloud to resolve security issues of accounts.

   - If the owners of MaxCompute and OSS are using the same Alibaba Cloud account, you can authorize MaxCompute to access OSS with one click in the RAM console.
   - If the owners of MaxCompute and OSS are using different Alibaba Cloud accounts, you can authorize MaxCompute to access OSS as follows:

     a. Create a role in the RAM console.

        Create a role, such as AliyunODPSDefaultRole or AliyunODPSRoleForOtherUser, and set the following policy:

```
-- The owners of MaxCompute and OSS are using
 different Alibaba Cloud accounts .
{
" Statement ": [
{
" Action ": " sts : AssumeRole ",
" Effect ": " Allow ",
" Principal ": {
" Service ": [
" The ID of the Alibaba Cloud account used by
 the owner of MaxCompute @ odps . aliyuncs . com "
]
}
}
],
" Version ": " 1 "
}
```

     b. Create the AliyunODPSRolePolicy permission policy that contains the permissions required for accessing OSS.

```
{
" Version ": " 1 ",
" Statement ": [
{
" Action ": [
 " oss : ListBucket  s ",
 " oss : GetObject ",
 " oss : ListObject  s ",
 " oss : PutObject ",
 " oss : DeleteObje  ct ",
 " oss : AbortMulti  partUpload ",
 " oss : ListParts "
],
" Resource ": "*",
" Effect ": " Allow "
}
```

```
]
}
-- You   can   also   add   other   permission  s   as
 required .
```

c. Grant the AliyunODPSRolePolicy permission policy to the role.

5. Go to Operation Center and view the run logs.

   If the following log information is found, the OSS object is not detected:

```
< Error >
 < Code > NoSuchKey </ Code >
 < Message > The   specified   key   does   not   exist . </ Message
 >
 < RequestId ></ RequestId >
 < HostId > OSS   object </ HostId >
 < Key > xc / 111 . txt </ Key >
</ Error >
```

1.5.18 PAI node

Machine Learning Platform for Artificial Intelligence (PAI) nodes are used to call tasks created on PAI and schedule production activities based on the node configuration. PAI nodes can be added to DataWorks only after PAI experiments are created on PAI.

Create a PAI experiment

Only experiments that can be found on PAI can be loaded into PAI nodes.

Create a PAI node

Follow the instructions in the preceding section to create a PAI experiment. In this example, the experiment name is Heart Disease Prediction_4294. Then, create a PAI node in DataWorks. The procedure is as follows:

1. Select a Business flow you created, right-click Algorithm and choose Create Algorithm Node > PAI.
2. Enter the node name.
3. Select a PAI experiment you created on PAI and load it.

   After the experiment is loaded, click Edit in PAI Console or directly submit the experiment.

1.5.19 Custom node type

1.5.19.1 Overview of custom node types

DataStudio supports default node types such as ODPS SQL and Shell. You can create custom node types to meet your special requirements.

To create a custom node type, you need to create a custom wrapper and use it to define a custom node type.

Open the Node Config page

1. Go to the DataStudio page.
2. Click Node Config in the upper-right corner to go to the Node Config page.

> Note:
> Only the workspace owner and administrators can access the Node Config page.

View the list of wrappers

The Wrappers page displays all the wrappers you have created. You can click Create in the upper-right corner to create a custom wrapper.

- If a node type is created and has not been deployed, Not Deployed is displayed in both the Version in Development Environment and Version in Production Environment columns.
- If a node type has been deployed, the version and the deployment time are displayed in these columns.
- If a node type is under deployment, Deploying is displayed as the version.

You can click Settings, View Versions, or Delete in the Actions column of each wrapper.

Action	Description
Settings	You can click Settings to configure the wrapper. The page that appears depends on the wrapper status. The Deploy in Production Environment page appears if the wrapper has been deployed in the production environment.

Action	Description
View Versions	You can click View Versions to view all historical versions of the wrapper. · View: you can click this button to view the settings of the selected version. · Roll Back: you can click this button to roll back to the selected version. After you click this button, the system creates a new version for the wrapper, and in the new version, the wrapper uses the basic settings and the resource file of the selected version. The new version equals the latest version among all the versions plus 1. · Download: you can click Download to download the resource file of the selected version.
Delete	If an error occurs while a node type is using the wrapper, you need to delete the node type. Note: Before deleting a wrapper, ensure that no node type is associated with the wrapper.

Create a custom wrapper

A wrapper is the core processing logic of a node type. For example, after you write SQL statements in an ODPS SQL node, the system calls the corresponding wrapper to parse and run the statements. You need to create a wrapper before creating a custom node type. Currently, only the Java programming language is supported.

The procedure of creating a wrapper includes four steps: specify settings for the wrapper, deploy the wrapper in the development environment, test the wrapper in the development environment, and deploy the wrapper in the production environment. For more information, see #unique_57.

View default node types

The Default Node Types page is for demonstration purpose only, and configurations displayed on this page cannot be modified. The value of the Tabs column is fixed to Data Analytics.

View the list of custom node types

The Custom Node Types page displays all custom node types in the workspace. You can click Create to create a custom node type. For more information, see #unique_58.

The workspace owner or node type creator can change and delete existing node types.

- Change: you can click Change to edit the settings for the node type.
- Delete: you need to delete a node type if an error occurs while the node type is using the wrapper.

> Note:
> Before deleting a wrapper, ensure that no node type is associated with the wrapper.

Use a custom node type

After a custom node type is created, go to the DataStudio page and click the Create button. The created custom node type is displayed in the cascading menu. Similar to default node types, you can create nodes of the custom type.

1.5.19.2 Create a wrapper

The procedure of creating a wrapper includes four steps: specify settings for a wrapper, deploy the wrapper in the development environment, test the wrapper in the development environment, and deploy the wrapper in the production environment.

Specify settings for a wrapper

1. Go to the Wrappers page, click Create in the upper-right corner.
2. Specify the parameters on the Settings page.

Parameter	Description
Name	A wrapper name must start with a letter and can only contain letters, numbers, and underscores (_).
Owner	You can select an owner from the workspace members. You are not allowed to edit wrappers owned by other members even if you are an administrator. Only the workspace owner can edit the wrappers of other members.
Resource Type	Two types are supported: JAR and Archive. Archive indicates the ZIP file format.

Parameter	Description
Resource File	You can either upload a local file or enter the path of a file stored in an OSS bucket. Note: The size of a local file can be up to 50 MB, and the size of a file that is stored in an OSS bucket can be up to 200 MB.
Class Name	Enter the full path of the class in user wrapper implementation.
Parameter Example	Design parameters based on the package you upload.
Version	Select Create Version if you are creating a new version. Select Overwrite Version if you are editing and rolling back a version.
Description	Enter a description for the wrapper version.

3. Click Save and then click Next.

Note:
The settings are updated to the database after you click Save.

- If you only modify basic settings of a wrapper without changing the resource file, the modification takes immediate effect after you click Save.
- If you change the resource file, the change only applies after deployment.

Deploy the wrapper in the development environment

After you specify the parameters on the Settings page and click Next, the information on the Deploy in Development Environment page is updated accordingly. You can identify the changes by checking the file name and MD5 checksum.

Click Deploy in Development Environment. You can view the deployment progress in real time. After the wrapper is deployed, click Next.

Test the wrapper in the development environment

Specify arguments for testing, and click Test to send the arguments to the wrapper. This step is to validate deployment and logic of the wrapper. You can also locally test the wrapper before upload it for deployment.

After the test, review the output logs in the Test Results section on the right to determine whether the test is passed. If the test is passed, select Test Passed and click Next.

Deploy the wrapper in the production environment

After you click Deploy in Production Environment, the wrapper is deployed in the production environment. You can view the deployment progress in real time.

 Note:
The wrapper to be deployed in the production environment must be the latest version that has been deployed in the development environment and have passed the test. Otherwise, a message appears, indicating that the deployment in the production environment fails.

Click Complete. You can view and edit your wrappers on the Wrappers tab.

1.5.19.3 Create a custom node type

The Configure Custom Node Type page consists of three sections: Basic Information, Interaction, and Wrapper.

1. On the DataStudio page, choose Node Config > Custom Node Types.
2. Click Create in the upper-right corner.
3. Specify the parameters in the Basic Information section.

Parameter	Description
Name	Name the node type. The name cannot be changed after the node type is created. Each node type has a unique name within the workspace. The name is up to 20 characters in length, and can only contain letters, spaces, and underscores (_).
Tabs	You can select Ad-Hoc Query, Data Analytics, and Manually Triggered Workflows.
Folder	You can select Data Integration or Data Analytics.

4. Specify the parameters in the Interaction section.

Parameter	Description
Shortcut Menu	· The following options are selected by default: Rename, Move, Clone, Steal Lock, View Versions , Locate in Operation Center, Delete, and Submit for Review. · You can also select Send to DataWorks Desktop (Shortcut).
Tool Bar	· The following options are selected by default : Save, Commit, Commit and Unlock, Steal Lock, Run, Show/Hide, Run with Arguments, Stop, Reload, Run Smoke Test in Developmen t Environment, View Smoke Test Log in Development Environment, Run Smoke Test, View Smoke Test Log, Go to Operation Center of Development Environment, and Format. · You can also select Precompile.
Editor Type	You can select Editor Only or Data Source Selection Section and Editor.
Right-Side Bar	· The Properties and Versions options are selected by default. · You can also select Lineage and Code Structure.
Auto Parse Option	If you enable Auto Parse Option, the Auto Parse option is displayed in the Properties tab. Otherwise , it is not displayed. If you set Auto Parse to Yes for a node, the input and output of the node is automatica lly parsed from the code.

5. Specify the parameters in the Wrapper section.

   · The following table describes the parameters you need to specify if you set the editor type to Editor Only.

Parameter	Description
Wrapper	Select a wrapper that has been deployed.
Editor Language	You can select JSON or ODPS SQL.

Parameter	Description
Use MaxCompute as Engine	Select Yes if your wrapper uses MaxCompute as the compute engine. Select No in other scenarios. This parameter is set to Yes by default.

- The following table describes the parameters you need to specify if you set the editor type to Data Source Selection Section and Editor.

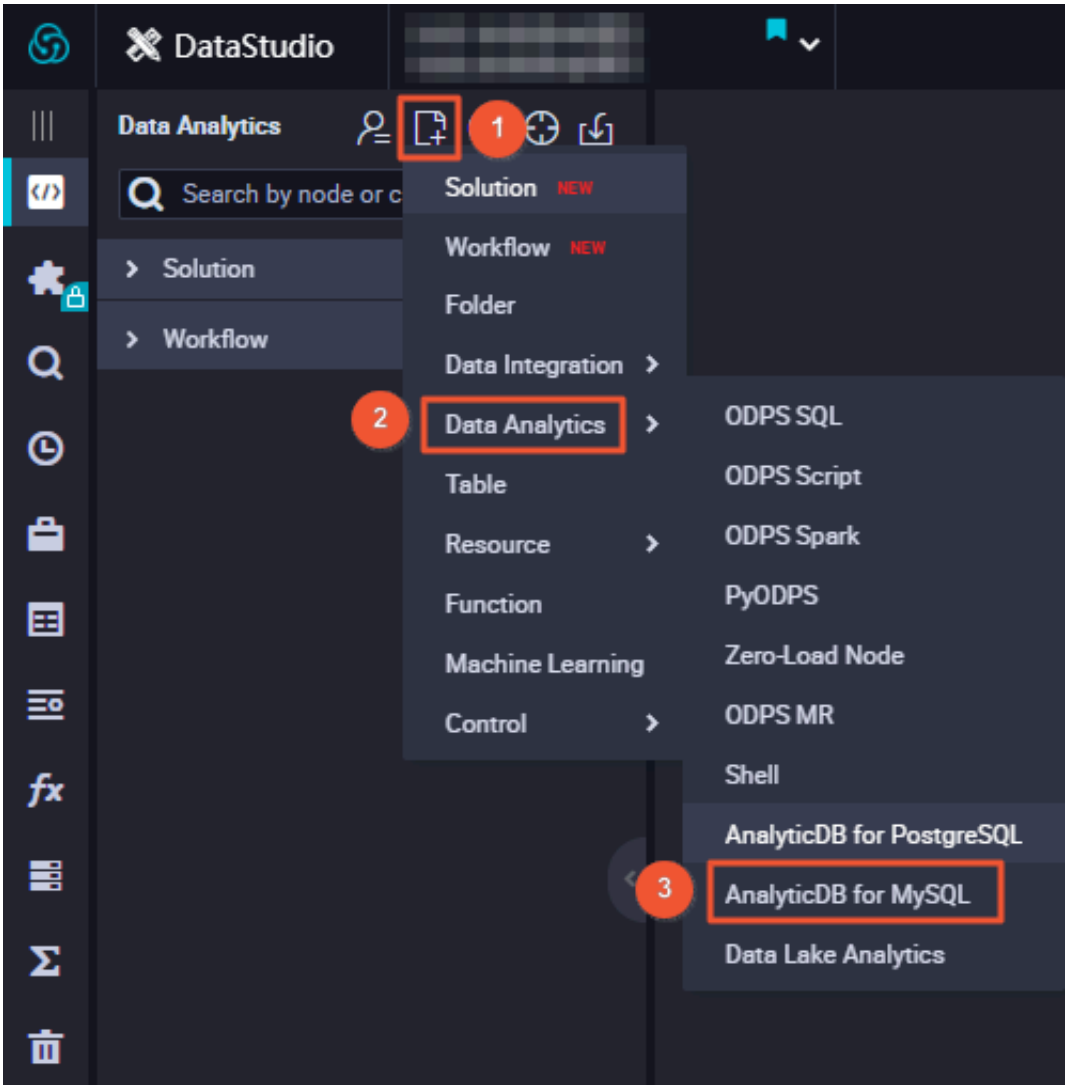
Parameter	Description
Wrapper	Select a wrapper that has been deployed.
Editor Language	You can select JSON or ODPS SQL.
Connection Type	Select the type of connections.

6. Click Save and Exit to create the custom node type. Then, you can use the custom node type that is created.

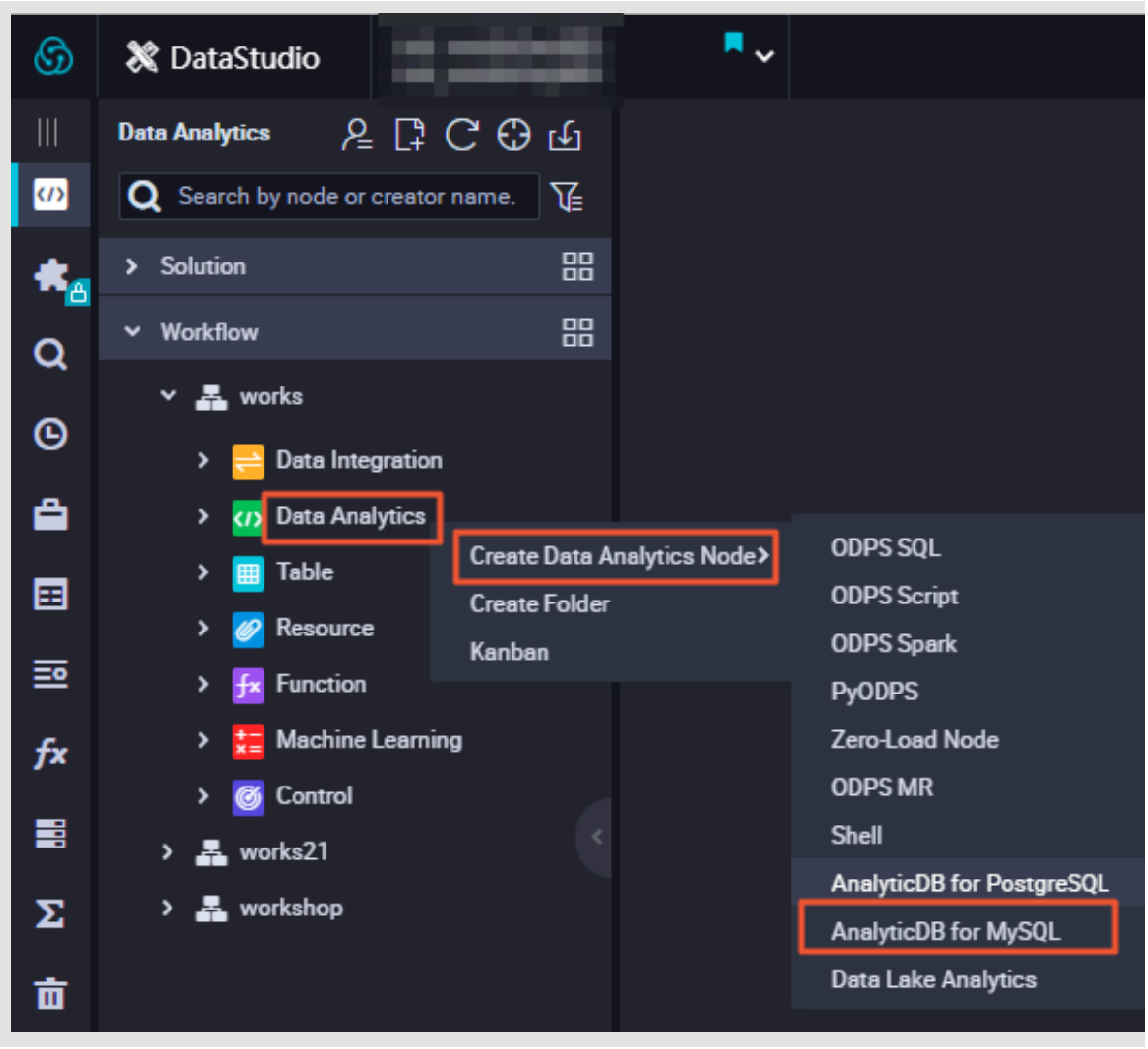
1.5.20 AnalyticDB for MySQL node

You can create an AnalyticDB for MySQL node in DataWorks to build an online ETL process.

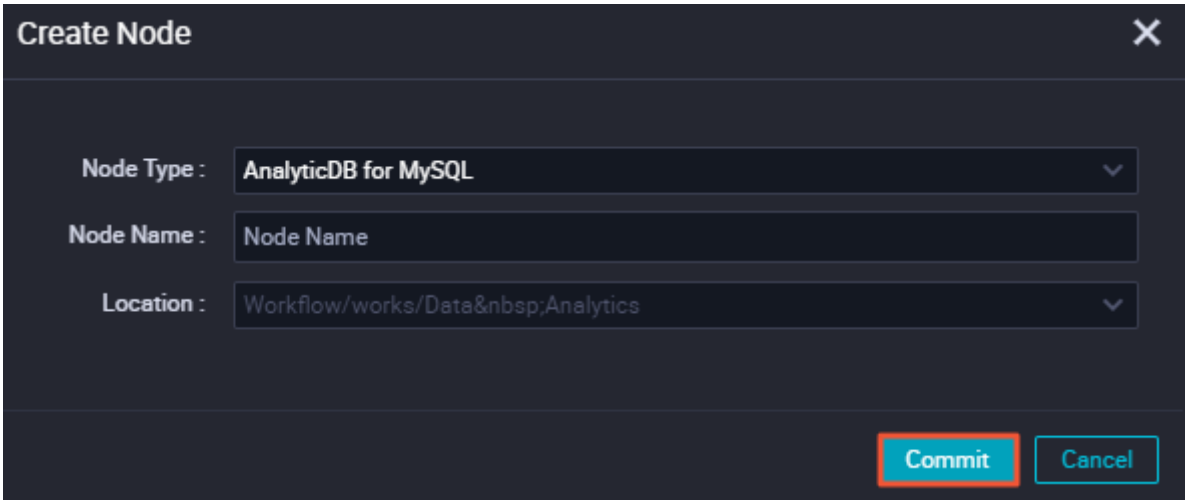
1. Go to the DataStudio page, and choose Create > Data Analytics > AnalyticDB for MySQL.



Note:
You can also select a workflow, right-click Data Analytics, and then choose Create Data Analytics Node > AnalyticDB for MySQL.



2. In the Create Node dialog box, enter the Node name, select the Destination folder, and then click Commit. The Location field is optional. You can specify this field to classify and manage nodes.

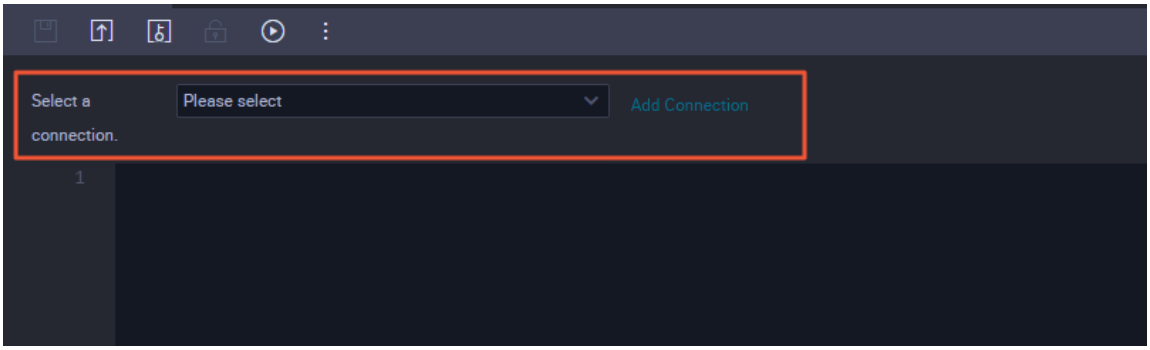


3. Edit the AnalyticDB for MySQL node.

   You can select a connection and edit SQL code on the node editing tab.

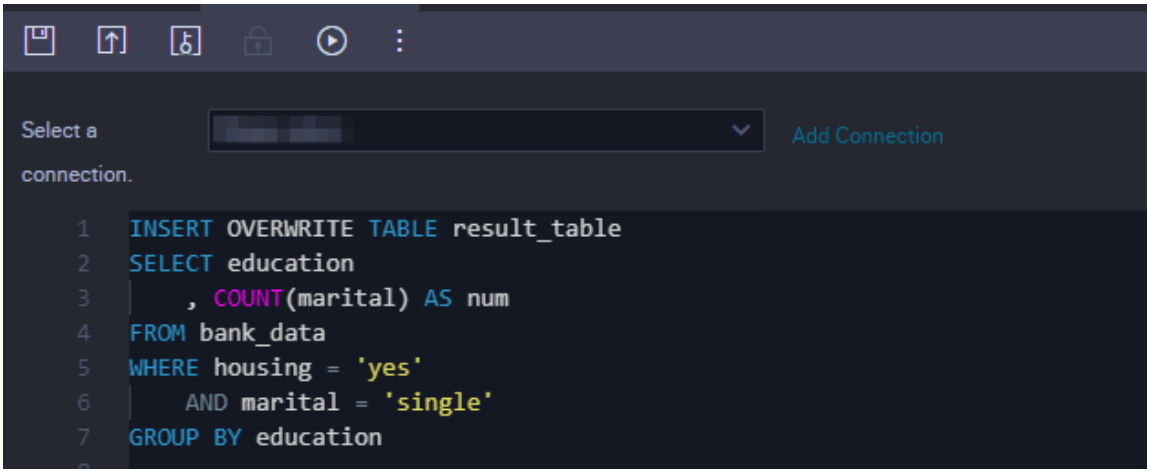
   a. Select a connection.

      Select a target connection for the node. If you cannot find the required connection in the drop-down list, click Add Connection to open the Add Connection page. You can add the connection on the Data Integration page. For more information, see Configure a connection.

   b. Edit SQL statements.

      After selecting a connection, you can write SQL statements based on the syntax supported by AnalyticDB for MySQL. You can write DML and DDL statements in the code editor.

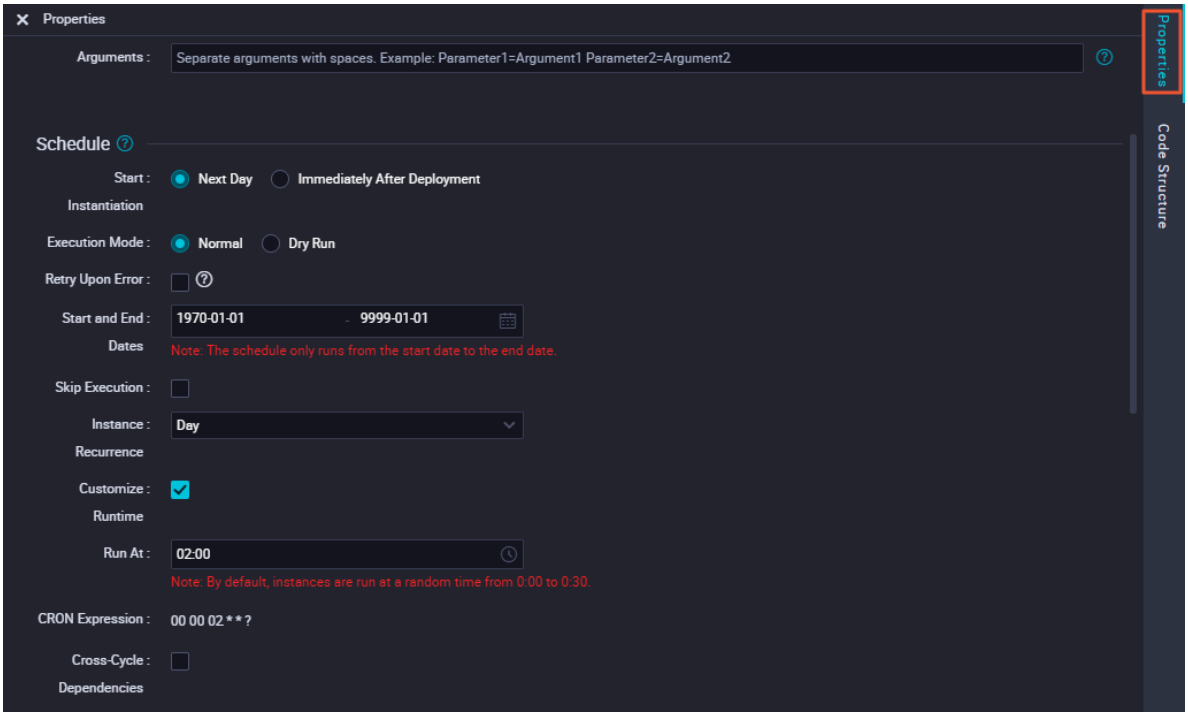
      

   c. Save and run the SQL statements.

      After you finish editing the SQL statements, click the Save button to save the settings of the node to the server. Then, click the Run button to run the SQL statements you have saved.

4. Set properties of the node.

   Click Properties on the right of the node editing tab to go to the Properties page. For more information, see Properties.



5. Commit the node.

   After you set the properties, click Save in the tool bar to commit the node to the development environment. After you commit the node to the development environment, the node is unlocked.

6. Deploy the node.

   For more information, see Deploy a node.

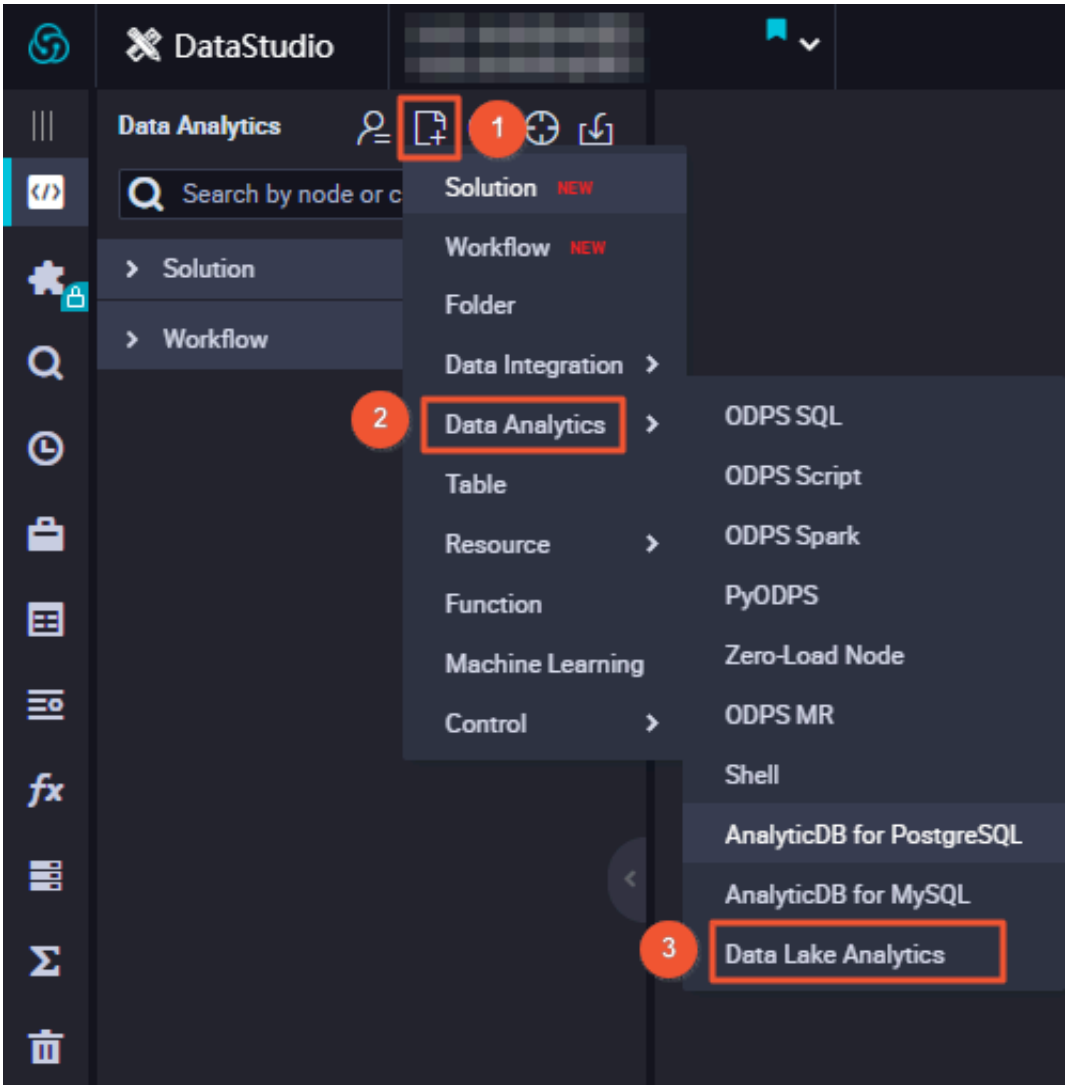
7. Test the node in the production environment.

   For more information, see #unique_32.

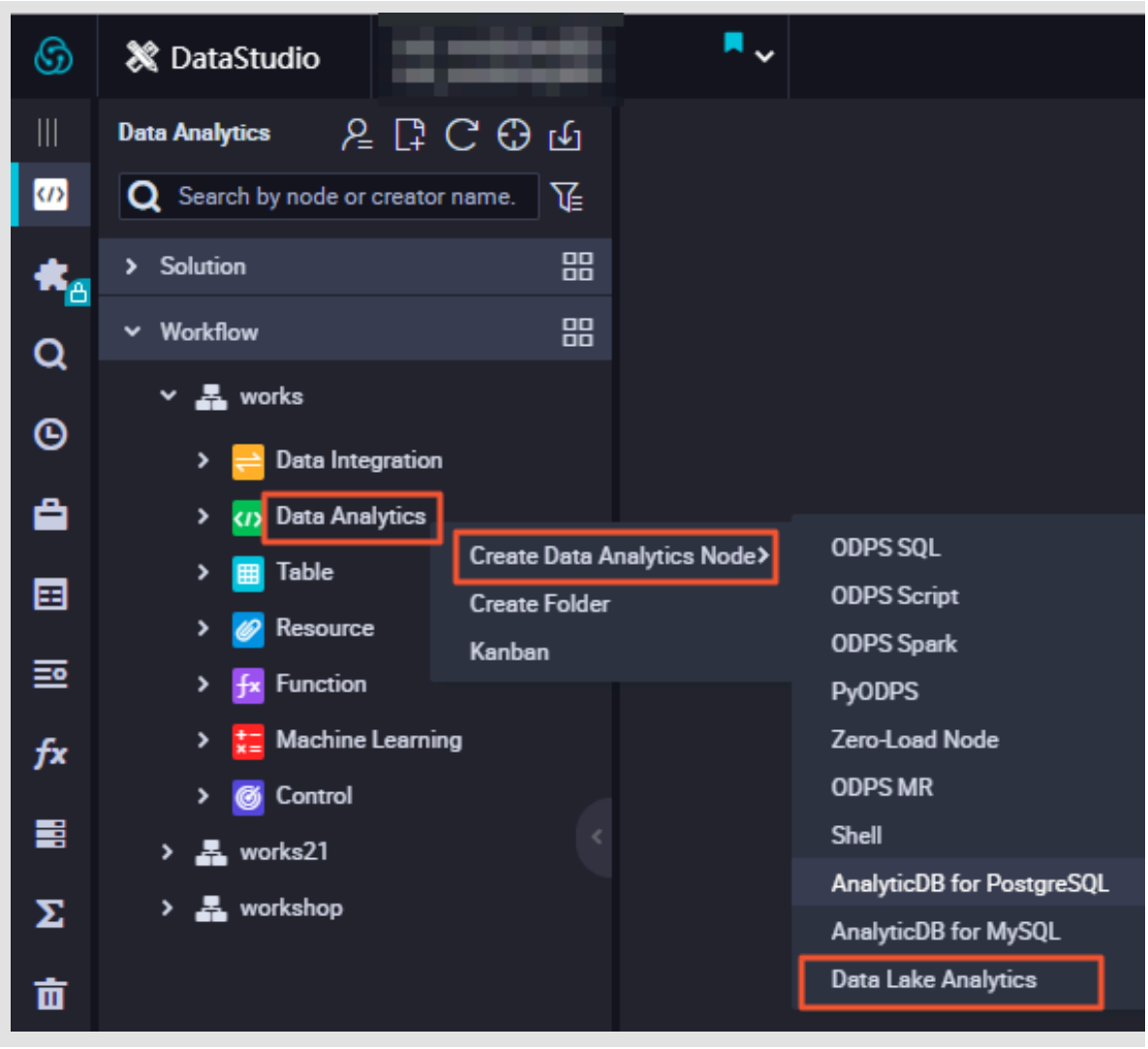
1.5.21 Data Lake Analytics node

You can create a Data Lake Analytics node in DataWorks to build an online ETL process.

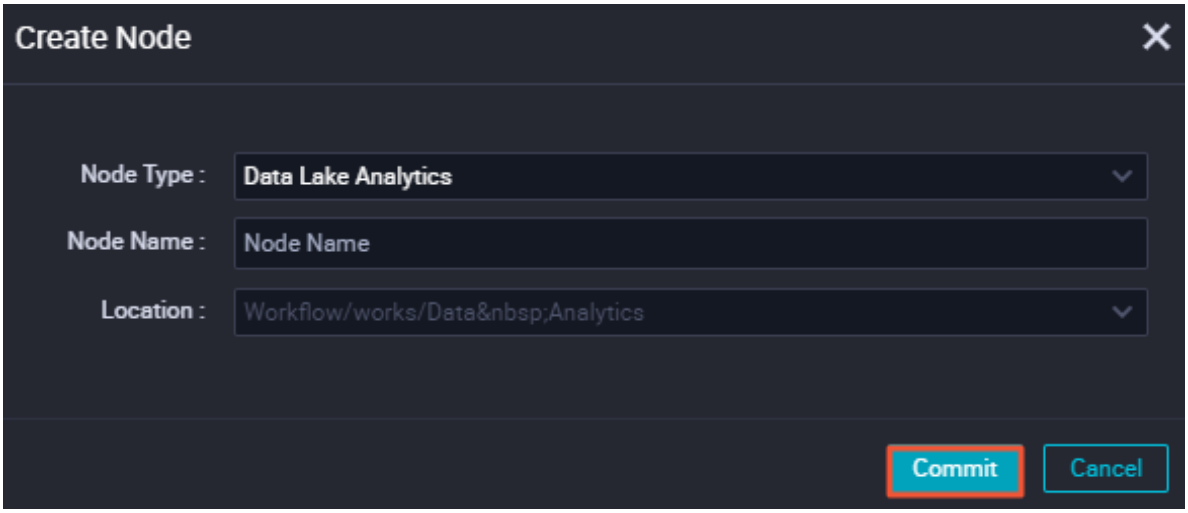
1. Go to the DataStudio page, and choose Create > Data Analytics > Data Lake Analytics.



Note:
You can also select a workflow, right-click Data Analytics, and then choose Create Data Analytics Node > Data Lake Analytics.



2. In the Create Node dialog box, enter the Node name, select the Destination folder, and then click Commit. The Location field is optional. You can specify this field to classify and manage nodes.

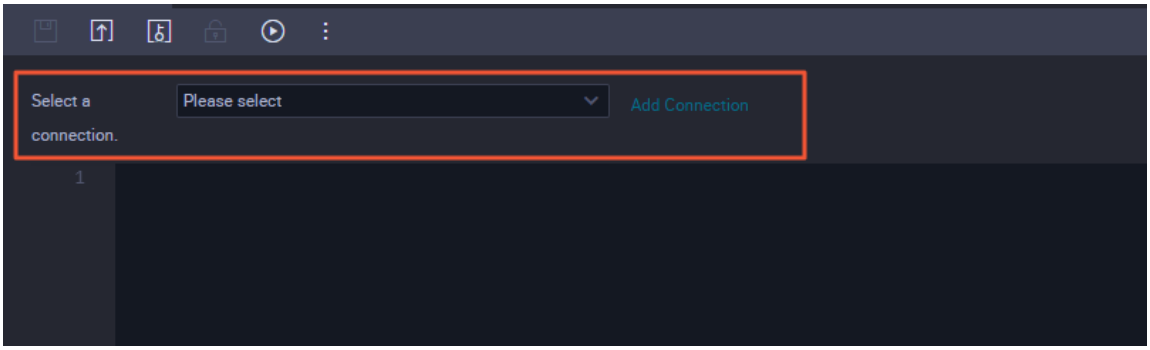


3. Edit the Data Lake Analytics node.

   You can select a connection and edit SQL code on the node editing tab.

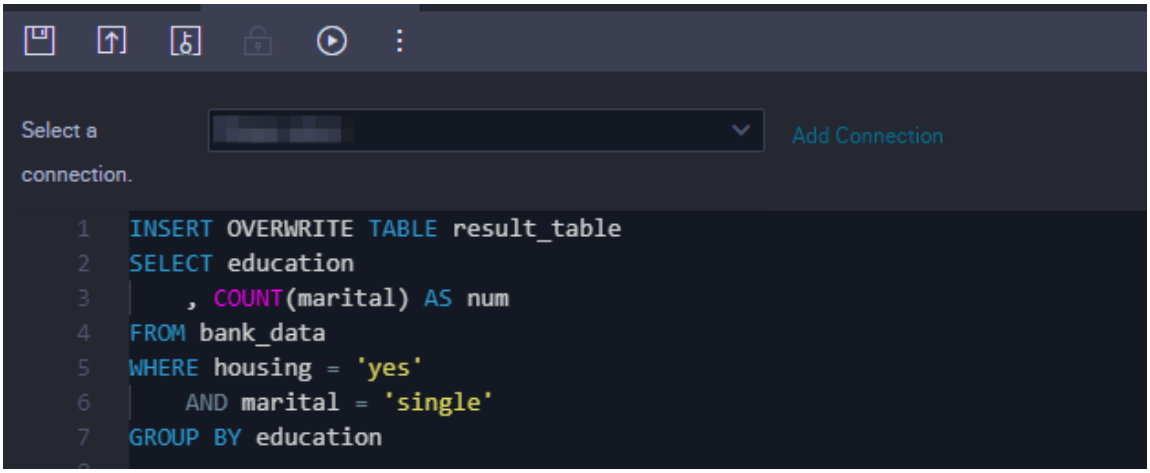
   a. Select a connection.

      Select a target connection for the node. If you cannot find the required connection in the drop-down list, click Add Connection to open the Add Connection page. You can add the connection on the Data Integration page. For more information, see Configure a connection.



   b. Edit SQL statements.

      After selecting a connection, you can write SQL statements based on the syntax supported by Data Lake Analytics. You can write DML and DDL statements in the code editor.



   c. Save and run the SQL statements.

      After you finish editing the SQL statements, click the Save button to save the settings of the node to the server. Then, click the Run button to run the SQL statements you have saved.

4. Set properties of the node.

   Click Properties on the right of the node editing tab to open the Properties tab. For more information, see Properties.

5. Commit the node.

   After you set the properties, click Save in the tool bar to commit the node to the development environment. After you commit the node to the development environment, the node is unlocked.

6. Deploy the node.

   For more information, see Deploy a node.

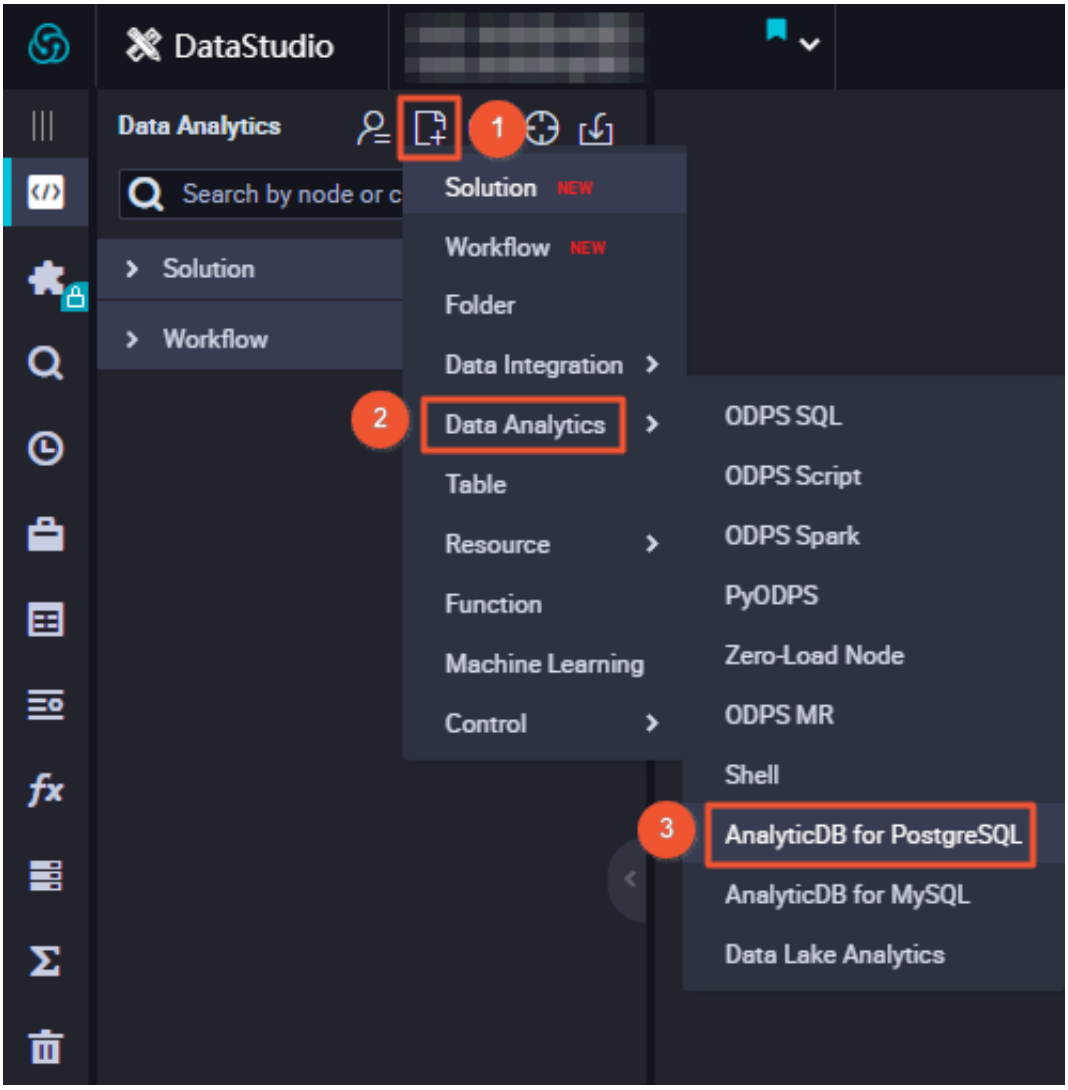
7. Test the node in the production environment.

   For more information, see #unique_32.

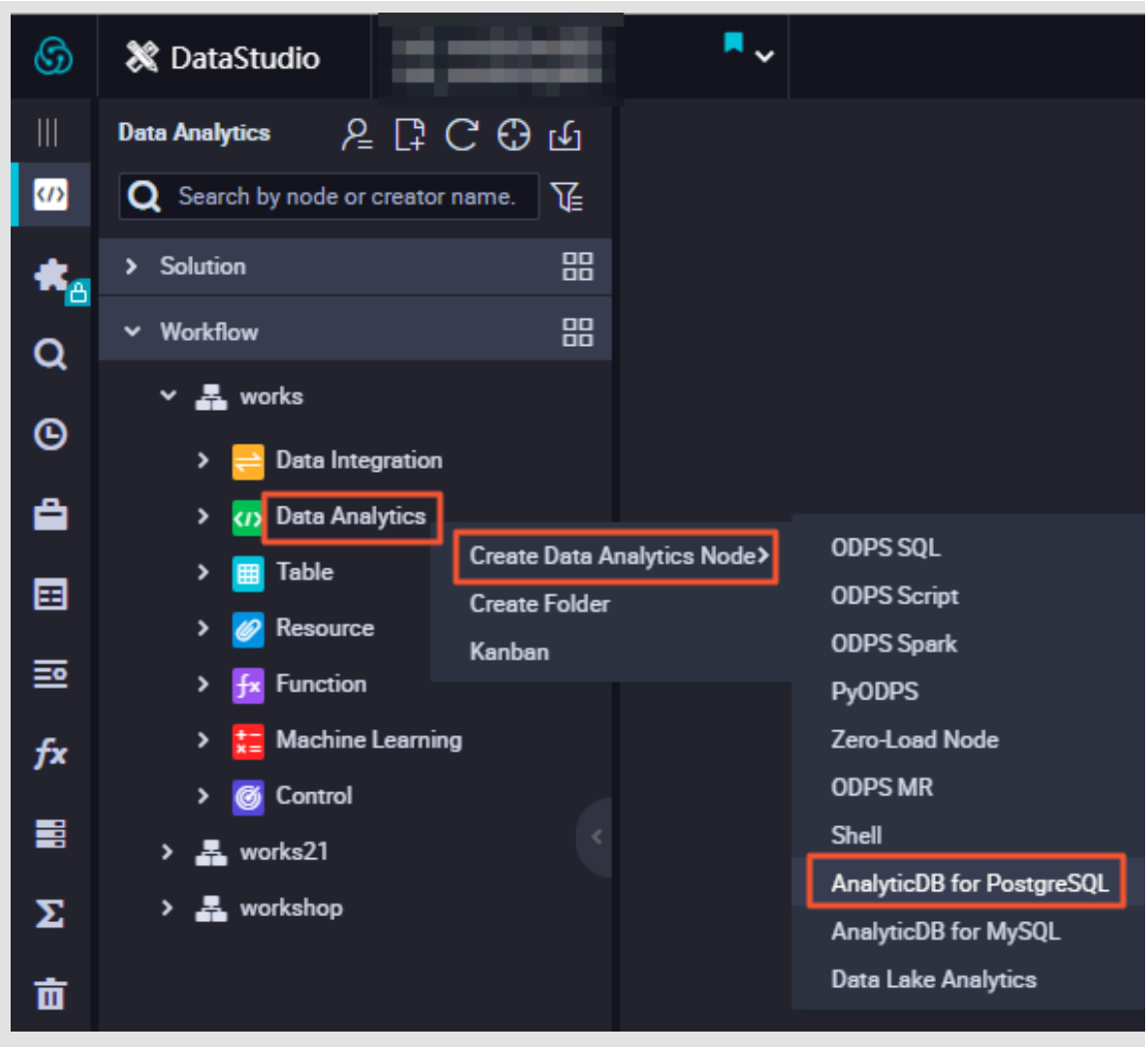
1.5.22 AnalyticDB for PostgreSQL node

You can create an AnalyticDB for PostgreSQL node in DataWorks to build an online ETL process.

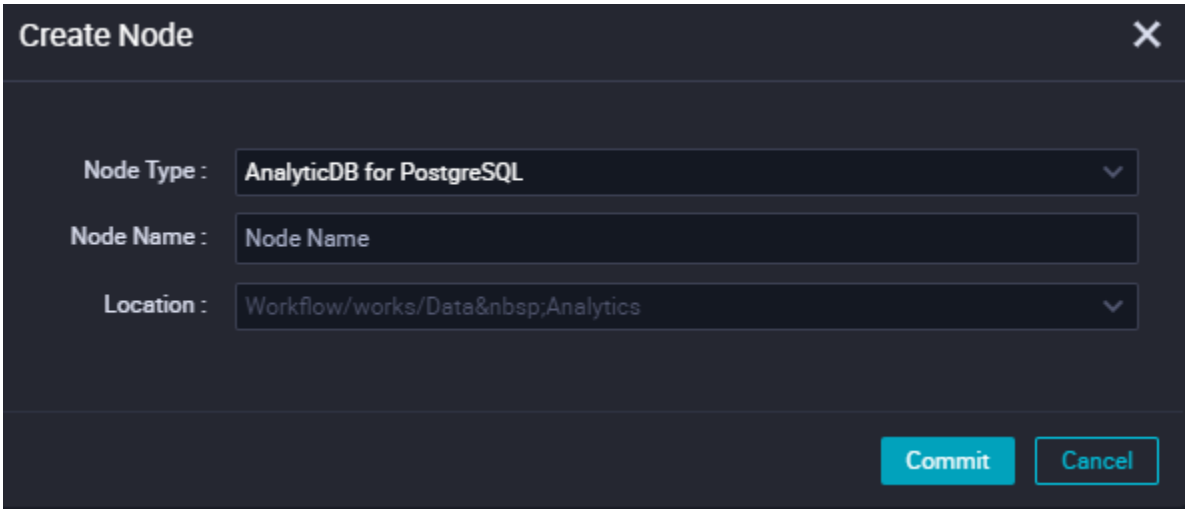
1. Go to the DataStudio page, and choose Create > Data Analytics > AnalyticDB for PostgreSQL.



Note:
You can also select a workflow, right-click Data Analytics, and then choose Create Data Analytics Node > AnalyticDB for PostgreSQL.



2. In the Create Node dialog box, enter the Node name, select the Destination folder, and then click Commit. The Location field is optional. You can specify this field to classify and manage nodes.

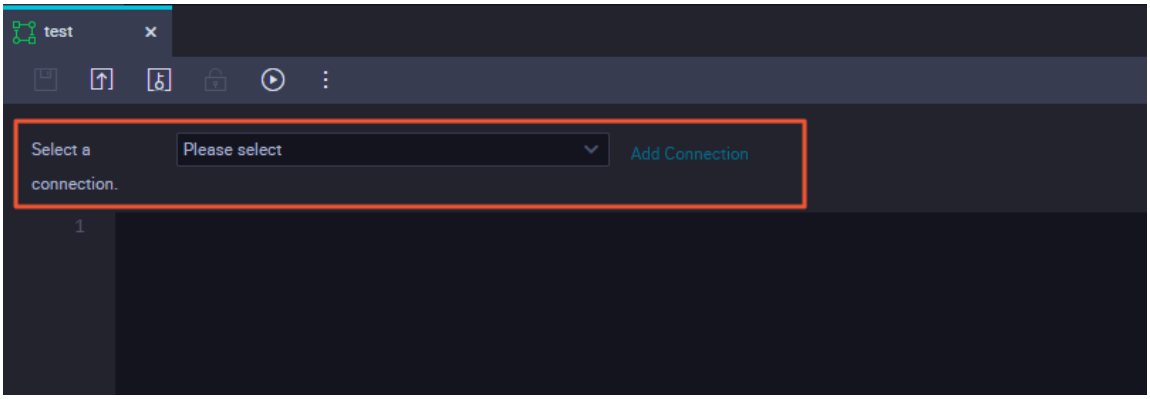


3. Edit the AnalyticDB for PostgreSQL node.

   You can select a connection and edit SQL code on the node editing tab.

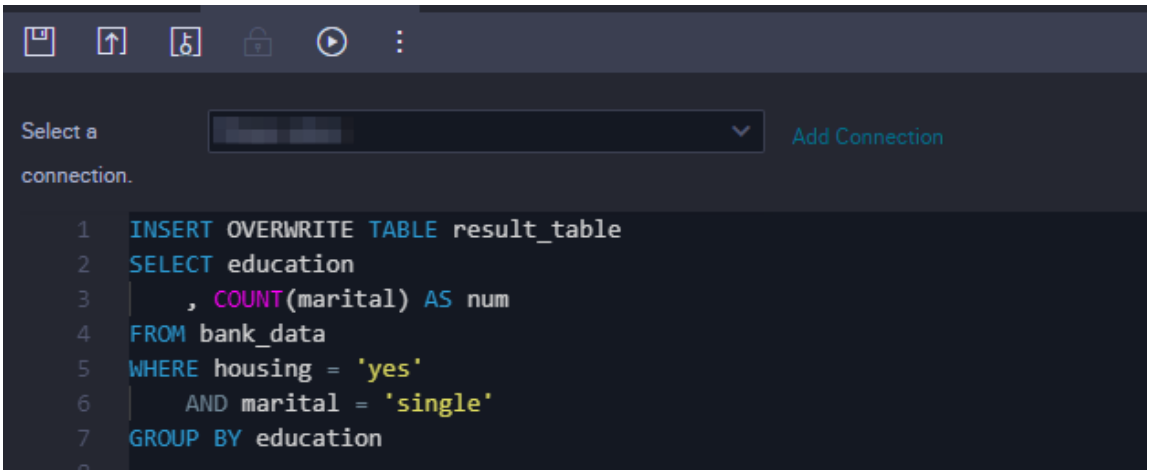
   a. Select a connection.

      Select a target connection for the node. If you cannot find the required connection in the drop-down list, click Add Connection to open the Add Connection page. You can add the connection on the Data Integration page. For more information, see Configure a connection.



   b. Edit SQL statements.

      After selecting a connection, you can write SQL statements based on the PostgreSQL syntax. You can write DML and DDL statements in the SQL code editor.

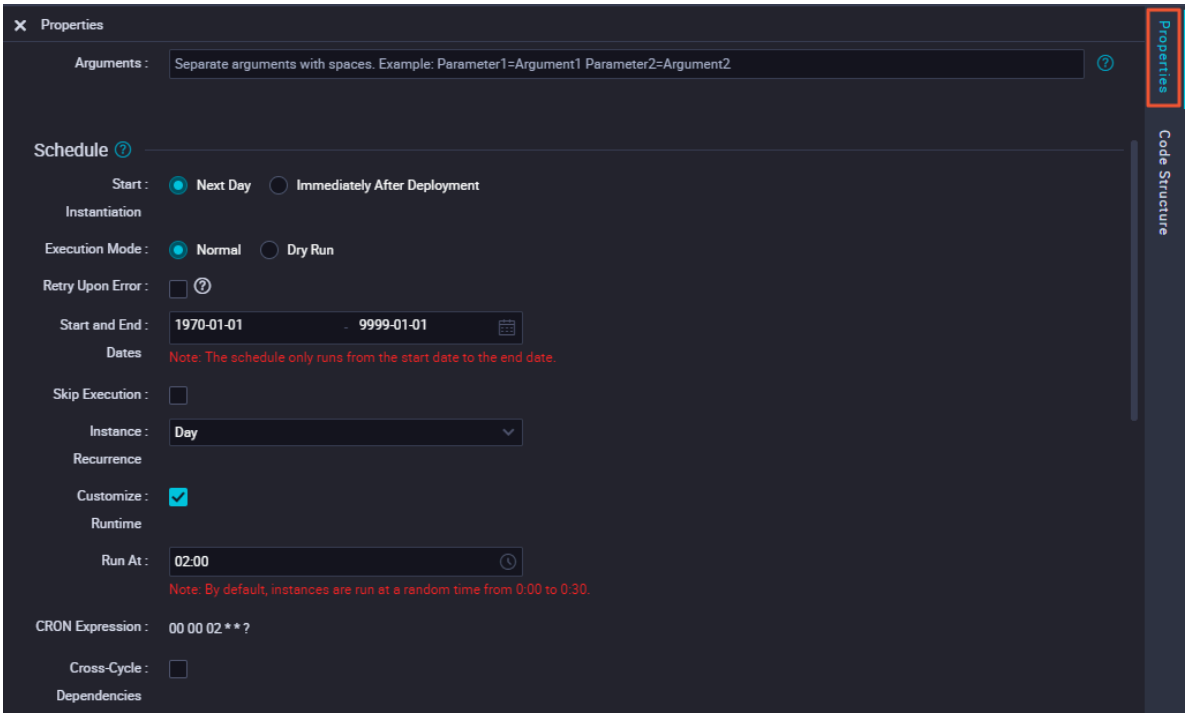


   c. Save and run the SQL statements.

      After you finish editing the SQL statements, click the Save button to save the settings of the node to the server. Then, click the Run button to run the SQL statements you have saved.

4. Set properties of the node.

   Click Properties on the right of the node editing tab to open the Properties tab. For more information, see Properties.



5. Commit the node.

   After you set the properties, click Save in the tool bar to commit the node to the development environment. After you commit the node to the development environment, the node is unlocked.

6. Deploy the node.

   For more information, see Deploy a node.

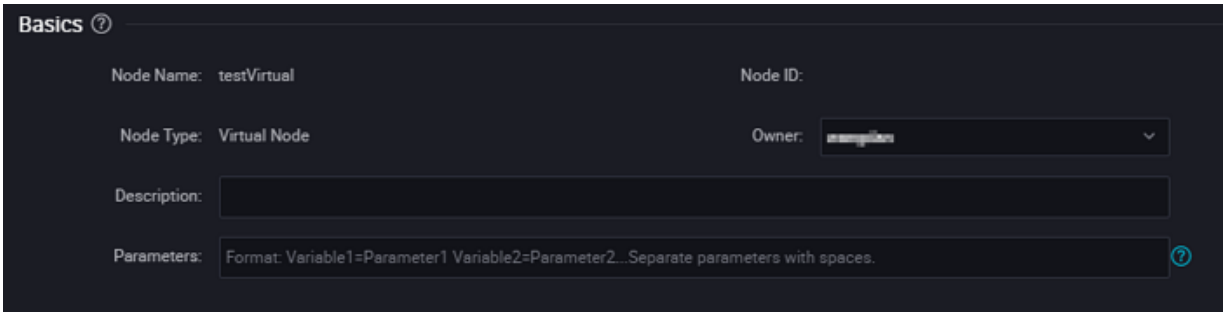
7. Test the node in the production environment.

   For more information, see #unique_32.

1.6 Scheduling configuration

1.6.1 Basic attributes

The figure below shows the basic attribute configuration interface:



- Node Name: The node name of the created workflow node. To modify the node name, right-click the node on the directory tree and choose Rename from the short-cut menu.
- Node ID: The unique node ID generated when a task is submitted and cannot be modified.
- Node Type: The node type that you select when creating a workflow node and cannot be modified.
- Owner: The node owner. By default, the owner of a newly created node is the current logon user. To modify the owner, click the input box, and enter the owner name or select another user.

  Note:
  When you select another user, the user must be a member of the current project.
- Description: Generally used to describe the business and node purpose.
- Parameter: A parameter used to assign value to a variable in the code during task scheduling.

  For example, when a variable "pt=${datetime}" is used to indicate the code time , you can assign a value to the variable here. The assigned value can use the scheduling built-in time parameter "datetime=$bizdate".

Parameter value assignment formats for various node types

- ODPS SQL, ODPS PL, ODPS MR types: `Variable name 1 = Parameter 1 Variable name 2 = Parameter 2 ...`, separate multiple parameters with spaces.
- SHELL type: `Parameter 1 Parameter 2 ...`, separate multiple parameters with spaces.

Some frequently-used time parameters are provided as built-in scheduling parameters. For more information about these parameters, see #unique_42.

1.6.2 Parameter configuration

In common data R&D scenarios, the node code of various types does not remain unchanged after compilation every time when it is called. Before calculation, You need to dynamically import values such as the date and time based on requirement changes and time changes to replace variable values.

DataWorks provides the parameter configuration feature to apply to such business scenarios. After configuring parameters, you can assign them to nodes that are automatically and periodically scheduled to parse the required values. Currently, parameters are divided into system parameters and custom parameters (recommended). This section describes the parameters in detail and uses examples to show how to operate them.

Parameter types

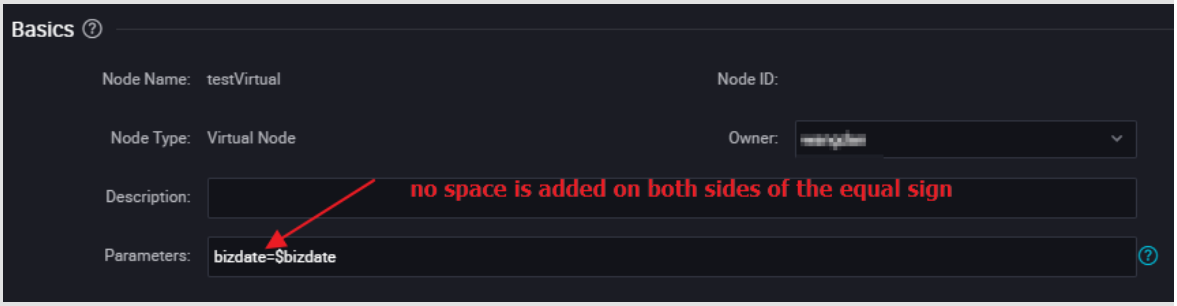
Parameter type	Call method	Applicable type	Parameter text box example
System parameters : including bdp. system.bizdate and bdp.system.cyctime	To use the system parameters in the scheduling system, directly reference ${bdp. system.bizdate} and ${bdp.system. cyctime} in the code without having to set them in the Parameter text box. The system automatically replaces the values of the parameters.	All node types	None

Parameter type	Call method	Applicable type	Parameter text box example
Non-system parameters: custom parameters (recommended)	Reference ${key1}, ${key2} in the code and enter "key1= value1 key2=value2 " in the Parameter text box.	Non-SHELL nodes	· Constant parameters: param1="abc" param2=1234 · Variables: param1=$[yyyymmdd], which is calculated based on the value of bdp.system. cyctime
	Reference $1 $2 $3 in the code and enter "value1 value2 value3" in the Parameter text box.	Shell nodes	Constant parameters: "abc" 1234 Variables: $[yyyymmdd], which is calculated based on the value of bdp. system.cyctime.

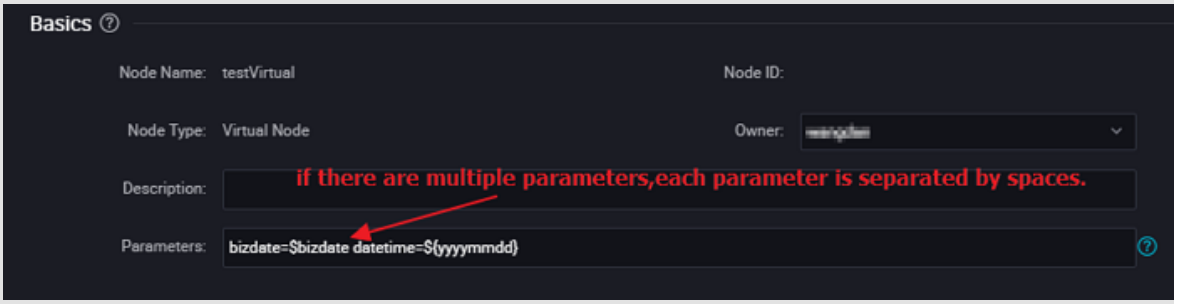
As described in the preceding table, The variable values are based on the values of system parameters. You can use custom variables to flexibly define the obtained part and format.

Note:
Choose Schedule > Basic Information and assign values to non-system parameters in the Parameter text box (assign values to scheduling variables). Note the following when configuring parameters:

- No space can be added on either side of the equation mark (=) of a parameter.
  Correct example: `bizdate =$ bizdate`



- Multiple parameters (if any) must be separated with spaces.



System parameters

DataWorks provides two system parameters, which are defined as follows:

- ${bdp.system.cyctime}: The scheduled runtime of an instance. Default format: yyyymmddhh24miss.
- ${bdp.system.bizdate}: The business date on which an instance is calculated, in the default format of yyyymmdd. The default business date is one day before the scheduled runtime.

According to the definitions, the formula for calculating the scheduled runtime and business date is as follows: `Scheduled runtime = Business date + 1`.

To use the system parameters, directly reference ${bizdate} in the code without having to set them in the Parameter text box. The system automatically replaces the fields that reference this system parameter in the code.

Note:
The scheduling attributes of a periodic node are configured to define the scheduling rules of the runtime. Therefore, you can calculate the business date based on the scheduled runtime of an instance and obtain the values of system parameters for the instance.

The scheduling parameter configuration of a PyODPS node is slightly different from that of a common node. For more information, see #unique_26.

Example of system parameters

Set a MaxCompute SQL node to be run once every hour from 00:00 to 23:59 every day. To use system parameters in the code, follow these steps:

1. Directly reference system parameters in the code. The node code is as follows:

```
insert   overwrite   table   tb1   partition ( ds  =' 20150304 ')
select
c1 , c2 , c3
from  (
select  *  from   tb2
where   ds  ='${ bdp . system . cyctime }')  t
full   outer   join (
select  *  from   tb3
where   ds  = '${ bdp . system . bizdate }')  y
on   t . c1  =  y . c1 ;
```

2. After the preceding step, your node is partitioned by using the system parameters. Set the scheduling time attributes and scheduling dependencies. In this example, Recurrence is set to Hour.
3. After setting the scheduling cycle and dependency, submit the node. You can check the node in O&M. The node generates periodic instances during running from the second day. You can right-click an instance and select View Log to view the time when the system parameters are parsed.

   For example, the scheduling system creates 24 running instances for the node on January 14, 2019. Because the business date is January 13, 2019 (the day before the running date) for all instances, ${bdp.system.bizdate} is always displayed as 20190113. The runtime is the running date plus the scheduled time. Therefore, ${

bdp.system.cyctime} is displayed as 20190114000000 plus the scheduled time of each instance.

Open the run logs of each instance and search for the replaced values of parameters in the code:

- Because the scheduled time for the first instance is 2019-01-14 00:00:00, bdp.system.bizdate is replaced with 20190113, and bdp.system.cyctime is replaced with 20190114000000.
- Because the scheduled time for the second instance is 2019-01-14 01:00:00, bdp.system.bizdate is replaced with 20190113, and bdp.system.cyctime is replaced with 20190114010000.
- Similarly, because the scheduled time for the twenty-fourth instance is 2019-01-14 23:00:00, bdp.system.bizdate is replaced with 20190113, and bdp.system.cyctime is replaced with 20190114230000.

Custom parameters for non-SHELL nodes

To configure a scheduling parameter for a non-SHELL node, add ${Variable name} in the code to reference the function, and then assign a value to the scheduling parameter.

 Note:
The name of a variable in the SQL code can contain only lowercase letters (a–z), uppercase letters (A–Z), digits, and underscores (_). If the variable name is `date`, the value of $bizdate is automatically assigned to this variable. (For more information, see the list of built-in scheduling parameters). You do not need to assign a value during scheduling parameter configuration. Even if another value is assigned, it is not used in the code because the value of $bizdate is automatically assigned in the code by default.

Example of custom parameters for non-SHELL nodes

Set a MaxCompute SQL node to run once every hour from 00:00 to 23:59 every day. To use the custom variables thishour and lasthour in the code, follow these steps:

1. Reference the parameters in the code as follows:

```
insert   overwrite   table   tb1   partition ( ds  =' 20150304 ')
select
 c1 , c2 , c3
from  (
 select  *  from   tb2
```

```
 where   ds  ='${ thishour }')  t
full   outer   join (
 select  *  from   tb3
 where   ds  = '${ lasthour }')  y
on   t . c1  =  y . c1 ;
```

2. Choose Schedule > Basic Information and assign values to the variables referenced in the code in the Parameter text box.

   Configure the custom parameters as follows:

   - thishour=$[yyyy-mm-dd/hh24:mi:ss]
   - lasthour=$[yyyy-mm-dd/hh24:mi:ss-1/24]

   > Note:
   > The value of yyyy-mm-dd/hh24:mi:ss corresponds to that of cyctime. For more information, see Custom variables.

   You can enter `thishour =$[ yyyy - mm - dd / hh24 : mi : ss ]  lasthour =$[ yyyy - mm - dd / hh24 : mi : ss - 1 / 24 ]` in the Parameter text box.

3. Set the node to be run once every hour.
4. After setting the scheduling cycle and dependency, submit the node. You can check the node in O&M. The node generates periodic instances during running from the second day. You can right-click an instance and select View Log to view the time when the custom parameters are parsed. Because the value of cyctime is 20190114010000, the value of thishour is 2019-01-14/01:00:00, and the value of lasthour, which represents the last hour, is 2019-01-14/00:00:00.

Custom parameters for SHELL nodes

The parameter configuration procedure of a SHELL node is similar to that of a non-SHELL node except that the rules are different. For a SHELL node, variable names cannot be customized. Instead, they must be named $1, $2, $3, and so on. For example, for a SHELL node, the SHELL syntax declaration in the code is: $1 is configured as $xxx (built-in scheduling parameter) during scheduling parameter configuration for the node. That is, the value of $xxx is used to replace $1 in the code.

> Note:
> For a SHELL node, when the number of parameters reaches 10, use ${10} to declare the variable.

Example of custom parameters for SHELL nodes

Set a SHELL node to be run once at 01:00 each day. To use the custom constant parameter myname and the custom variable parameter ct in the code, follow these steps:

1. Reference the parameters in the code as follows:

```
echo " hello $ 1 , two days ago is $ 2 , the system
param is ${ bdp . system . cyctime }";
```

2. Choose Schedule > Basic Information and assign values to the variables referenced in the code in the Parameter text box. Value assignment rule: parameter 1 parameter 2 parameter 3 ... (Replaced variables are parsed based on the parameter location, for example, $1 is replaced with the value of parameter 1). In this example, set $1 and $2 to abcd and $[yyyy-mm-dd-2], respectively.
3. Set the node to be run once at 01:00 every day.
4. After setting the scheduling cycle and dependency, submit the node. You can check the node in O&M. The node generates periodic instances during running from the second day. Right-click an instance and select View Log. The logs show that $1 in the code is replaced with constant abcd, $2 is replaced with 2019-01-12 (two days before the running date), and ${bdp.system.cyctime} is replaced with 20190114010000.

Custom variables

A custom variable can be a constant parameter or a built-in scheduling parameter.

Variable value being a constant value

For example, an SQL node includes the variable ${Variable name} in the code. Configure the parameter item Variable name='Fixed value' for the node.

Code: select xxxxxx type='${type}'

Value assigned to the scheduling variable: type='aaa'

When the node is being scheduled, the variable in the code is replaced with type='aaa'.

Variable value being a variable

Variables are built-in scheduling parameters whose values depend on the system parameters ${bdp.system.bizdate} and ${bdp.system.cyctime}.

For example, an SQL node includes the variable ${Variable name} in the code. Configure the parameter item Variable name=Scheduling parameter for the node.

Code: select xxxxxx dt=${datetime}

Value assigned to the scheduling variable: datetime=$bizdate

When the node is being scheduled on July 22, 2017, the variable in the code is replaced with dt=20170721.

List of variables

$bizdate: The business date in the format of yyyymmdd. Note: For daily scheduling, this parameter is set to the day before the current date by default.

For example, the code of a MaxCompute SQL node includes pt=${datetime}, and the parameter configured for the node is datetime=$bizdate. When the node is run on July 22, 2017, $bizdate is replaced with pt=20170721.

$cyctime: The time at which the node is scheduled to run. If no scheduling time is configured for a daily node, cyctime is set to 00:00 of the current day. The time is accurate to the second. This parameter is usually used for nodes that are scheduled by hour or minute. Example: cyctime=$cyctime.

 Note:
The time parameters configured by using $[] and ${} are different. $bizdate indicates the business date that is one day before the current date by default. $cyctime indicates the time at which the node is scheduled to run. If no scheduling time is configured for a daily node, cyctime is set to 00:00 of the current day. The time is accurate to the second. This parameter is usually used for nodes that are scheduled by hour or minute. For example, if the node is scheduled to run at 00:30 on the current day, $cyctime is yyyy-mm-dd 00:30:00. A {} parameter is involved in the computation with bizdate as the benchmark. During data patching, the parameter value is replaced with the selected business date. A [] parameter is involved in the computation with cyctime as the benchmark, which is calculated in the same way as the time in Oracle. During data patching, the parameter value is replaced with the selected business date plus one day. For example, if the date 20140510 is selected as the business date, cyctime is replaced with 20140511.

Examples of $cyctime: (Assume that $cyctime=20140515103000)

- $[yyyy] = 2014, $[yy] = 14, $[mm] = 05, $[dd] = 15, $[yyyy-mm-dd] = 2014-05-15, $[hh24:mi:ss] = 10:30:00, $[yyyy-mm-dd hh24:mi:ss] = 2014-05-1510:30:00
- $[hh24:mi:ss - 1/24] = 09:30:00

- $[yyyy-mm-dd hh24:mi:ss -1/24/60] = 2014-05-1510:29:00
- $[yyyy-mm-dd hh24:mi:ss -1/24] = 2014-05-15 09:30:00
- $[add_months(yyyymmdd,-1)] = 20140415
- $[add_months(yyyymmdd,-12*1)] = 20130515
- $[hh24] =10
- $[mi] =30

Method for testing $cyctime:

After the instance start to run, right-click the node and select View Node Attribute. Check whether the scheduling time is the time at which the instance is run periodically.

The value of the parameter in Running Parameter is replaced with the scheduling time minus 1 hour.

$jobid: The ID of the workflow to which a node belongs. Example: jobid=$jobid.

$nodeid: The ID of a node. Example: nodeid=$nodeid.

$taskid: The ID of a node, that is, the ID of a node instance. Example: taskid=$taskid.

$bizmonth: The business month in the format of yyyymm.

- Note: If the month of a business date is equal to the current month, the value of $bizmonth is the month of the business date minus 1. Otherwise, the value of $ bizmonth is the month of the business date.
- For example, the code of a MaxCompute SQL node includes pt=${datetime}, and the parameter configured for the node is datetime=$bizmonth. When the node is run on July 22, 2017, $bizmonth is replaced with pt=201706.

$gmtdate: The current date in the format of yyyymmdd. The value of this parameter is the current date by default. During data patching, the value of gmtdate that is imported is the business date plus 1.

For example, the code of a MaxCompute SQL node includes pt=${datetime}, and the parameter configured for the node is datetime=$gmtdate. When the node is run on July 22, 2017, $gmtdate is replaced with pt=20170722.

${...} custom parameter

- You can customize a time format based on $bizdate, where yyyy indicates the four-digit year, yy indicates the two-digit year, mm indicates the month, and dd

indicates the day. You can use any combination of these parameters, for example, ${yyyy}, ${yyyymm}, ${yyyymmdd}, and ${yyyy-mm-dd}.

- $bizdate is accurate to the day. Therefore, ${...} can only represent the year, month , or day.
- Methods for obtaining the period plus or minus a certain duration:

  Next N years: ${yyyy+N}

  Previous N years: ${yyyy-N}

  Next N months: ${yyyymm+N}

  Previous N months: ${yyyymm-N}

  Next N weeks: ${yyyymmdd+7*N}

  Previous N weeks: ${yyyymmdd-7*N}

  Next N days: ${yyyymmdd+N}

  Previous N days: ${yyyymmdd-N}

${yyyymmdd}: The business date in the format of yyyymmdd. The value of this parameter is the same as that of $bizdate, and the parameter supports various separators, for example, yyyy-mm-dd.

- Note: For daily scheduling, this parameter is set to the day before the current date by default. You can customize a time format for this parameter, for example, yyyy-mm-dd for ${yyyy-mm-dd}.

- Examples:
  - The code of a MaxCompute SQL node includes pt=${datetime}, and the parameter configured for the node is datetime=${yyyy-mm-dd}. When the node is run on July 22, 2018, ${yyyy-mm-dd} is replaced with pt=2018-07-21.
  - The code of a MaxCompute SQL node includes pt=${datetime}, and the parameter configured for the node is datetime=${yyyymmdd-2}. When the node is run on July 22, 2018, ${yyyymmdd-2} is replaced with pt=20180719.
  - The code of a MaxCompute SQL node includes pt=${datetime}, and the parameter configured for the node is datetime=${yyyymm-2}. When the node is run on July 22, 2018, ${yyyymmdd-2} is replaced with pt=201805.
  - The code of a MaxCompute SQL node includes pt=${datetime}, and the parameter configured for the node is datetime=${yyyy-2}. When the node is run on July 22, 2018, ${yyyy-2} is replaced with pt=2016.
  - You can assign values to multiple parameters during MaxCompute SQL node configuration, for example, startdatetime=$bizdate enddatetime=${yyyymmdd+1} starttime=${yyyy-mm-dd} endtime=${yyyy-mm-dd+1}.

FAQs

- Q: The table partition format is pt=yyyy-mm-dd hh24:mi:ss, but spaces are not allowed in scheduling parameters. How can I configure the format of $[yyyy-mm-dd hh24:mi:ss]?

  A: Use the custom variables datetime=$[yyyy-mm-dd] and hour=$[hh24:mi:ss] to obtain the date and time, respectively. Then, join them together to form pt="${datetime} ${hour}" in the code. (Separate the two custom variables with a space).
- Q: The table partition is pt="${datetime} ${hour}" in the code. To obtain the data for the last hour when the node is run, the custom variables datetime=$[yyyymmdd] and hour=$[hh24-1/24] can be used to obtain the date and time, respectively. However, for an instance running at 00:00, the calculation result is 23:00 of the current day, instead of 23:00 of the previous day. What measures can I take in this case?

  A: Modify the formula of datetime to $[yyyymmdd-1/24] and keep the formula of hour unchanged at $[hh24-1/24]. The calculation result is as follows:
  - For an instance that is scheduled to run at 2015-10-27 00:00:00, the values of $[yyyymmdd-1/24] and $[hh24-1/24] are 20151026 and 23, respectively. This

is because the scheduling time minus 1 hour is a time value that belongs to yesterday.

- For an instance that is scheduled to run at 2015-10-27 01:00:00, the values of $[yyyymmdd-1/24] and $[hh24-1/24] are 20151027 and 00, respectively. This is because the scheduling time minus 1 hour is a time value that belongs to the current day.

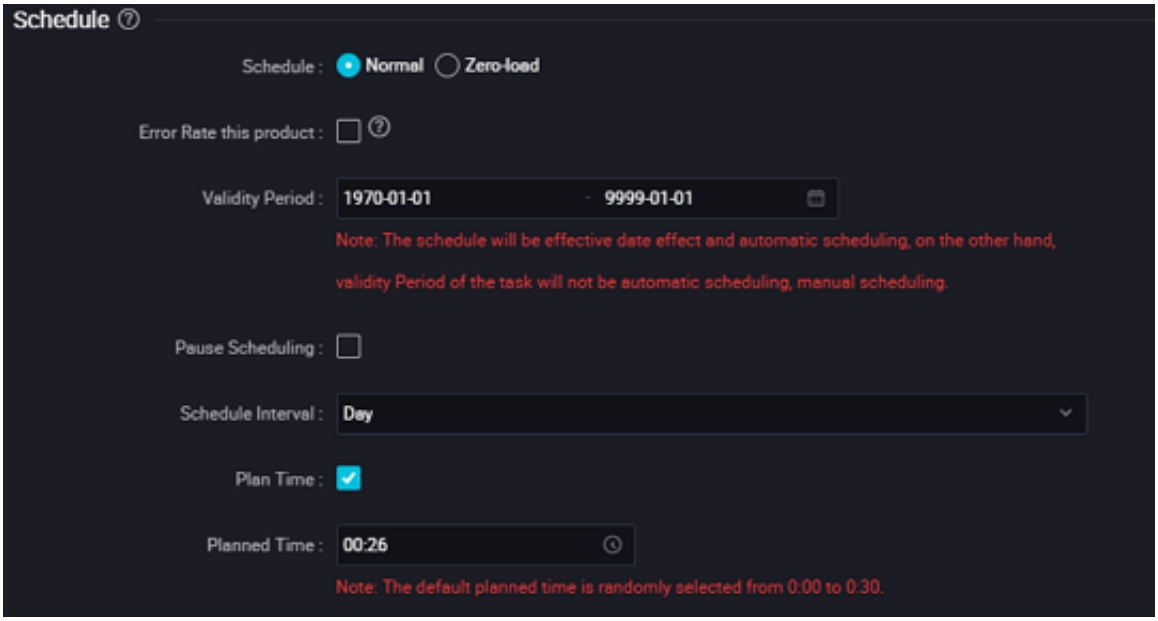
DataWorks offers four node execution modes.

- Running on DataStudio: You must assign a temporary value on the parameter configuration page to ensure proper running. The configurations are not saved as node attributes and do not take effect in the other three execution modes.
- Automatic running at an interval: No configuration is needed in the Parameter text box. The scheduling system automatically replaces the parameters with the scheduled runtime of the current instance.
- Running triggered by testing or data patching: You must specify a business date when the run is triggered. The scheduled runtime is derived from the formula described earlier to get the two system parameter values of each instance.

1.6.3 Scheduling time

This section describes how to set the scheduling time of nodes, including the scheduling cycle and dependencies. You can also specify whether a node depends on the instance of the last cycle.

The following figure shows the scheduling time configuration page.



Instance creation modes

- Next Day: If you select this option, instances are generated in full mode. (Nodes published before 22:00 create instances the next day, while nodes published after 22:00 create instances the third day.)
- Immediately After Publishing: If you select this option, instances are immediately generated after nodes are published.

Node status

- Normal: If you select this option, a node is scheduled and run normally based on the following scheduling cycle configuration. It is the default option for nodes.
- Zero-load: If you select this option, a node is scheduled based on the following scheduling cycle configuration. However, once this node is scheduled, the system does not actually run the node but directly returns a success response.
- Error Retry: If you select this check box, a node is rerun when it encounters an error. By default, a node can be automatically rerun for a maximum of three times with an interval of 2 minutes.
- Pause Scheduling: If you select this check box, a node is scheduled based on the following scheduling cycle configuration. However, once this node is scheduled, the system does not actually run the node but directly returns a failure response. It is used when a node is suspended but will be run later.

Scheduling cycle

In DataWorks, after a node is submitted, the underlying scheduling system generates an instance every day from the next day based on the scheduling time of the node, and runs the instances based on the running results and time points of its ancestor instances. Nodes that are submitted after 23:30 create instances from the third day.

 Note:

If you schedule a node to run on every Monday, the node is run only on Mondays. On the other days, once this node is scheduled, the system does not actually run the node but directly returns a success response. Therefore, you need to set the business date to one day earlier than the runtime for weekly scheduled nodes during testing or data patching.

For a node that runs cyclically, the priority of its dependency is higher than that of its scheduling time. That is, when the scheduling time is reached, the node instance

is not run immediately but first checks whether all the ancestor instances have been run.

- If not all the ancestor instances have been run but the scheduling time is reached, the instance remains in the Not Running status.
- If all the ancestor instances have been run but the scheduling time is not reached, the instance enters the Waiting for Scheduled Time status.
- If all the ancestor instances have been run and the scheduling time is reached, the instance enters the Waiting for Resource status.

Dependency on the last-cycle instance

Before specifying whether a node depends on instance of the last cycle, you must understand the following concepts:

- Last-cycle instance: indicates the instance generated on the last calendar day. Assume that the current day is August 8, 2018. The instance generated on August 7, 2018 is called the last-cycle instance.
- Dependency on the last-cycle instance: indicates that a node depends on the last-cycle instance of its parent node. Assume that you have configured daily scheduling nodes A and B. If you want the instance of node B to be run only after that of node A generated on the last day is run, you can configure a cross-cycle dependency. That is, you can configure node B to make it depend on the last-cycle instance of node A.

The following figure shows how to configure the dependency on the last-cycle instance.

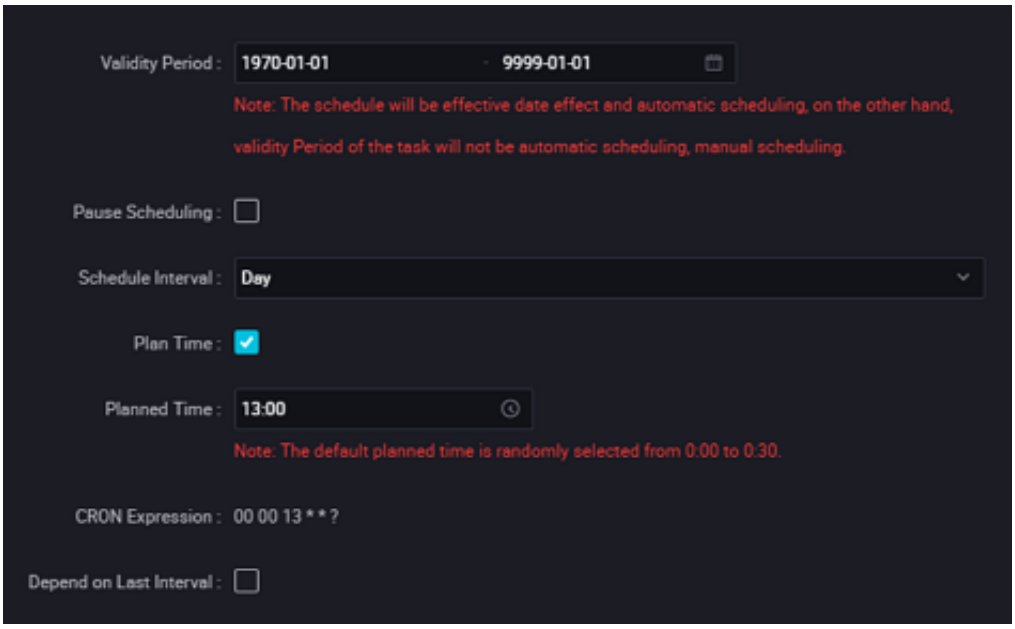
You can select any of the following options for Dependent Node:

- Level 1 Child Node: indicates that the current node depends on the last-cycle instances of its descendant nodes. For example, node A has descendant nodes B, C, and D. If you select this option, node A depends on the last-cycle instances of nodes B, C, and D.
- Current Node: indicates that the current node depends on its own last-cycle instance.
- Customize: indicates that the current node depends on the last-cycle instance of a specified node. You need to enter the ID of the specified node. If multiple nodes exist, separate their IDs with commas (,), for example, 12345,23456.

Scheduling by day

Nodes scheduled by day are run automatically once every day. When you create a periodic node, the node is set to run at 00:00 every day by default. You can specify another runtime as needed. For example, you can specify the runtime to 13:00 every day, as shown in the following figure.

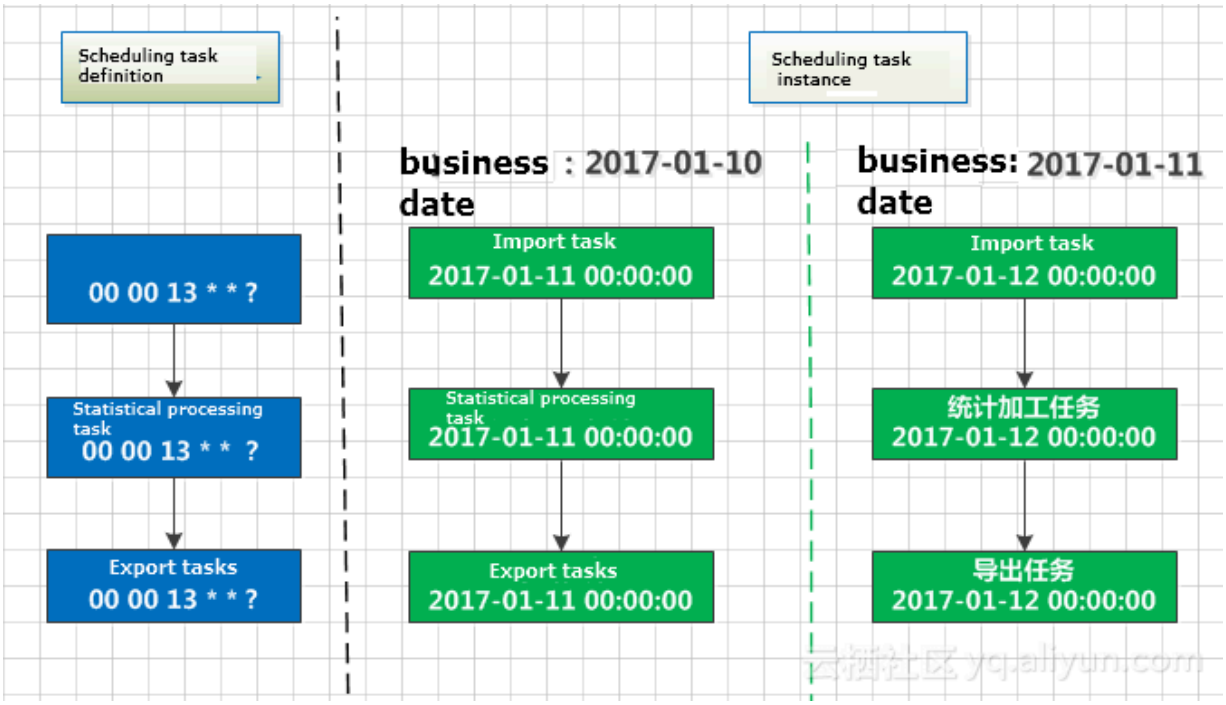
1. If Specify Time is cleared, the scheduled date of the daily node is the date of the current day in the YYYY-MM-DD format and the scheduling time of the node is randomly generated between 00:00 and 00:30.
2. If Specify Time is selected, the scheduling time of the daily node is a specified time of the current day in the YYYY-MM-DD HH:MM format. A scheduled node can be run only after its ancestor node is run and the scheduling time is reached. The node cannot be run if either one of the conditions is not met. The conditions are in no particular order.



Scenarios:

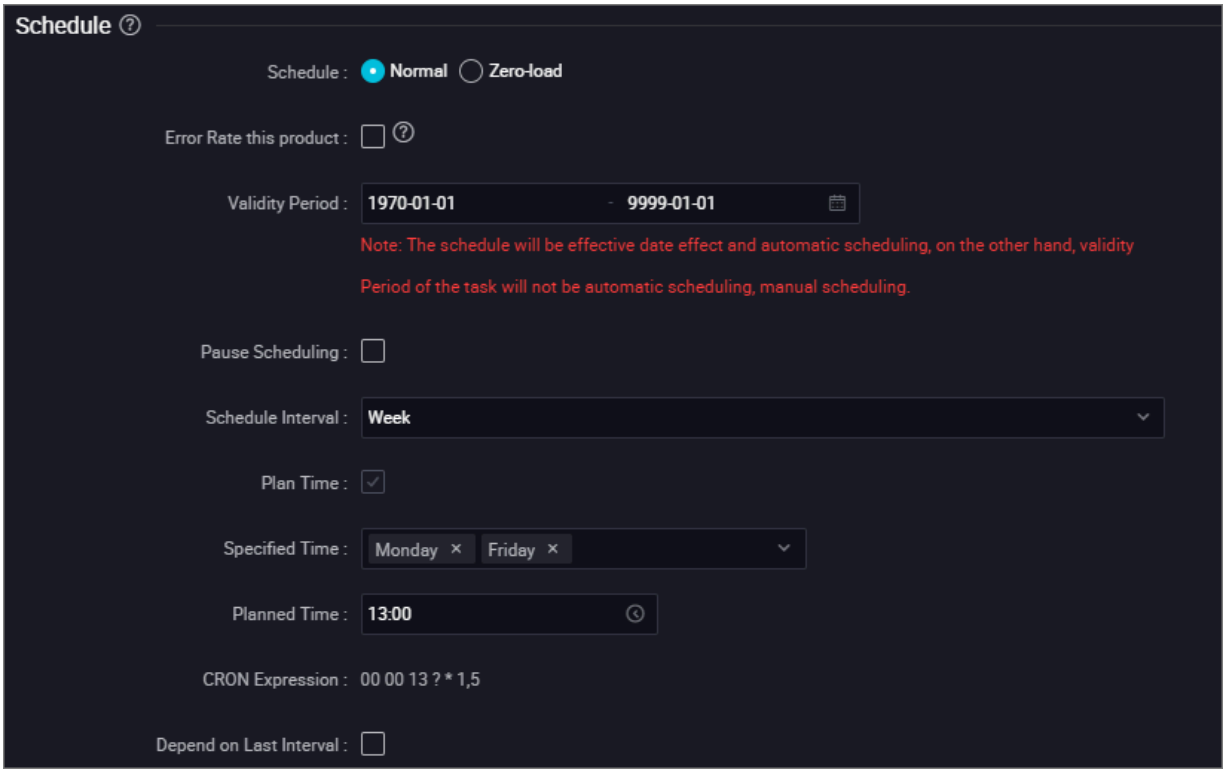
Import, statistical processing, and export nodes are all daily nodes with the runtime of 13:00, as shown in the preceding figure. Statistical processing nodes depend on import nodes, and export nodes depend on statistical processing nodes. The following figure shows the configuration of their dependencies. (When configuring the dependencies for a statistical processing node, set its ancestor node to an import node).

Based on the configuration in the preceding figure, the scheduling system automatically generates instances for the nodes and runs them as follows:



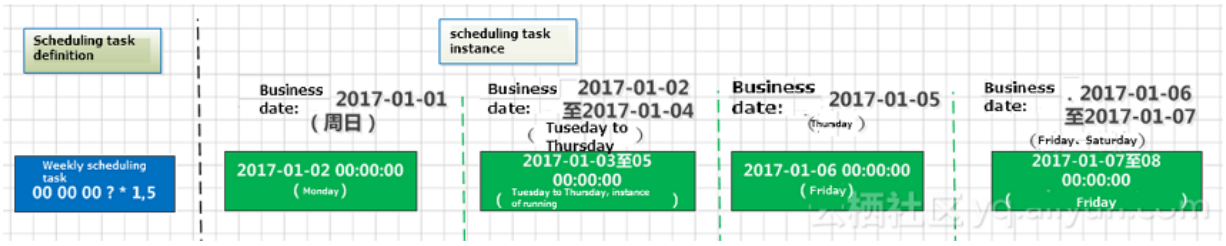
Scheduling by week

Nodes scheduled by week are automatically run at specific time points of specific days each week. On the other days, the system still generates instances to ensure the proper running of descendant instances. However, once a node is scheduled, the system does not actually run any logic or consume any resources but directly returns a success response.



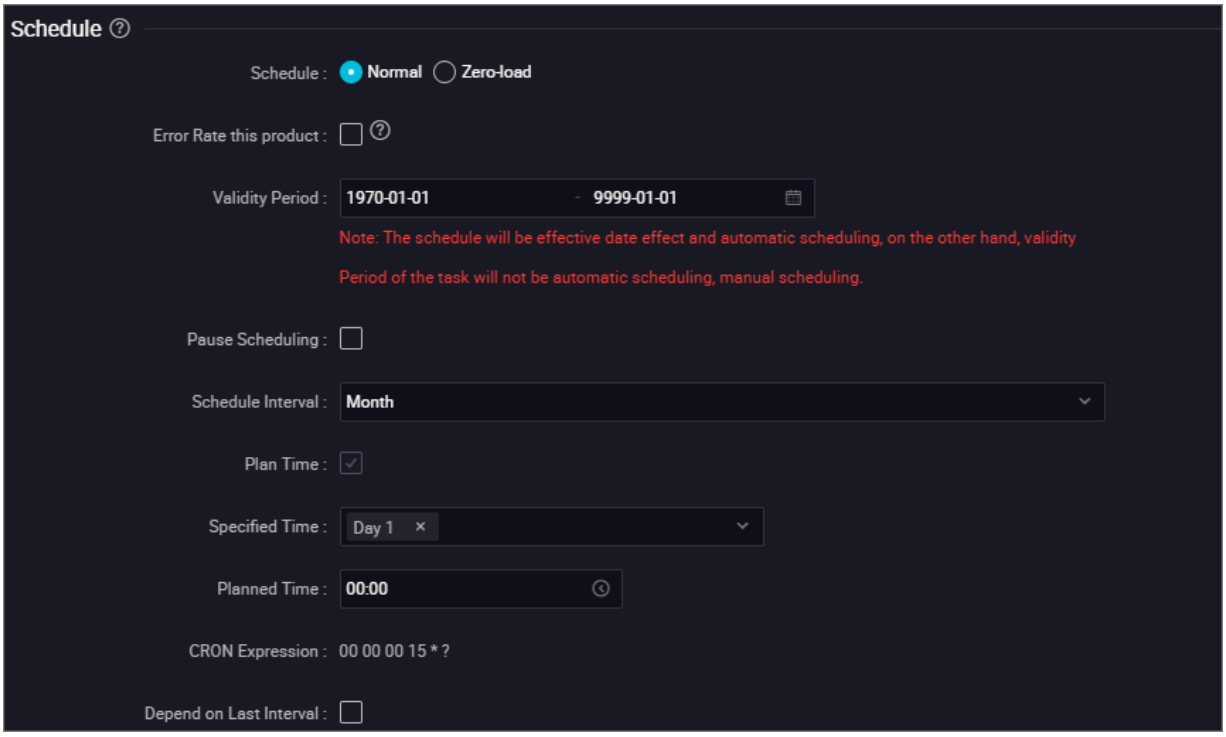
As shown in the preceding figure, the system schedules instances on every Monday and Friday, but returns success responses without scheduling instances on every Tuesday, Wednesday, Thursday, Saturday, and Sunday.

Based on the configuration in the preceding figure, the scheduling system automatically generates instances for the nodes and runs them as follows:



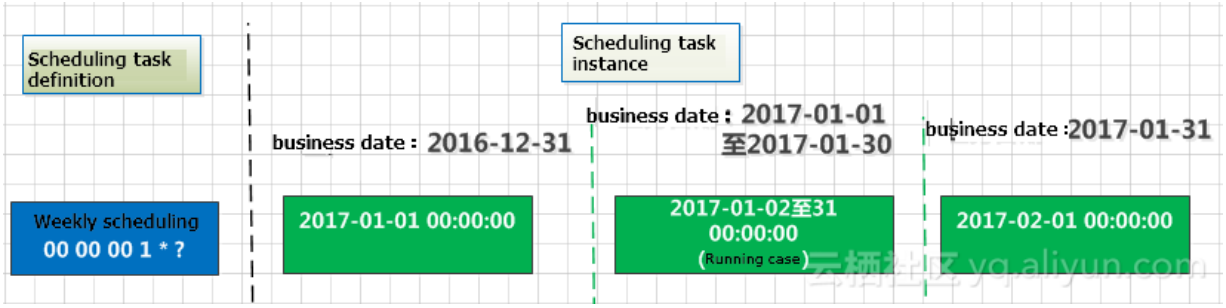
Scheduling by month

Nodes scheduled by month are automatically run at specific time points of specific days each month. On the other days, the system still generates instances to ensure the proper running of descendant instances. However, once a node is scheduled, the system does not actually run any logic or consume any resources but directly returns a success response.



As shown in the preceding figure, the system schedules instances on the first day of each month, but returns success responses without scheduling instances for the rest days of the month.

Based on the configuration in the preceding figure, the scheduling system automatically generates instances for the nodes and runs them as follows:

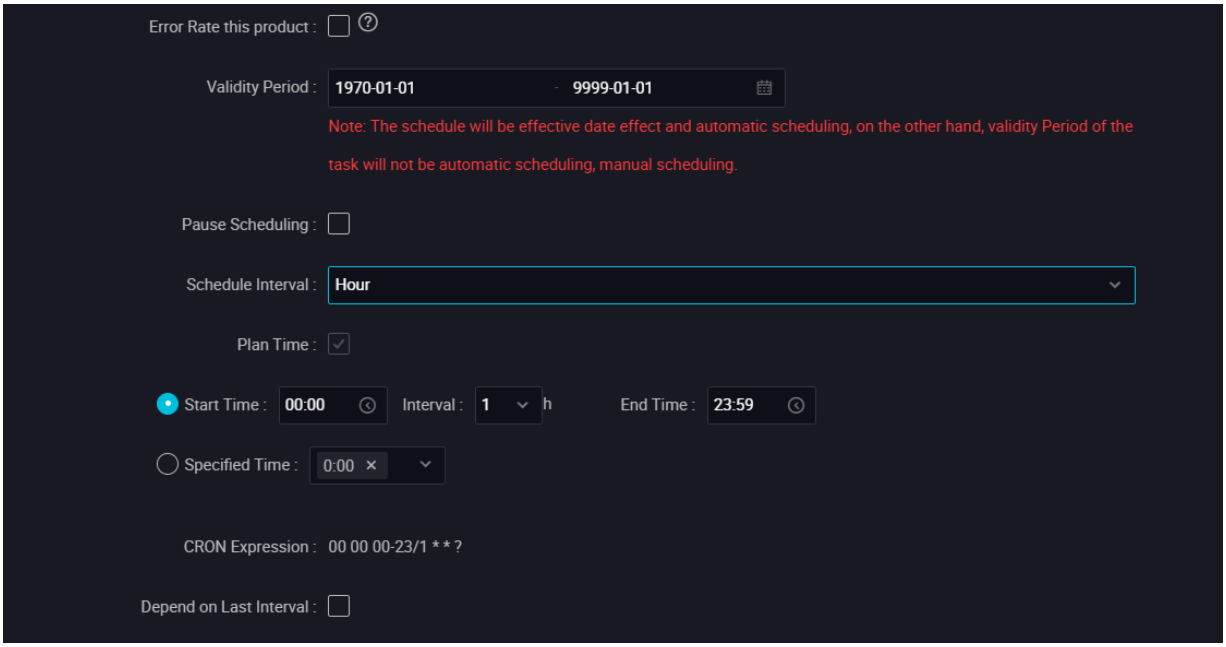


Scheduling by hour

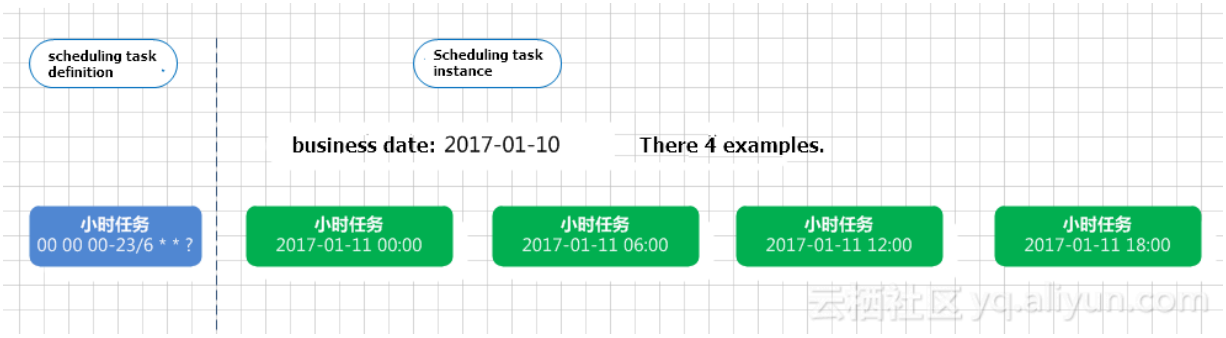
Nodes scheduled by hour are run once every N hours in a specific period every day, for example, once every hour from 01:00 to 04:00 every day.

Note:

The scheduling time is a closed interval. For example, if a node scheduled by hour is configured to run once every hour from 00:00 to 03:00, the time period is [00:00, 03:00], and the interval is 1 hour. The scheduling system generates four instances every day, which are run at 00:00, 01:00, 02:00, and 03:00, respectively.



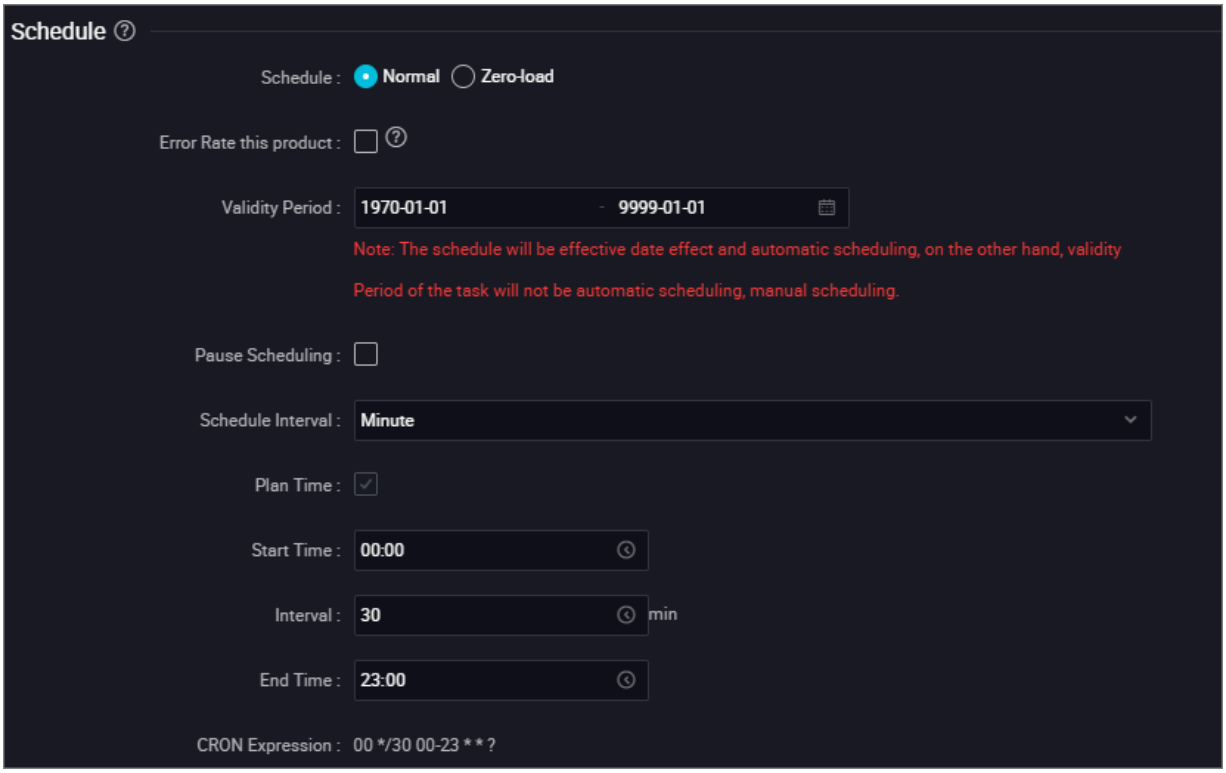
As shown in the preceding figure, the node is automatically scheduled once every 6 hours from 00:00 to 23:59 every day. Therefore, the scheduling system automatically generates instances for the node and runs them as follows:



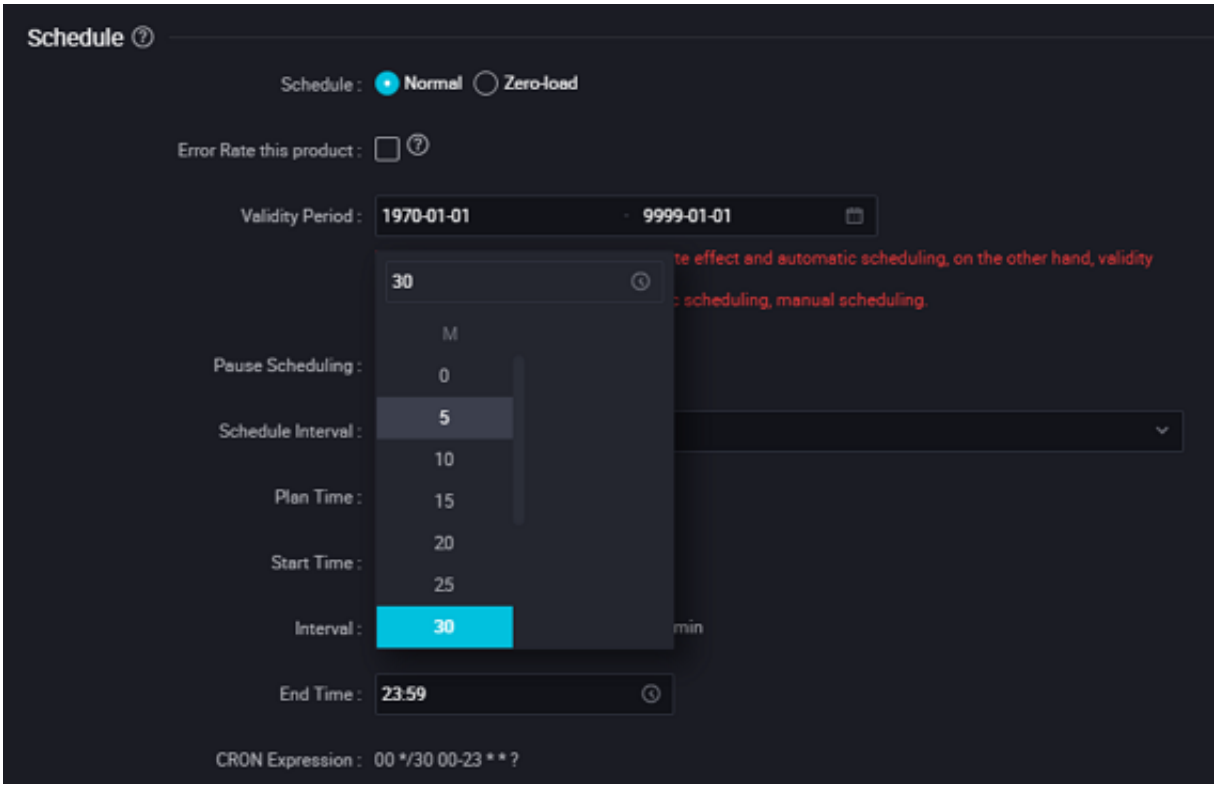
Scheduling by minute

Nodes scheduled by minute are run once every N minutes in a specific period every day, as shown in the following figure.

The node is scheduled once every 30 minutes from 00:00 to 23:00 every day.



Currently, the minimum interval is 5 minutes for a scheduled node by minute. The CRON expression is automatically generated based on the preceding configuration and cannot be manually modified.



Note:

For more information about how to generate instances immediately after nodes are published, see #unique_70.

FAQs

Q: Node A is scheduled by hour, and its descendant node B is scheduled by day. Is it feasible that node B is automatically run every day after all instances of node A are executed?

A: A node can depend on any other node, and there are no limits on the scheduling type of the node. Therefore, a node scheduled by day can depend on a node scheduled by hour. To enable node B to be automatically run every day after all 24 instances of node A are run, do not specify the daily runtime for node B. Then, configure node A as an ancestor of node B. For more information, see the Dependencies section. [DO NOT TRANSLATE]

Q: Node A is run once every hour, and node B is run once every day. How do I configure the scheduling time for the two nodes so that the instance of node B is run after the first instance of node A is run every day?

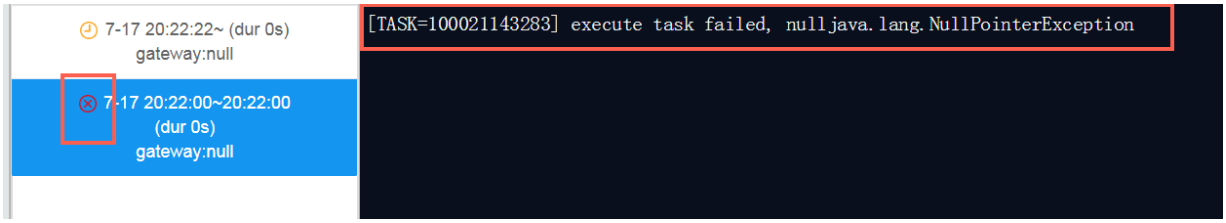
A: For node A, select Depend on Last Interval and set Dependent Node to Current Node. For node B, set Recurrence to Day, select Specify Time, and set Run At to 00:00.

Q: Node A is run every Monday and node B depends on node A. How do I enable node B to be run every Monday?

A: Configure the scheduling time of node B to be the same as those of node A. That is, you need to set Recurrence to Week and Run Every to Monday.

Q: How are the instances of a node affected when the node is deleted?

A: When a node is deleted after running for a period, its instances are remained because the scheduling system still generates one or more instances for the node based on the scheduling time. Therefore, when the instances are initiated after the node is deleted, an error message appears because the required code is unavailable, as shown in the following figure.



Q: Can I enable a node to process monthly data on the last day of each month?

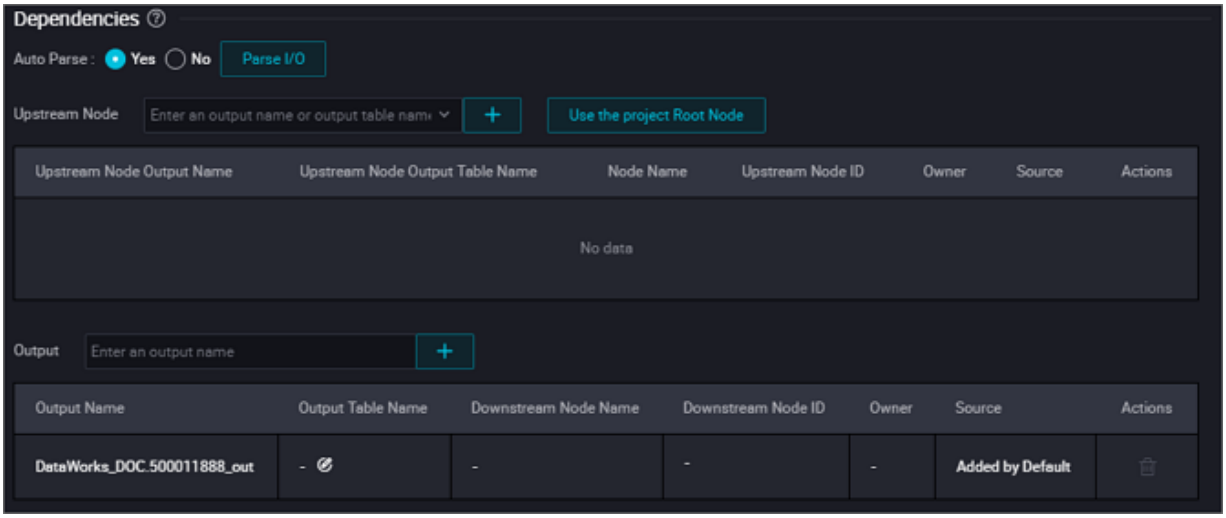
A: No. Currently, the system does not support setting the execution date to the last day of each month. If you enable a node to run on the thirty-first day of each month , the scheduling system runs a node instance in each month that has 31 days and returns a success response without running the node instance in any other month.

We recommend that you configure a node to process the data of the past month on the first day of each month.

1.6.4 Dependencies

Scheduling dependencies are the foundation for building orderly business flows. You need to correctly configure the dependencies between nodes to ensure that business data is produced effectively and in time. This helps standardize data R&D scenarios.

DataWorks V2.0 allows you to configure dependencies in any of the following modes: automatic recommendation, automatic parsing, and custom configuration. For more information about dependency configuration examples, see #unique_75.



Regardless of the dependency configuration mode, the overall scheduling logic is that descendant nodes can be scheduled only after ancestor nodes are run. Therefore, each workflow node must have at least one parent node. The dependencies between the parent nodes and child nodes are the core of scheduling dependencies. The following sections describe the principle and configuration methods of scheduling dependencies in detail.

Note:

> Data problems exist for workspaces created before January 10, 2019. You must submit a ticket to apply for troubleshooting. Projects created on January 10 and later are not affected.

Standardized data development scenarios

- Before configuring scheduling dependencies, you need to understand the following basic concepts:
  - DataWorks node: defines the operations performed on data. For more information, see #unique_76.
  - Output name: refers to the default output name that the system assigns to each node. Each output name ends with `.out`. You can also customize the output name, but make sure that the node output name is unique for the tenant. For more information, see #unique_76.
  - Output table: refers to the table following INSERT or CREATE in the SQL statement of a node.
  - Input table: refers to the table following FROM in the SQL statement of a node.
  - SQL statement: refers to MaxCompute SQL.

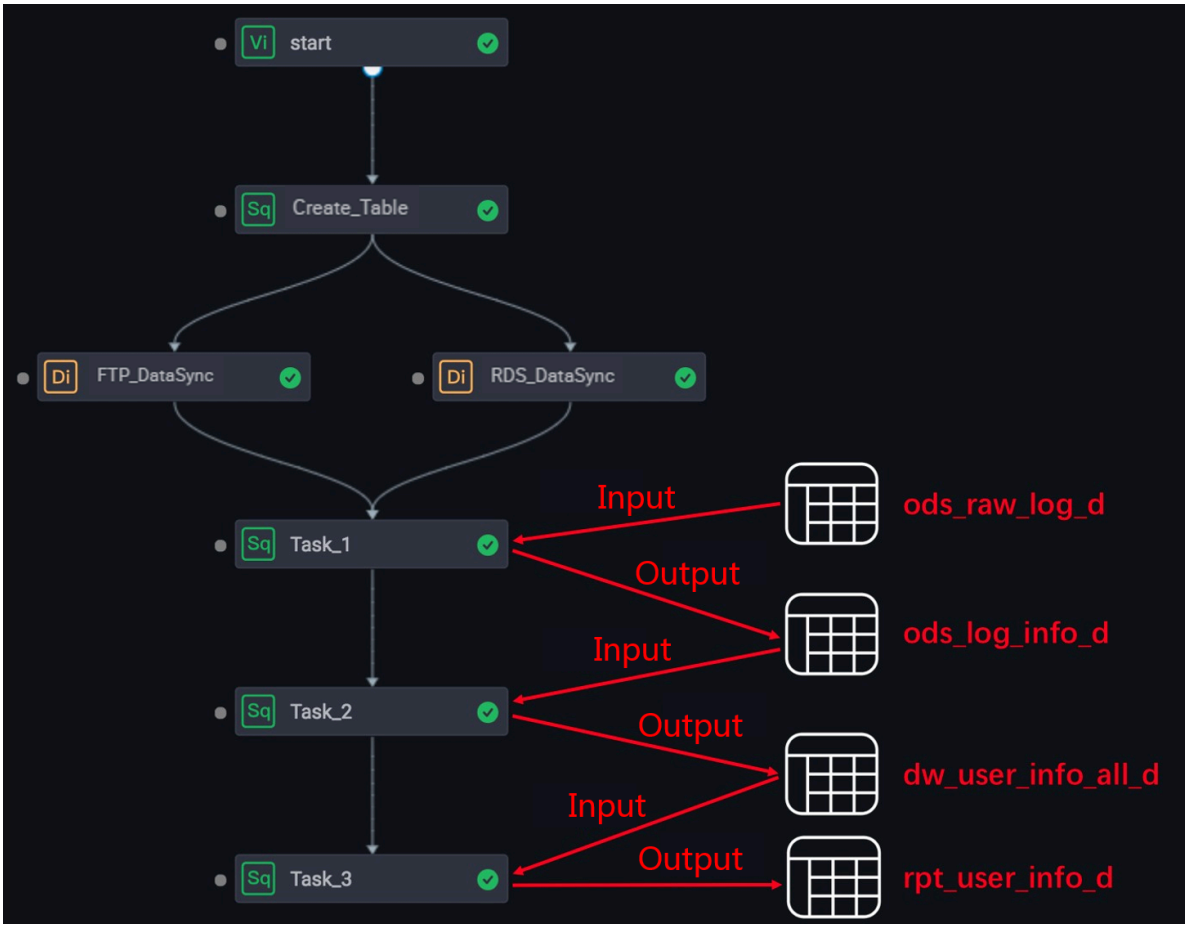
  In practice, a DataWorks node can contain a single SQL statement or multiple SQL statements.

  Ancestor and descendant nodes are associated by output names. The ancestor node of the upmost node that is created can be configured as the root node of the workspace (workspace name: `projectnam e_root`).
- Principles of standardized data development

  In a standardized data development process, multiple ancestor and descendant SQL nodes are created. We recommend that you follow these principles:
  - The input table of a descendant node must be the output table of its ancestor node.
  - One table can only be exported by one node.

  The purpose is to quickly configure complex dependencies by using the automatic parsing feature when business processes are inflated.

- Example of a standardized data development process



Each node and its code in the preceding figure are described as follows:

- The code of Task_1 is as follows. The input data of this node comes from the `ods_raw_log_d` table, and the data is exported to the `ods_log_info_d` table.

```
INSERT OVERWRITE TABLE ods_log_info_d PARTITION (dt=${bdp.system.bizdate})
SELECT ... // It represents your SELECT operation.
FROM (
SELECT ... // It represents your SELECT operation.
FROM ods_raw_log_d
WHERE dt = ${bdp.system.bizdate}
) a;
```

- The code of Task_2 is as follows. The input data of this node comes from the `ods_user_info_d` and `ods_log_info_d` tables, and the data is exported to the `dw_user_info_all_d` table.

```
INSERT OVERWRITE TABLE dw_user_info_all_d PARTITION (
dt='${bdp.system.bizdate}')
SELECT ... // It represents your SELECT operation.
FROM (
  SELECT *
```

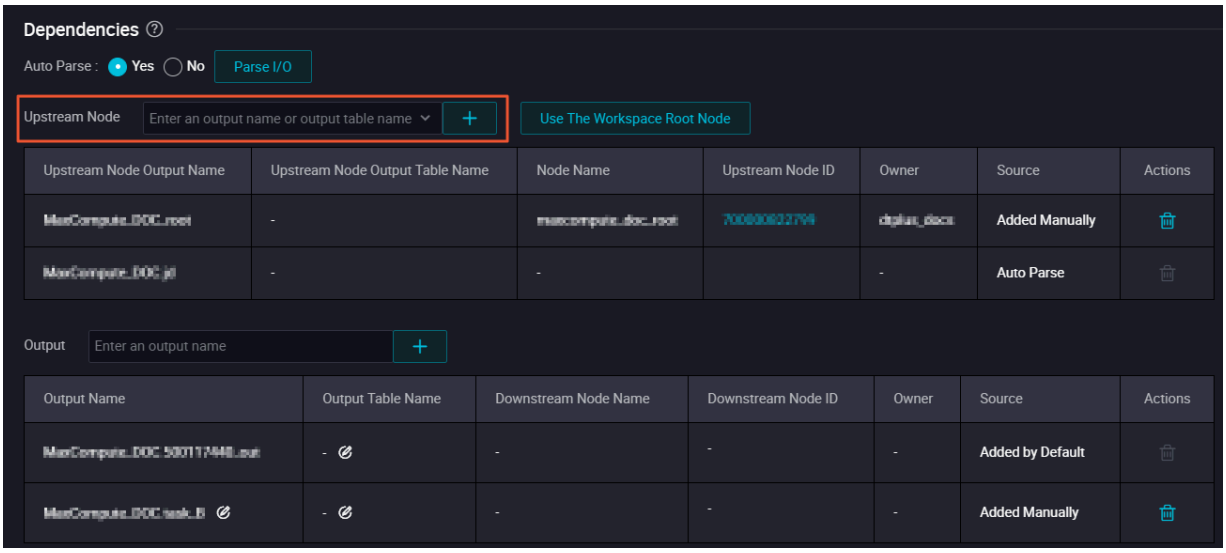
```
    FROM   ods_log_in  fo_d
    WHERE   dt  = ${ bdp . system . bizdate }
)  a
 LEFT   OUTER   JOIN  (
    SELECT  *
    FROM   ods_user_i  nfo_d
    WHERE   dt  = ${ bdp . system . bizdate }
)  b
 ON   a . uid  =  b . uid ;
```

- The code of Task_3 is as follows. The input data of this node comes from the `dw_user_in fo_all_d` table, and the data is exported to the `rpt_user_i nfo_d` table.

```
INSERT   OVERWRITE   TABLE   rpt_user_i  nfo_d   PARTITION  ( dt
='${ bdp . system . bizdate }')
SELECT  ...  // It   represents   your   SELECT   operation .
FROM   dw_user_in  fo_all_d
WHERE   dt  = ${ bdp . system . bizdate }
GROUP   BY   uid ;
```

Ancestor node

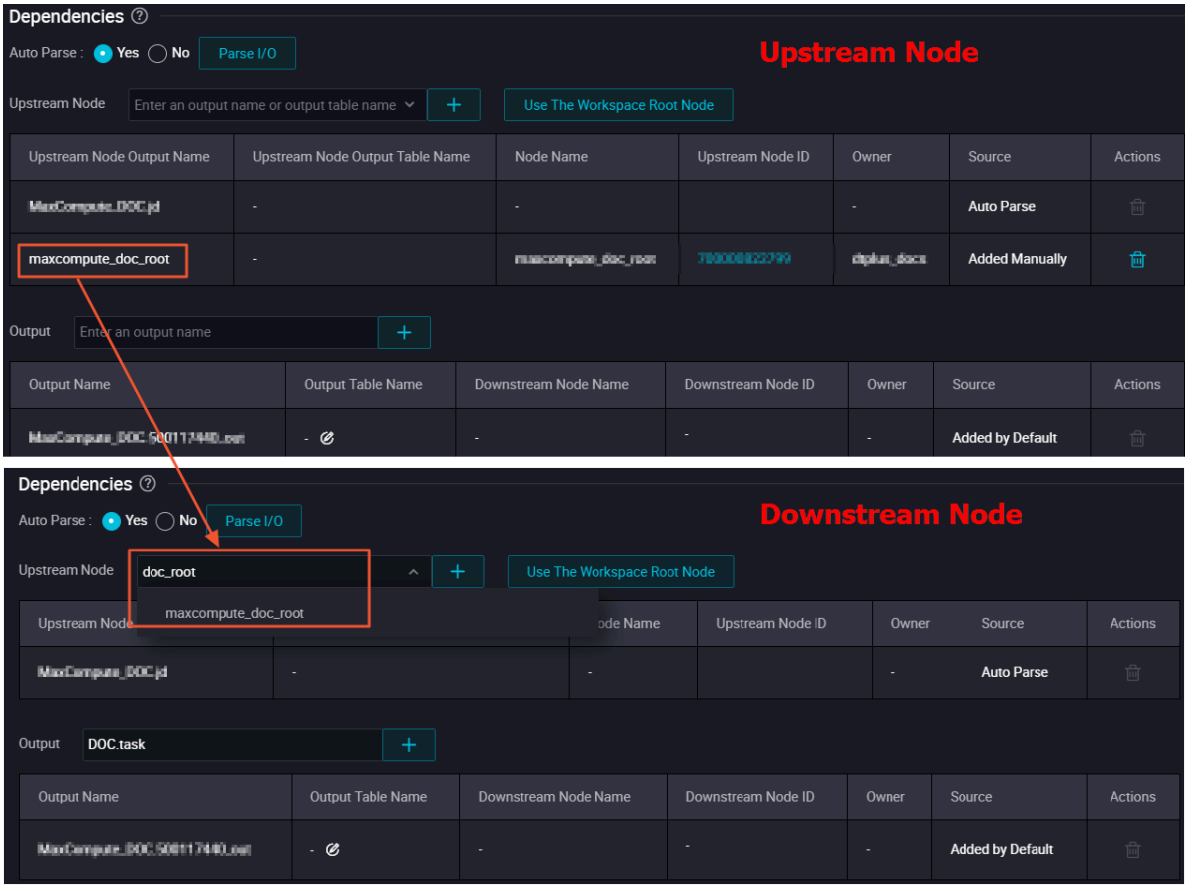
An ancestor node indicates the parent node on which the current node depends. You must enter the output name of the ancestor node, rather than the ancestor node name. (A node may contain multiple output names. Enter an output name as needed.) You can search for the output name of the ancestor node to be added, or run the SQL statement for lineage analysis to parse the output name.



Note:
If you use the first method, the crawler searches for the output name among the output names of nodes that have been submitted to the scheduling system.

- Search by entering the output name of the parent node

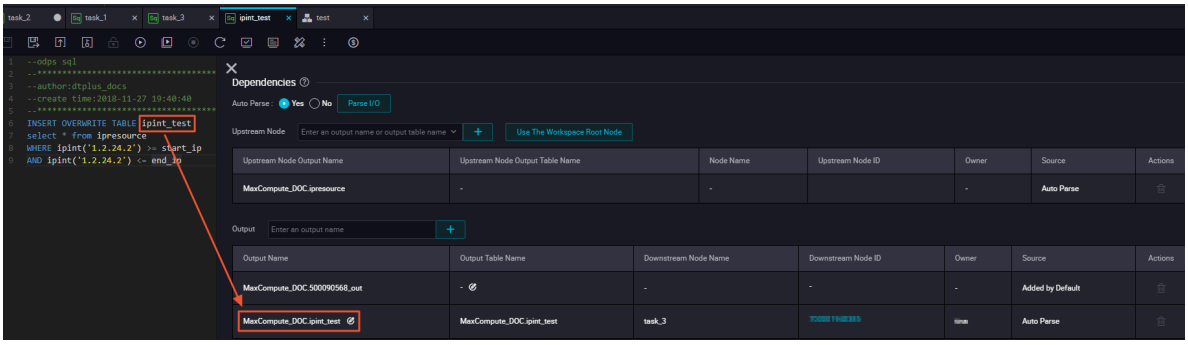
  You can search for the output name of a node and configure the node as the ancestor node of the current node to create a dependency.



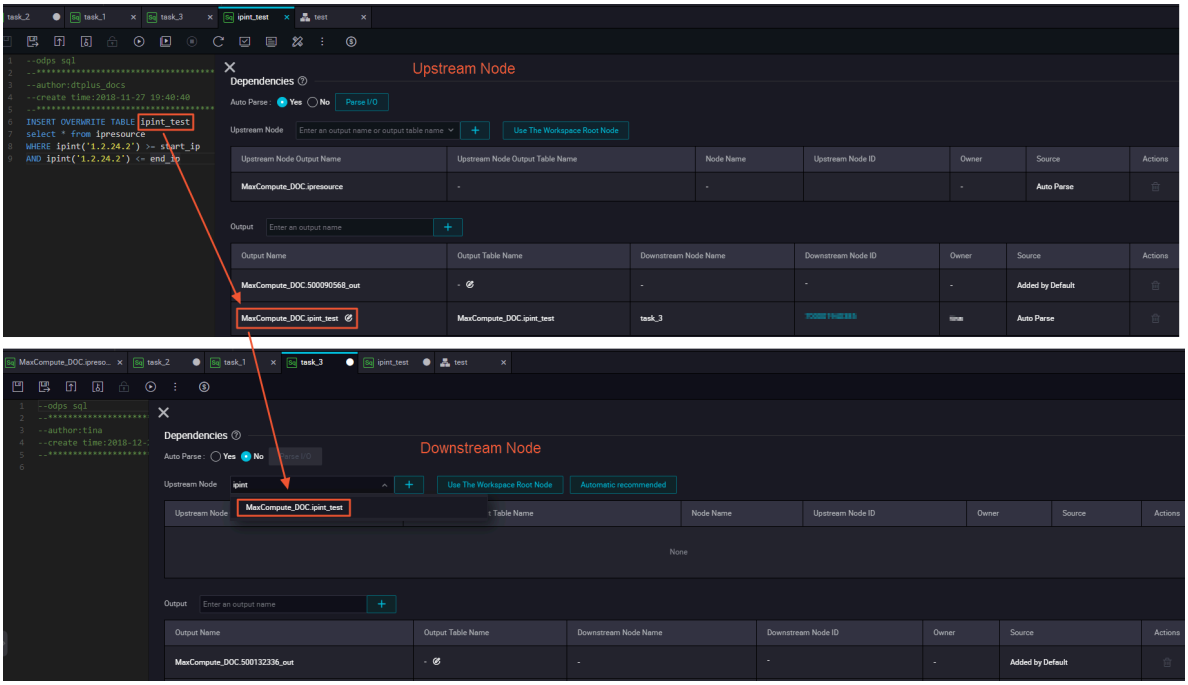
- Search by entering the table name corresponding to the output name of the parent node

  When using this method, make sure that one of the output names of the parent node is the table name following INSERT or CREATE in the SQL statement of the

node, such as projectname. Table name. (Such output names can be automatically parsed.)



Click Submit. The output name can be searched by other nodes by searching the table name.



Current node output

The current node output indicates the output of the current node.

The system assigns a default output name that ends with .out to each node. You can also customize the output name or obtain an output name by automatic parsing.

> **Note:**
> The output name of a node must be globally unique for your Alibaba Cloud account.

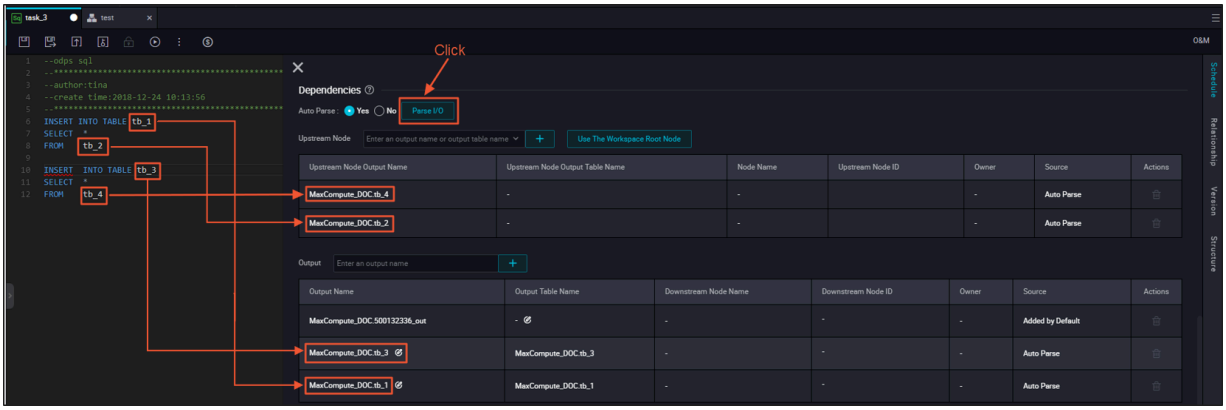
Automatic dependency parsing

DataWorks can parse different dependencies based on the actual SQL content of the node. The parsed output names of the parent node and the current node are as follows:

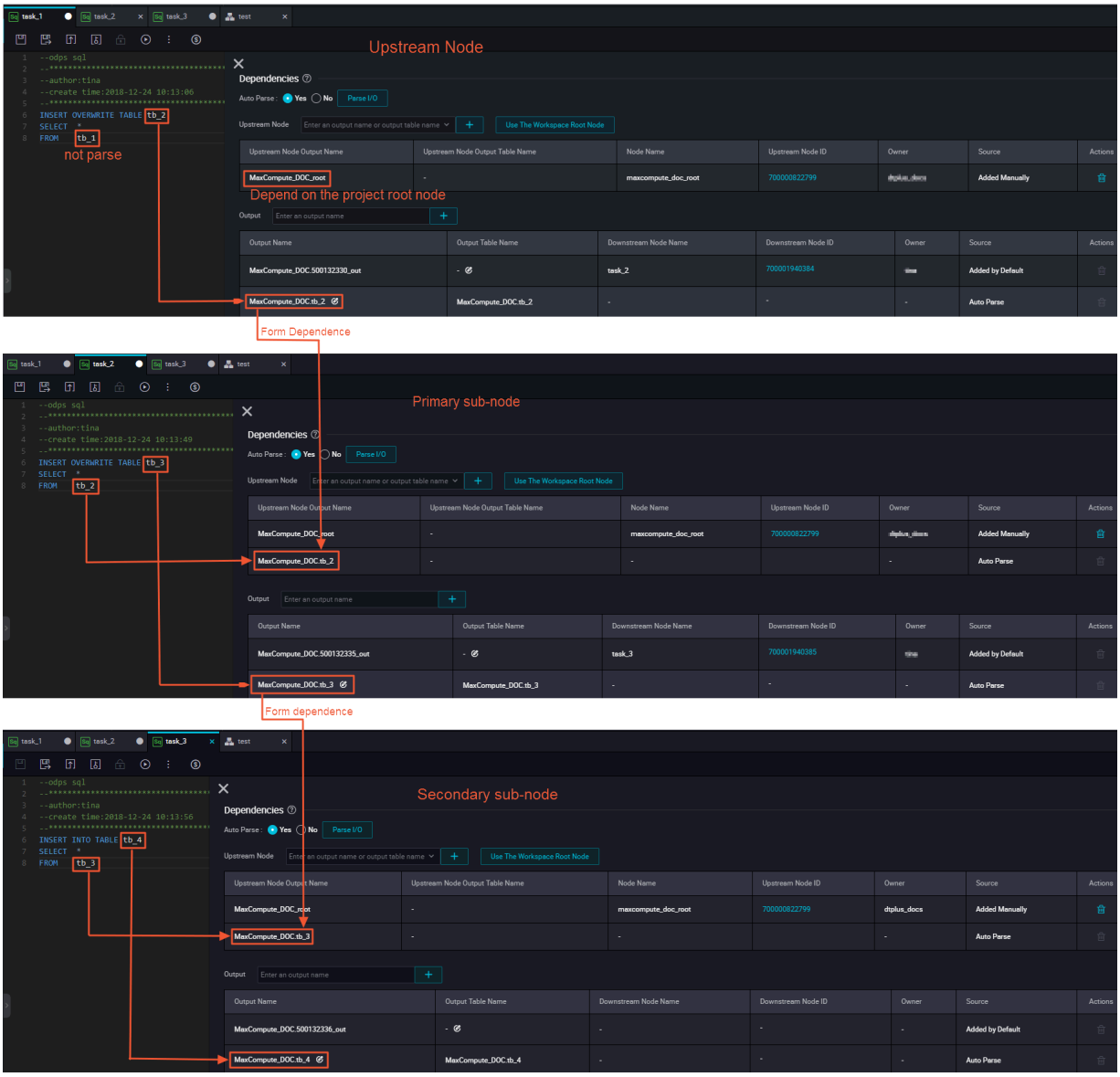
- Output name of the parent node: projectname.The table name following INSERT.
- Output names of the current node:
  - projectname.The table name following INSERT.
  - projectname.The table name following CREATE (generally used for temporary tables).

> **Note:**
> If you upgrade from DataWorks V1.0 to DataWorks V2.0, the output name of the current node is projectname. The name of the current node.

If multiple INSERT and FROM clauses are displayed, multiple output and input names are automatically parsed.



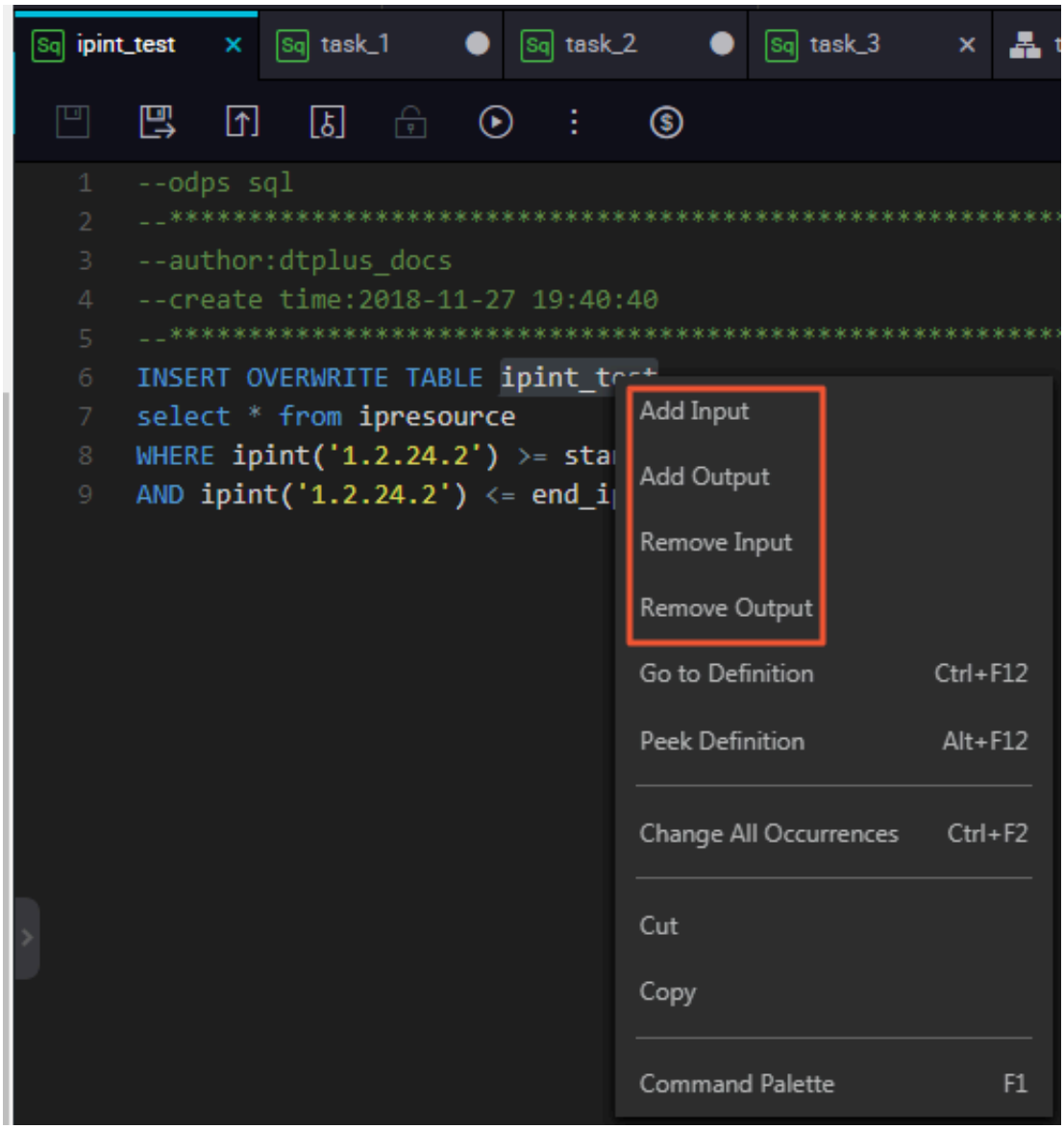
If you create multiple nodes with dependencies, and all input tables of descendant nodes come from the output tables of ancestor nodes, the automatic parsing feature can be used to quickly configure dependencies for the entire workflow.



Note:

- To make the node more flexible, we recommend that a node contain only one output node, so that you can flexibly assemble SQL business processes for decoupling.
- If a table in an SQL statement is both an output table and a referenced table (a depended table), the table is parsed only as an output table.
- If a table in an SQL statement is referenced or exported for multiple times, only one scheduling dependency is parsed.
- If the SQL code contains a temporary table (for example, a table whose name starts with t_ is specified as a temporary table during attribute configuration), the table is not involved in a scheduling dependency.

When automatic parsing applies, you can add input and output to enable characters in SQL statements to be automatically parsed into input and output names, or delete input and output to avoid characters from being automatically parsed into input and output names.



Right-click a table name and select Add Input, Add Output, Remove Input, or Remove Output to modify the dependencies. This method applies to all table names in SQL statements. If you select Add Input, the characters are parsed as the input name of the parent node. If you select Add Output, the characters are parsed as the output name of the current node. If you select Remove Input or Remove Output, the characters are not parsed.

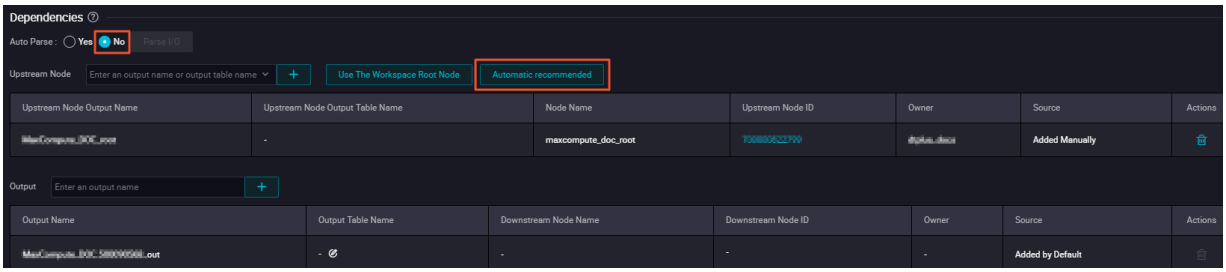
Note:

In addition to right-clicking characters in SQL statements, you can add comments to modify the dependencies. The specific code is as follows:

```
 --@ extra_inpu  t = table   name  -- Add   an   input .
--@ extra_outp  ut = table   name  -- Add   an   output .
--@ exclude_in  put = table   name  -- Delete   an   input .
 --@ exclude_ou  tput = table   name  -- Delete   an   output .
```

Custom dependencies

When dependencies between nodes cannot be accurately parsed by running the SQL statement for lineage analysis, you can set Auto Parse to No in the following figure and configure dependencies.



When Auto Parse is set to No, you can click Auto Recommendation to enable automatic recommendation of ancestor dependencies. The system recommends all other SQL nodes that export the input table of the current node based on the SQL lineage of the project. You can select one or more nodes in the recommendation list as needed and configure them as the ancestor nodes of the current node.

Note:
The recommended nodes must be submitted to the scheduling system on the previous day, and can be recognized by the automatic recommendation feature after data is generated on the second day.

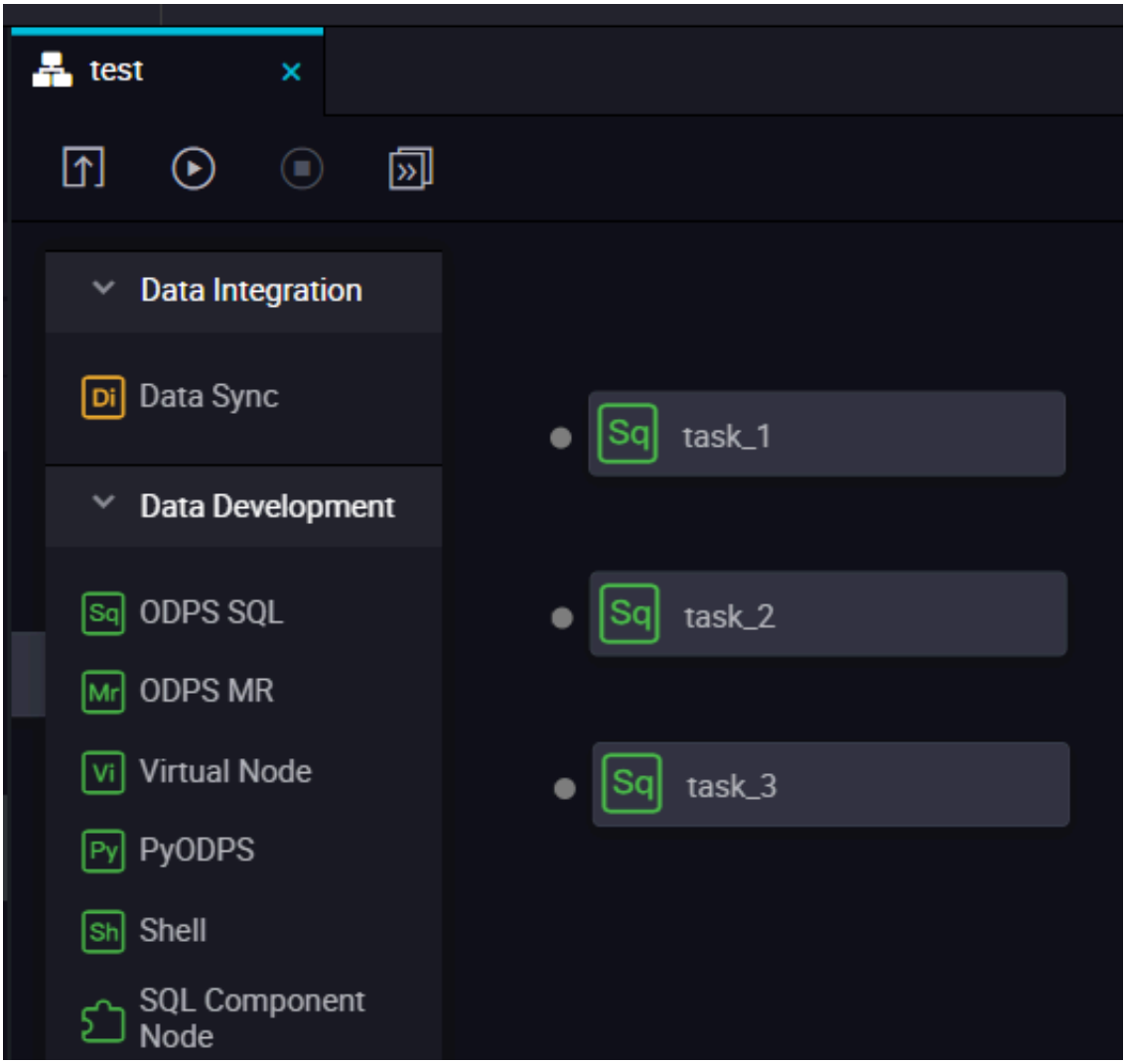
Common scenarios:

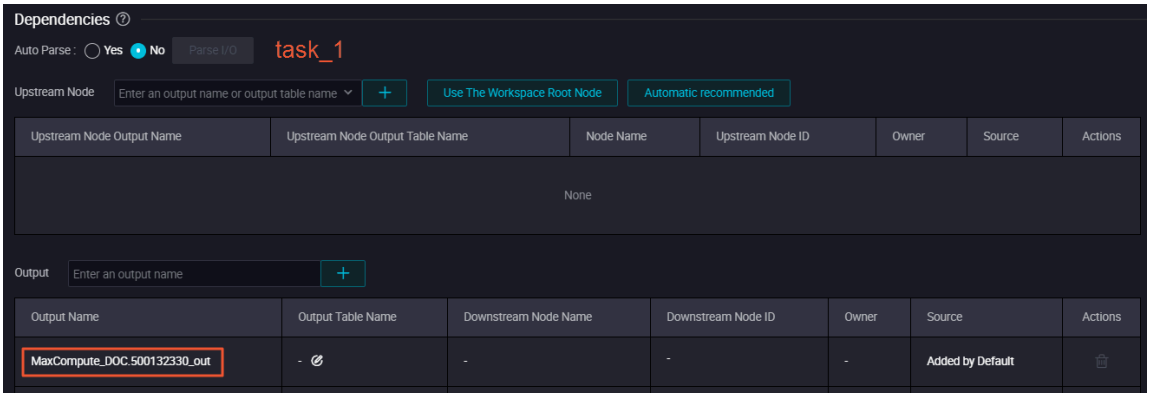
- The input table of the current node is not equivalent to the output table of the ancestor node.
- The output table of the current node is not equivalent to the input table of the descendant node.

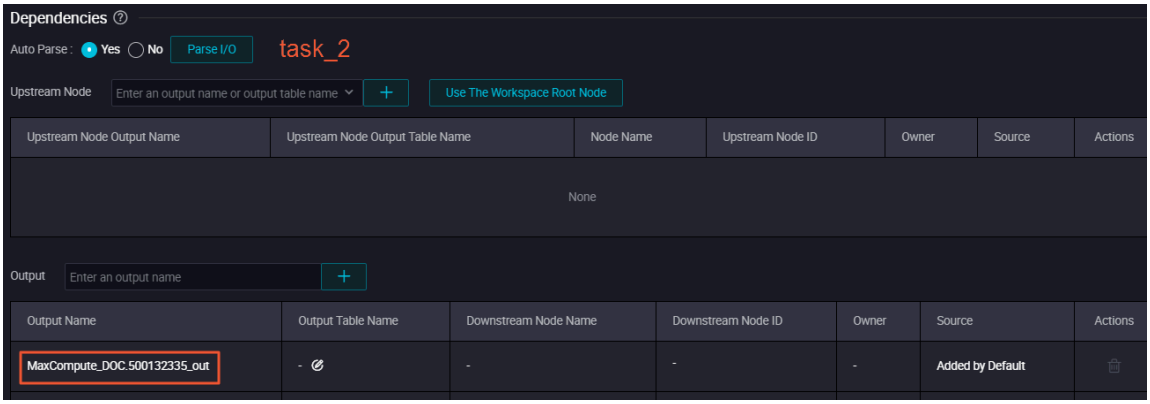
In custom mode, you can configure dependencies in the following ways:

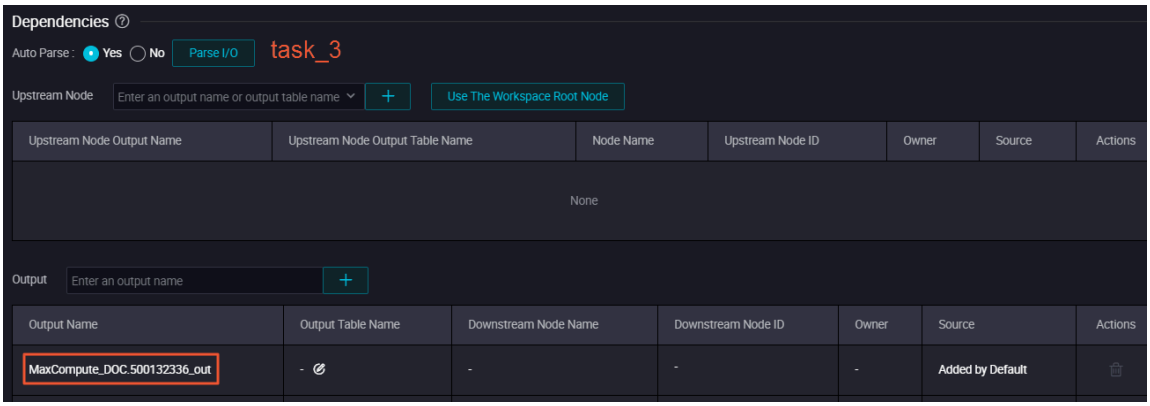
- Manually add ancestor nodes

  1. Create three nodes. The system configures an output name for each of them by default.

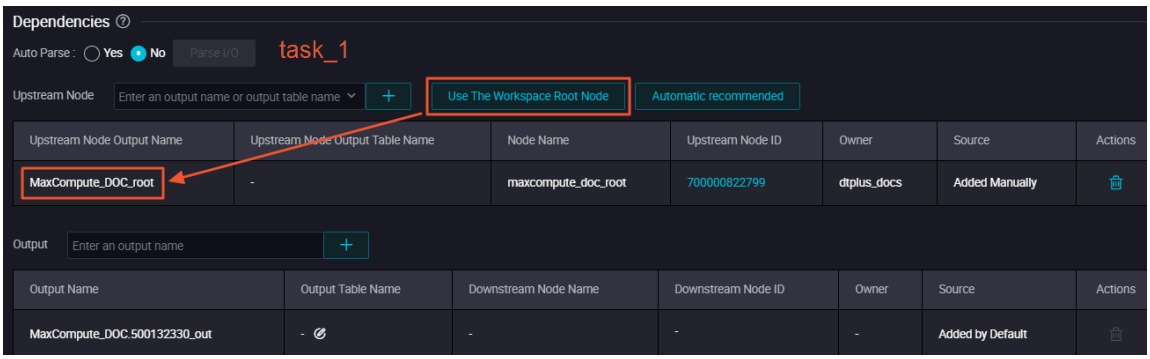




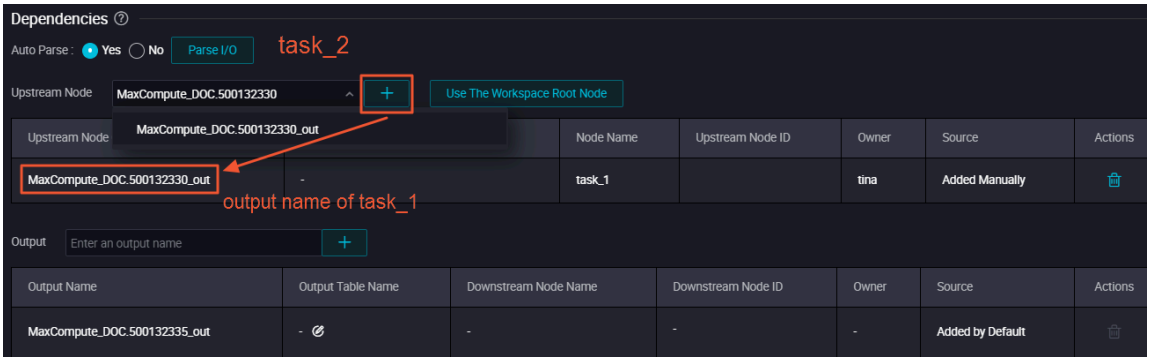




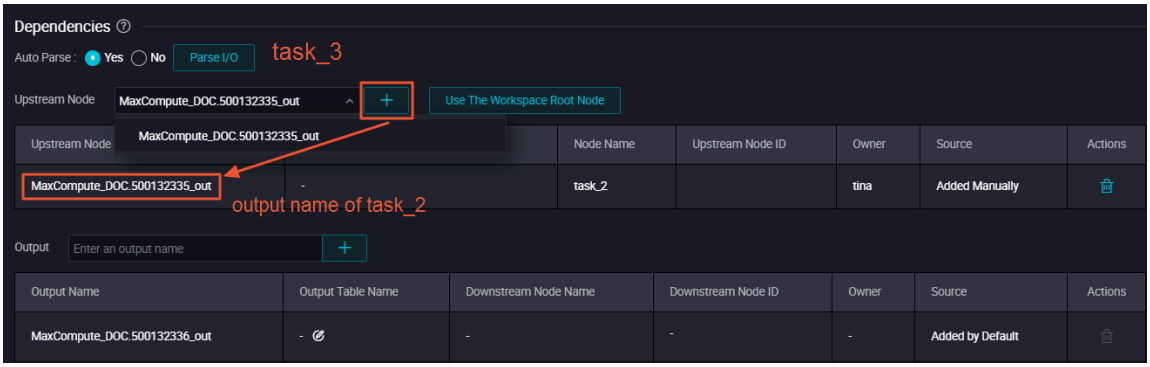
2. Configure the upmost node task_1 to depend on the root node of the workspace, and click Save.



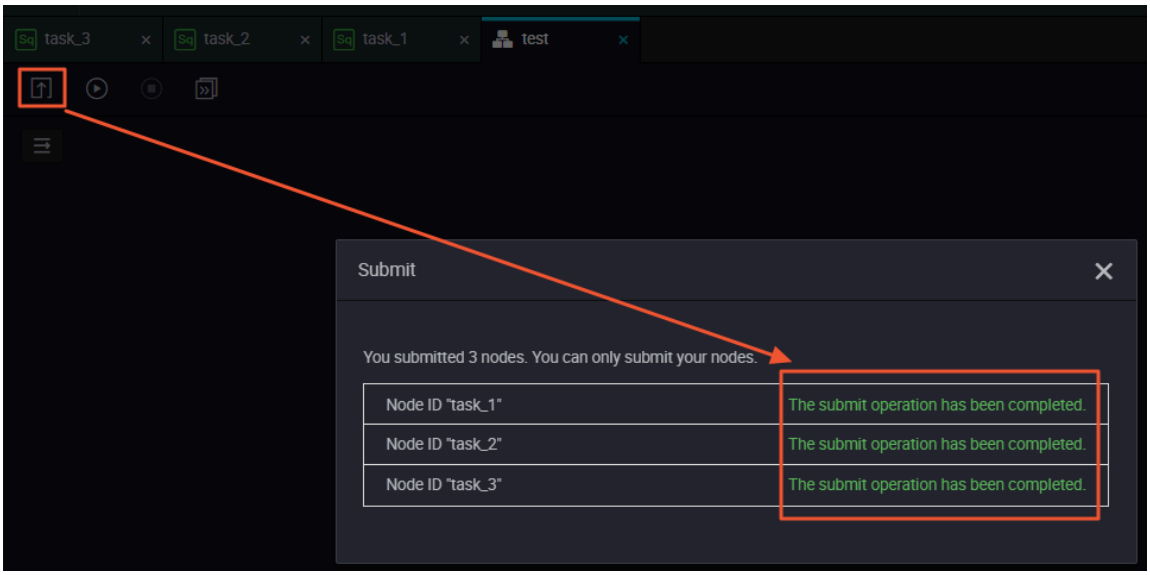
3. Configure task_2 to depend on the output name of task_1, and click Save.



4. Configure task_3 to depend on the output name of task_2, and click Save.



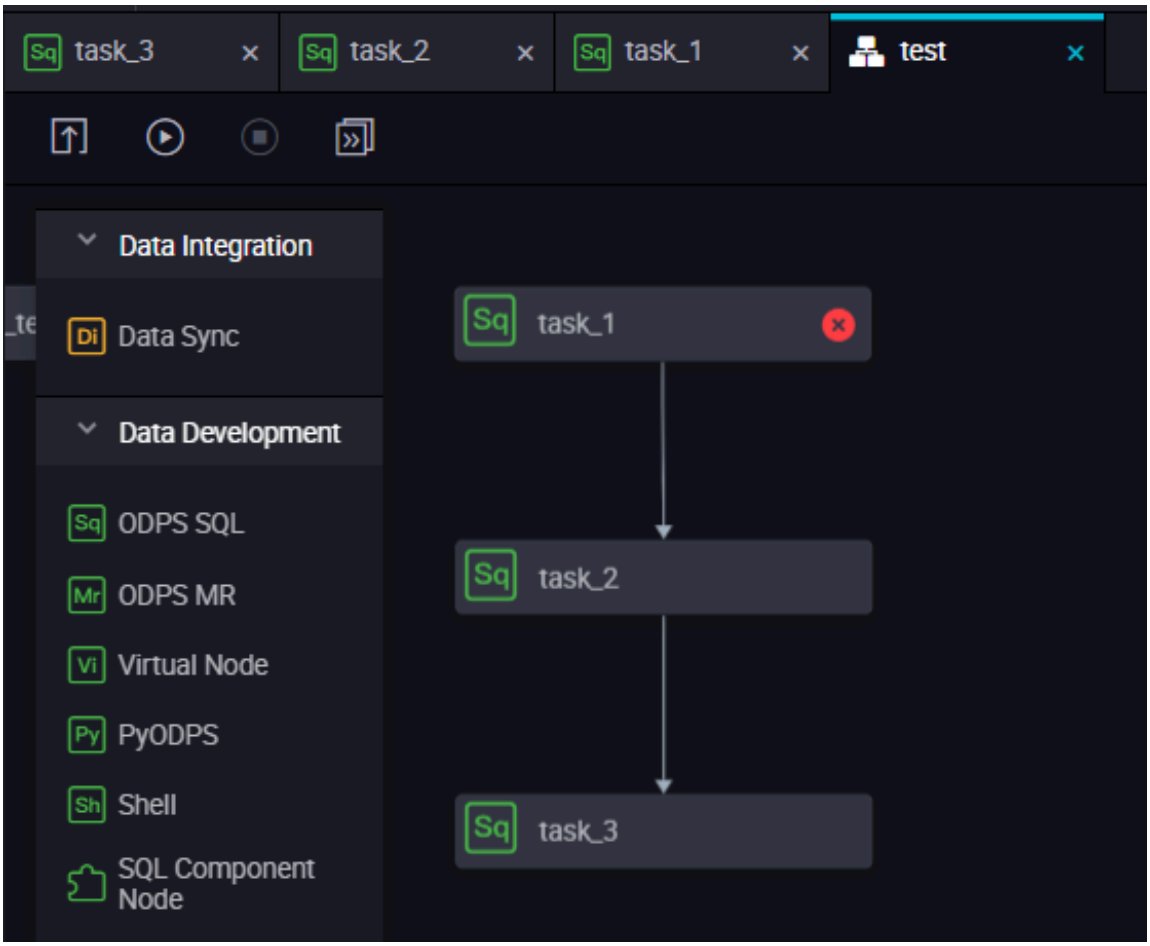
5. After the configuration is complete, click Submit to check whether the dependency is correct. If the submission is successful, the dependency configuration is correct.



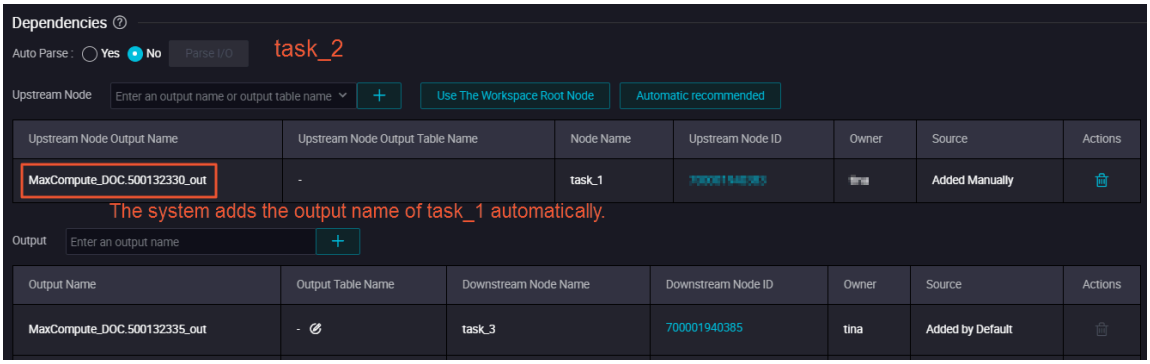
- Build dependencies by dragging and dropping

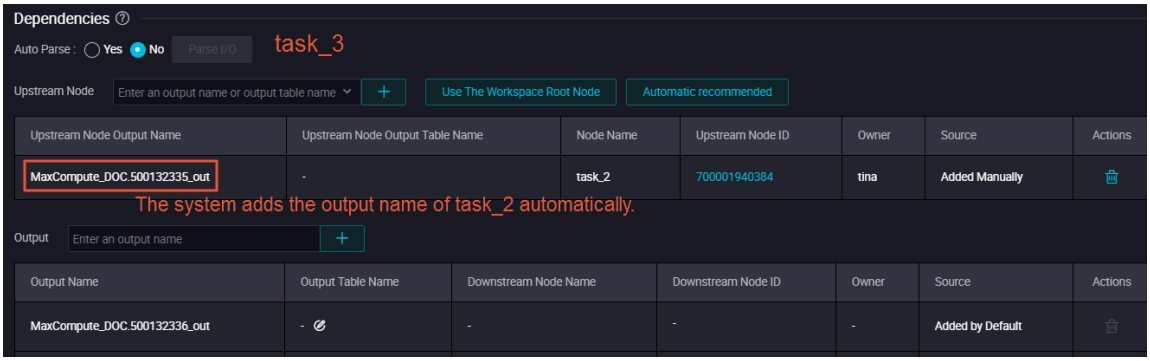
  1. Create three nodes, configure the upmost node task_1 to depend on the root node, and click Save.

  2. Drag the three nodes to connect them.

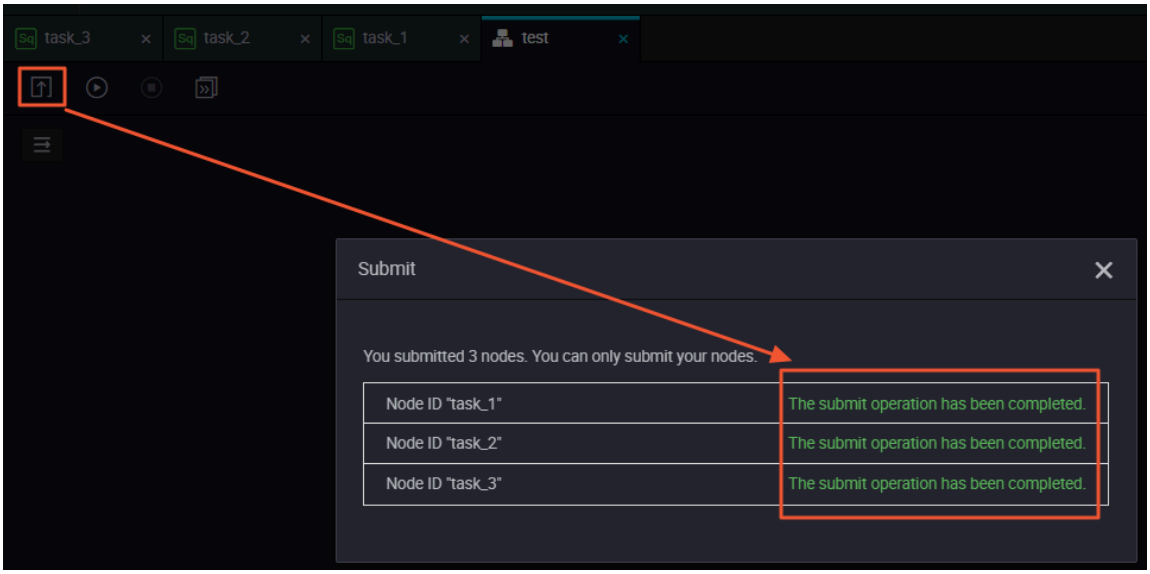


  3. View the dependency configuration of task_2 and task_3. The output name of the parent node is automatically generated.



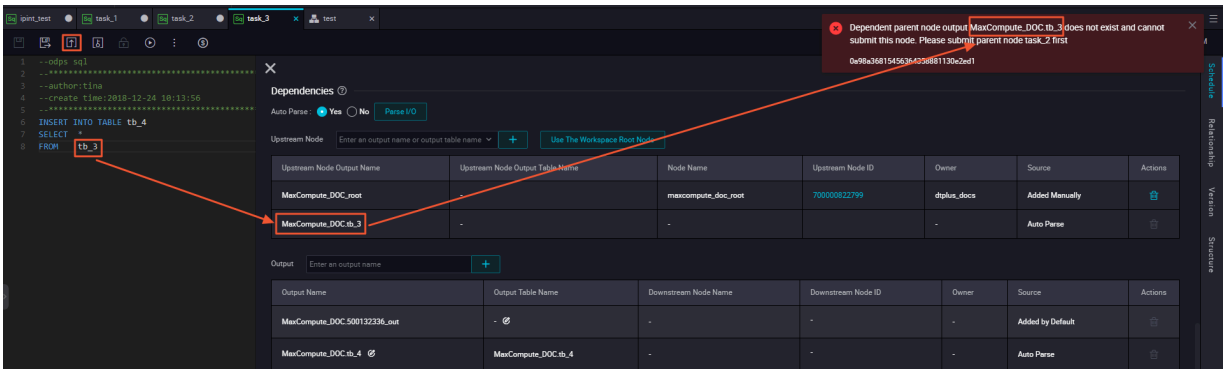


4. After the configuration is complete, click Submit to check whether the dependency is correct. If the submission is successful, the dependency configuration is correct.



FAQs

Q: After automatic parsing, the submission fails. The following error message is displayed: The output workshop_yanshi.tb_2 of the parent node does not exist. Submit this node after submitting the parent node.



A: This problem can be caused by either of the following reasons:

- The ancestor node is not submitted. Submit the ancestor node and try again.
- The ancestor node is submitted, but the output name of the ancestor node is not workshop_yanshi.tb_2.

> Note:
> Usually, the output names of the parent node and the current node are automatically parsed based on the table name following INSERT, CREATE, or FROM. Make sure that the configuration method is consistent with that described in the section "Automatic dependency parsing."

Q: In the output of the current node, the descendant node name and ID are empty and cannot be specified. Why does this happen?

A: If the current node does not have any descendant node, the descendant node name and ID are empty. After a descendant node is configured for the current node, the corresponding content is automatically parsed.

Q: What is the output name of a node used for?

A: The output name of a node is used to establish dependencies with other nodes. If the output name of node A is ABC and node B takes ABC as its input, the dependency is established between nodes A and B.

Q: Can a node have multiple output names?

A: Yes. If a descendant node references an output name of the current node as its parent node output name, the dependency is established between the descendant node and the current node.

Q: Can multiple nodes have the same output name?

A: No. The output name of each node must be unique for your Alibaba Cloud account . If multiple nodes export data to the same MaxCompute table, we recommend that you use Table name_Partition ID as the output of these nodes.

Q: How can I configure no parsing of intermediate tables during automatic dependency parsing?

A: Right-click the intermediate table name in the SQL code and select Remove Input or Remove Output, and then perform the automatic parsing of the input and output again.

Q: How do I configure dependencies of the upmost node?

A: In general, the upmost node depends on the root node of the workspace.

Q: Why do I find a non-existent output name of node B when searching for the ancestor node output name on node A?

A: Because the search feature works based on the submitted node information. After node B is submitted, if you delete the output name of node B and does not submit node B to the scheduling system, the deleted output name of node B can still be found on node A.

Q: How do I implement the node flow of A->B->C once an hour (run node B after node A is completed, and run node C after node B is completed)?

A: Set the output of node A as the input of node B and the output of node B as the input of node C, and set the scheduling periods of nodes A, B, and C to 1 hour.

1.6.5 Cross-cycle dependencies

Cross-cycle dependencies allow nodes to depend on instances of nodes in the last cycle.

DataWorks supports the following types of cross-cycle dependencies:

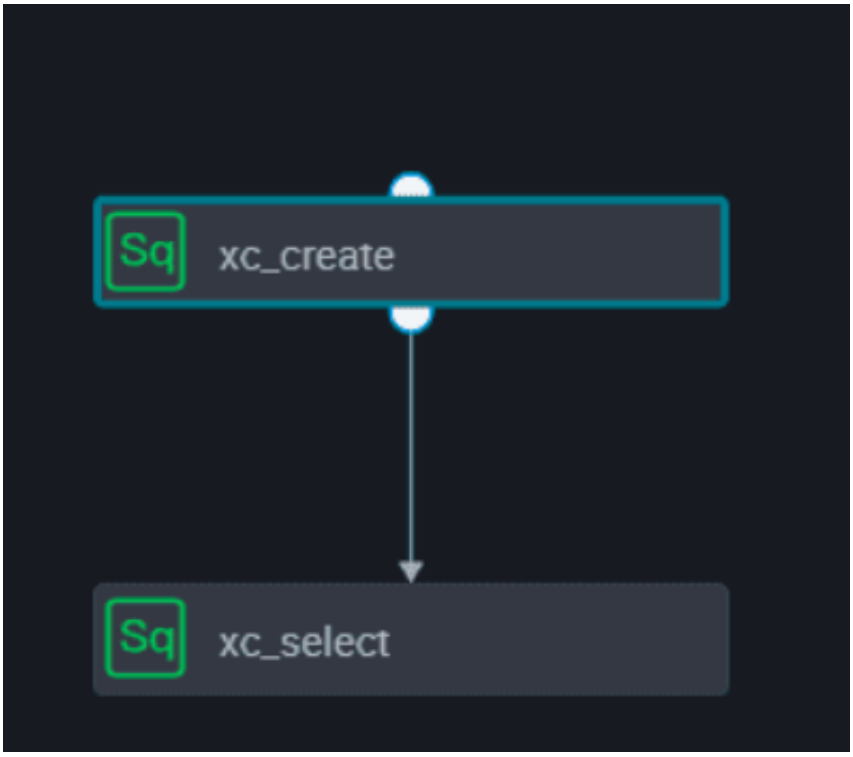
- Dependency on instances of child nodes
  - Node dependency: depends on instances of child nodes in the last cycle. For example, node A has three child nodes, namely, nodes B, C, and D, and node A depends on instances of nodes B, C, and D in the last cycle.
  - Business scenario: The current node depends on instances of child nodes in the last cycle to cleanse the output tables of the current node and check whether the final result is generated properly.
- Dependency on the instance of the current node
  - Node dependency: depends on the instance of the current node in the last cycle.
  - Business scenario: The current node depends on the data output result of the instance of the current node in the last cycle.

- Dependency on instances of custom nodes: You need to manually enter the IDs of the nodes on which the current node depends. Separate the IDs with commas (,), for example, 12345,23456.
  - Node dependency: depends on instances of custom nodes in the last cycle.
  - Business scenario: In the business logic, the current node depends on the proper output of other service data that is not processed by the current node.

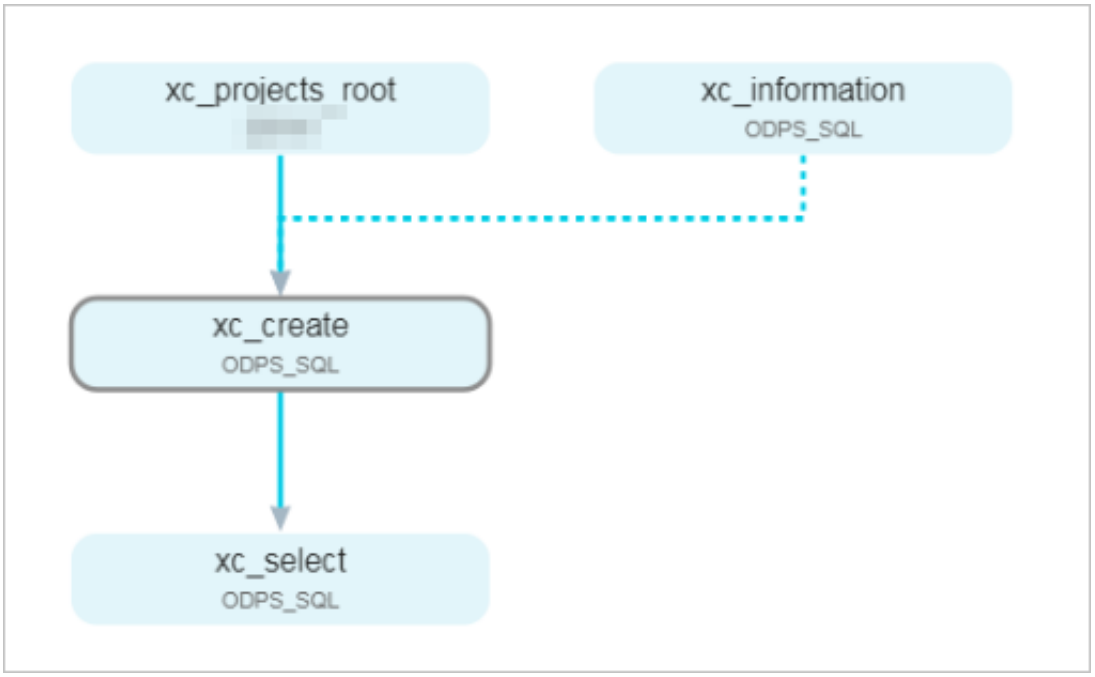
Note:
The difference between cross-cycle dependencies and dependencies in the current cycle lies in that cross-cycle dependencies are displayed as dotted lines in Operation Center.

Before bringing a node offline, you must delete all dependencies of the node so that other nodes can run properly.

Take the xc_create and xc_select nodes in a workflow as an example. The following figure shows the dependency between the two nodes.



The following figure shows how the dependency is displayed in Operation Center.

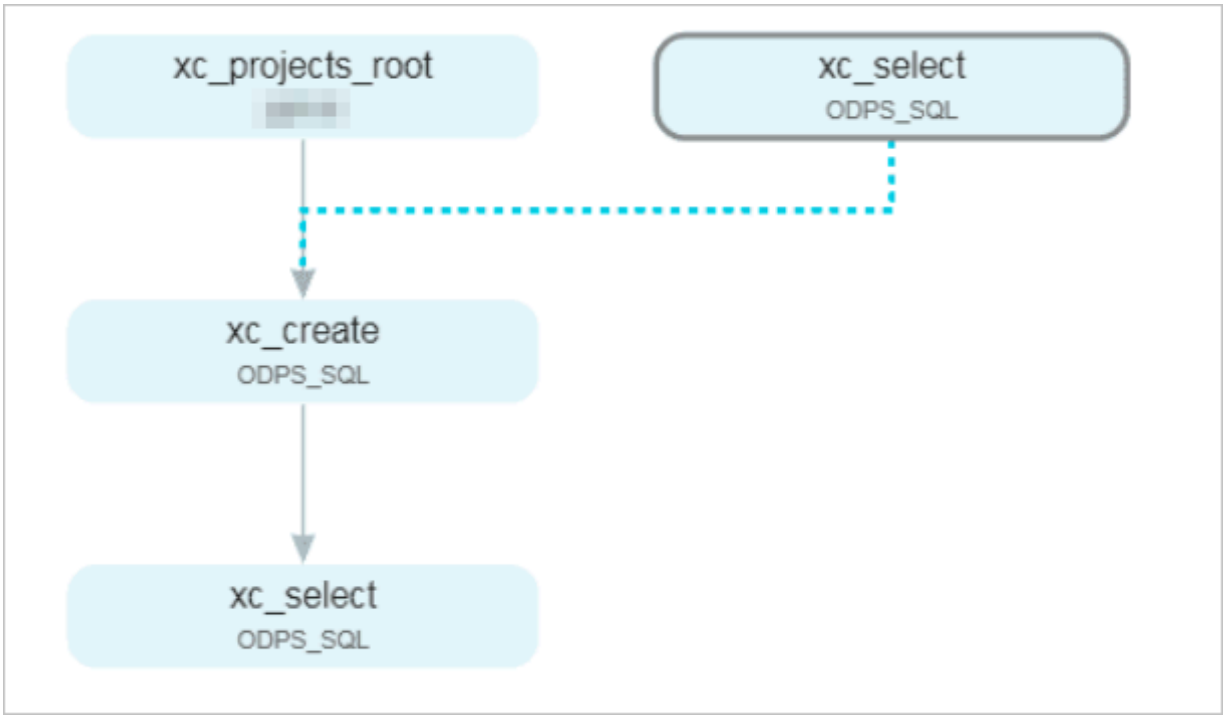


Cross-cycle dependencies: instances of child nodes

Node dependency: depends on instances of child nodes in the last cycle. For example , node A has three child nodes, namely, nodes B, C, and D, and node A depends on instances of nodes B, C, and D in the last cycle.

Business scenario: The current node depends on instances of child nodes in the last cycle to cleanse the output tables of the current node. The current node starts to run in the current cycle only when the child nodes were run successfully in the last cycle.

Select Cross-Cycle Dependencies and set Depend On to Instances of Child Nodes for the xc_create node. The following figure shows how the dependency is displayed in Operation Center.

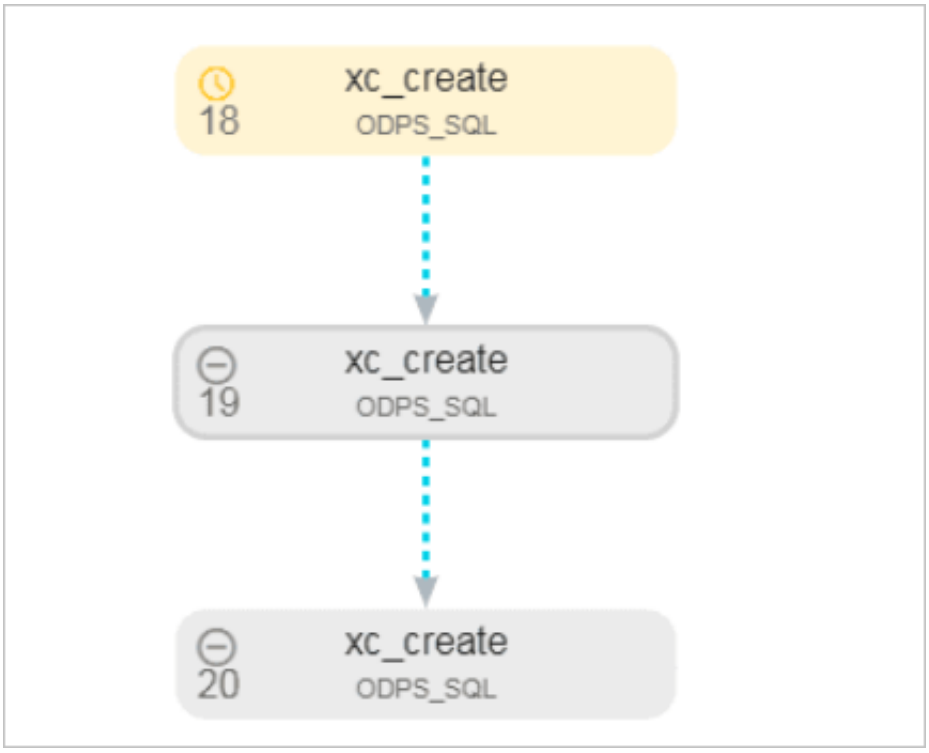


Cross-cycle dependencies: instances of current node

Node dependency: The current node depends on the running status of the instance of the current node in the last cycle. If the current node is not finished in the last cycle, the current node will not run in the current cycle.

Business scenario: The current node depends on the data cleansing result in the last cycle. In this example, Instance Recurrence is set to Hour.

The following figure shows how the dependency is displayed in Operation Center.



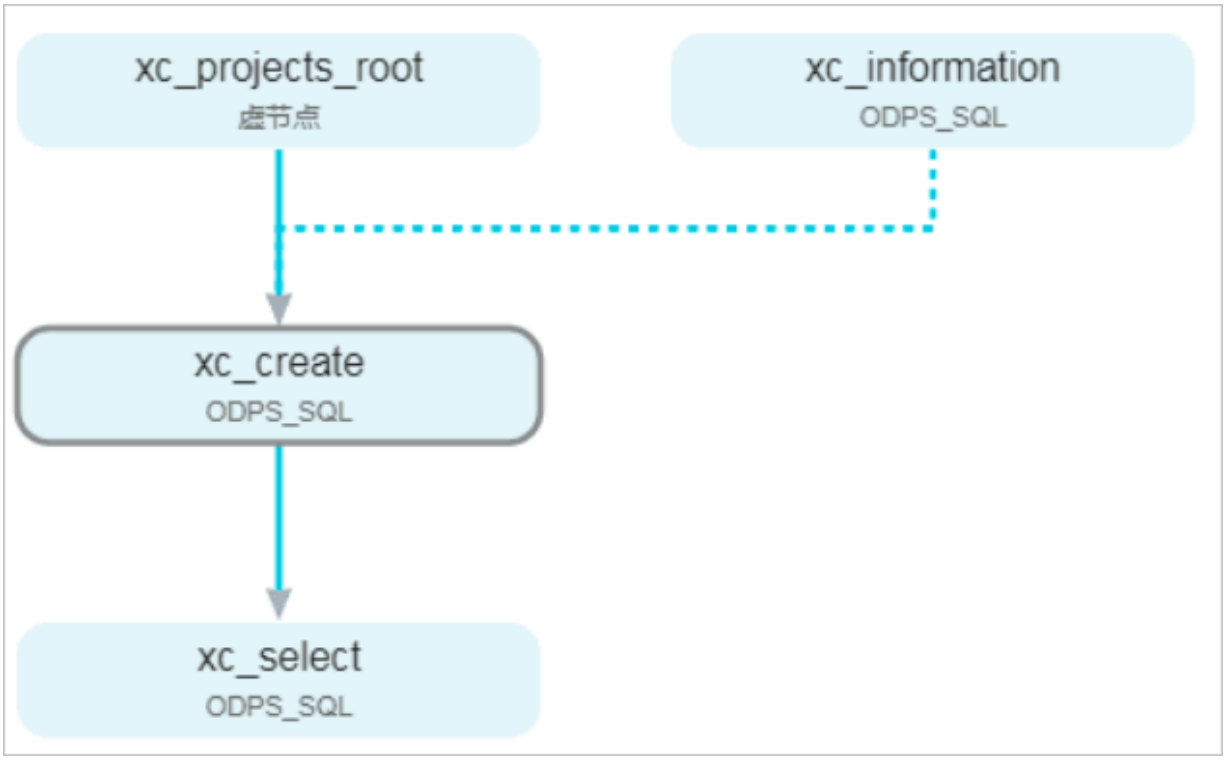
Cross-cycle dependencies: instances of custom nodes

Node dependency: The output tables of the xc_information node are not used in the code of the xc_create node. However, the xc_create node depends on the data output result of the xc_information node in the last cycle according to the business logic. That is, the xc_create node depends on the instance of the xc_information node in the last cycle.

Business scenario: In the business logic, the xc_create node depends on proper data output the xc_information node in the last cycle, but the xc_create node does not process the output data of the xc_information node.

For example, the ID of the xc_information node is 1000374815. Set Depend On to Instances of Custom Nodes and enter the node ID 1000374815 for the xc_create node.

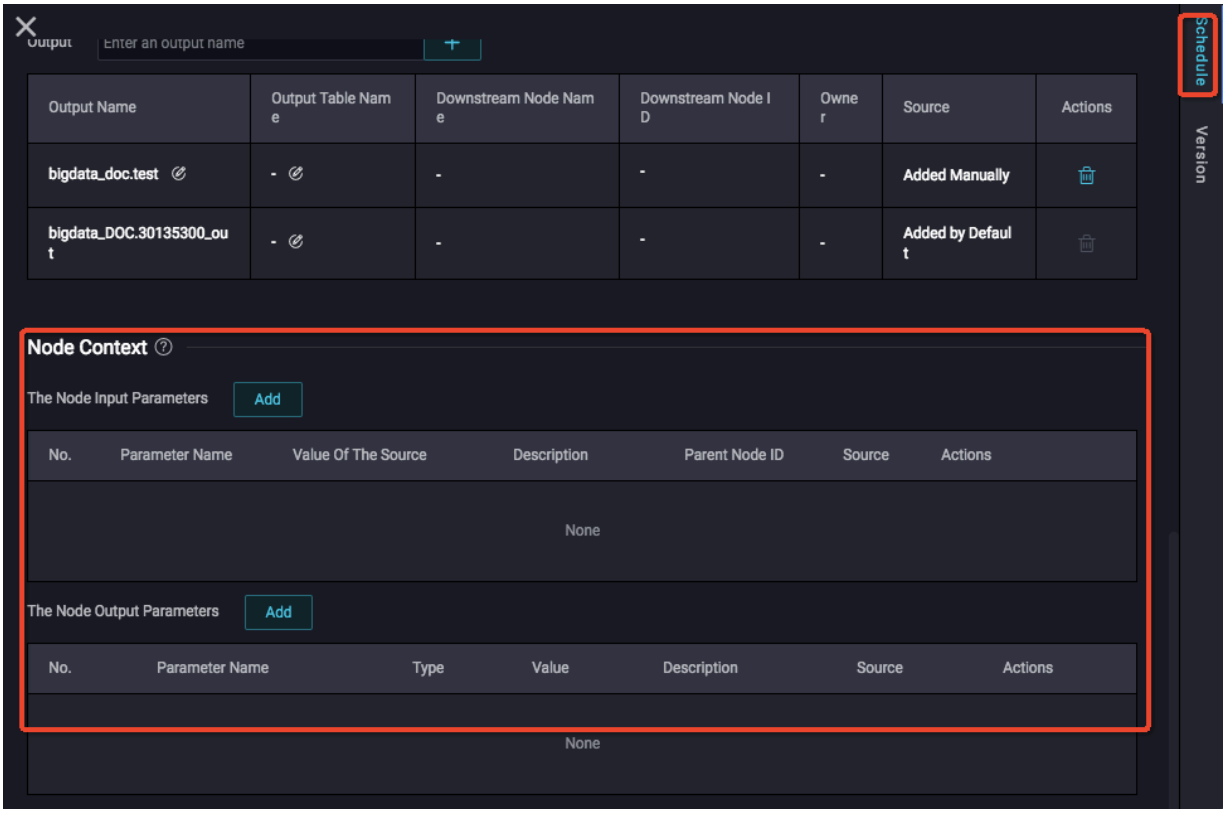
The following figure shows how the dependency is displayed in Operation Center.



1.6.6 Node context

This topic describes the node context functions. The node context is used to transfer the parameter between upstream and downstream nodes. The basic method uses the node context function as the first defined output parameters, and their values on the upstream node. Then the defined input parameter on the downstream node. The value references the output parameters of the upstream node. You can use this parameter in the downstream node to obtain values, which is transferred from the upstream node.

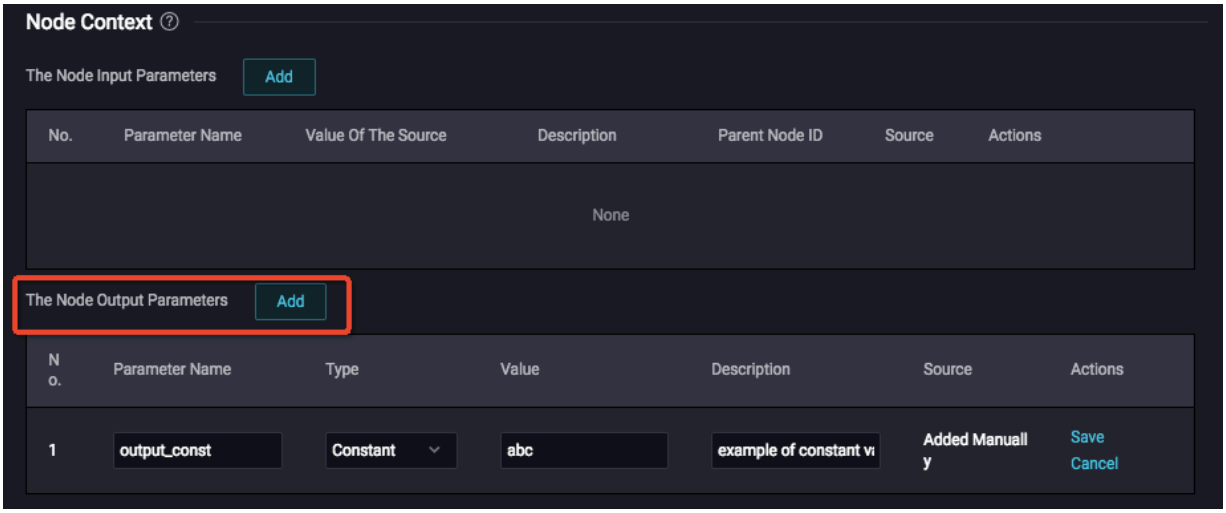
Node context parameter can be configured at Schedule > Node Context in a specific node, as shown in the following figure.



Output parameters

The Node Output Parameters can be defined in Node Context. The two types of Output Parameter values are the Constant and Variable. The Constant is a fixed string and the Variable are global variables supported by the system. The output parameter can be reused in the downstream node as an input parameter value, after the upstream node is submitted with the output parameter.

Note:
The assigned value to the defined Output parameter on the current node through internal code writing, for example the PyODPS node is not supported.



The fields are described as follows.

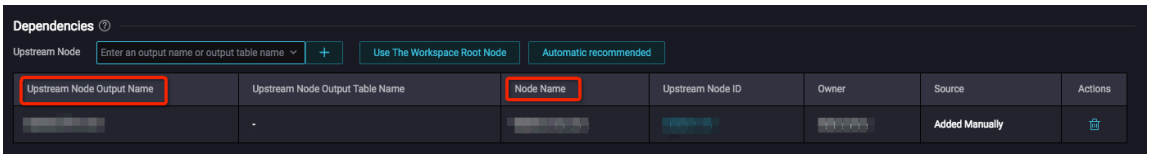
Field	Description	Note
No.	The value of No. is generated by the system and automatically increased.	N/A
Parameter name	The defined output parameter name.	N/A
Type	The parameter type.	There are two types of output parameter values, which are the `Constant` and `Variable`.
Value	The source value.	1. The string can be entered when the selected Type is `Constant`. 2. When the selected type is `Variable`, the following parameters are supported: System variables, Schedule built-in parameters, Customized parameters ${...}and $[...].
Description	A brief description of the parameters.	N/A
Action	Edit and Delete can be selected	Edit and Delete are not supported when a downstream node dependency exists. Before adding references to the upstream nodes, please ensure the upstream output is defined correctly.

Input parameters

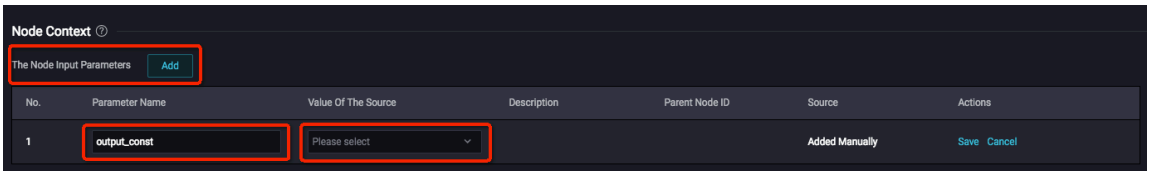
The Node Input Parameters are used for defining a reference to the output of the upstream node which it is dependent on, and it can be used inside the node similar to that as other parameters.

- The definition of The Node Input Parameters

1. Add a dependent upstream node on the Scheduling Dependencies.



2. Add an input parameter definition with value, which references the upstream node, in the Node Context > The Node Input Parameters.



The fields are described as follows.

Field	Description	Note
No.	The value of No. is generated by the system and is automatically increased.	N/A
Parameter name	The defined input parameter name.	N/A
Value of the source	The parameter value source that is a reference to the upstream node value.	The specific parameter value when the upstream node is running.
Description	A brief description of the parameters.	Automatically parsed from the upstream node.
Parent Node ID	Parent Node ID	Automatically parsed from the upstream node.
Action	Edit and Delete can be selected	N/A

- Use of input parameters

  The reuse format defined input parameter is similar to that as to other systems. The format is `${ input parameter name }`. For example, a reference in a shell node is shown in the following figure.

```
echo 'input_from_up_const:' ${input_from_up_const}
echo 'input_from_up_var:' ${input_from_up_var}
```

Global variables supported by the system

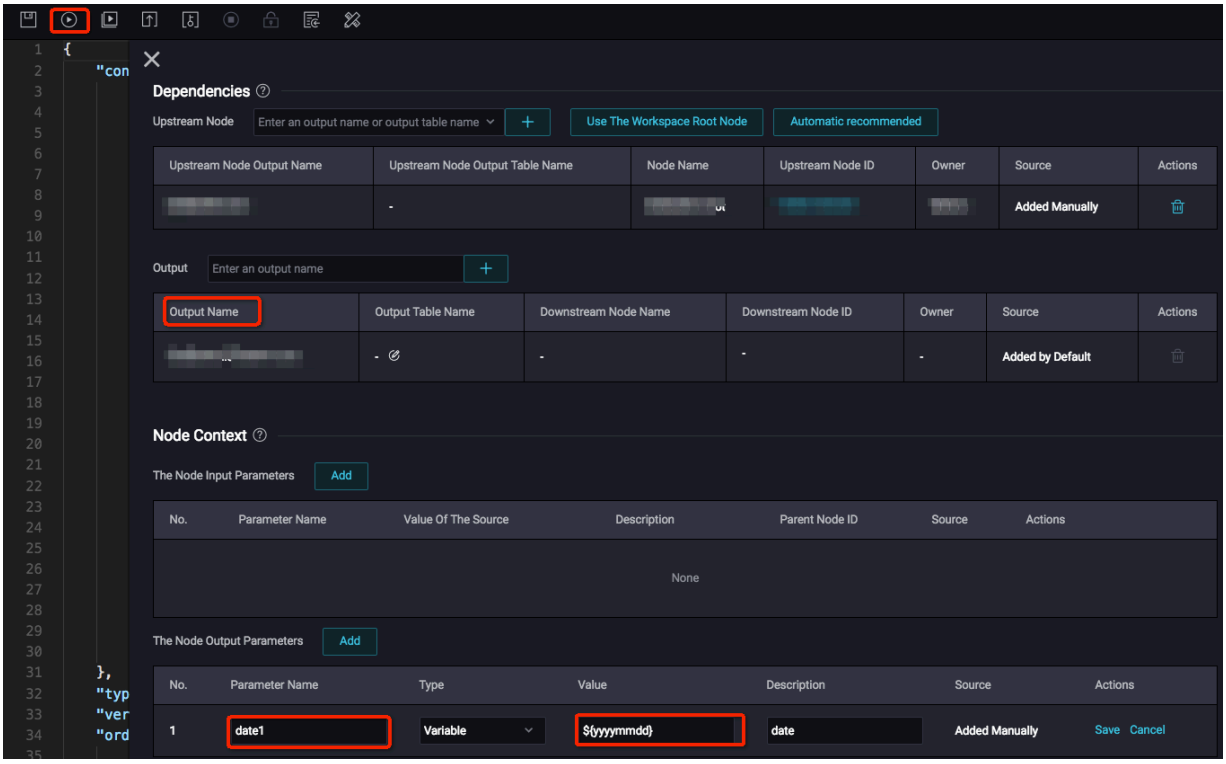
- System variable

```
$ { projectid }: Project ID
$ { project name }: MaxCompute project name
$ { nodeid }: Node ID
$ { gmtdate }: 00 : 00 : 00 at the instance date , format :
 ' yyyy - mm - dd 00 : 00 : 00 '.
$ { taskid }: Instance Task ID
$ { seq }: Task instance sequence number , represents the
 instance ' s sequence number in the same node on
 current day .
$ { cyctime }: instance time
$ { status }: Status of instance - Success , Failure
$ { bizdate }: Business Date
$ { finishtime }: Instance End Time
$ { taskType }: Instance Status —— NORMAL , MANUAL , PAUSE , SKIP
 , UNCHOOSE , SKIP_CYCLE
$ { nodeName }: Node name
```

- For more information about additional parameter settings, see #unique_42.

Examples

The node test22 is the upstream node of node test223. Please configure the Node Context > The Node Output Parameters on node test22. In this example, the parameter name is `date1` and the value is `${ yyyymmdd }`, click Run as shown in the following figure.

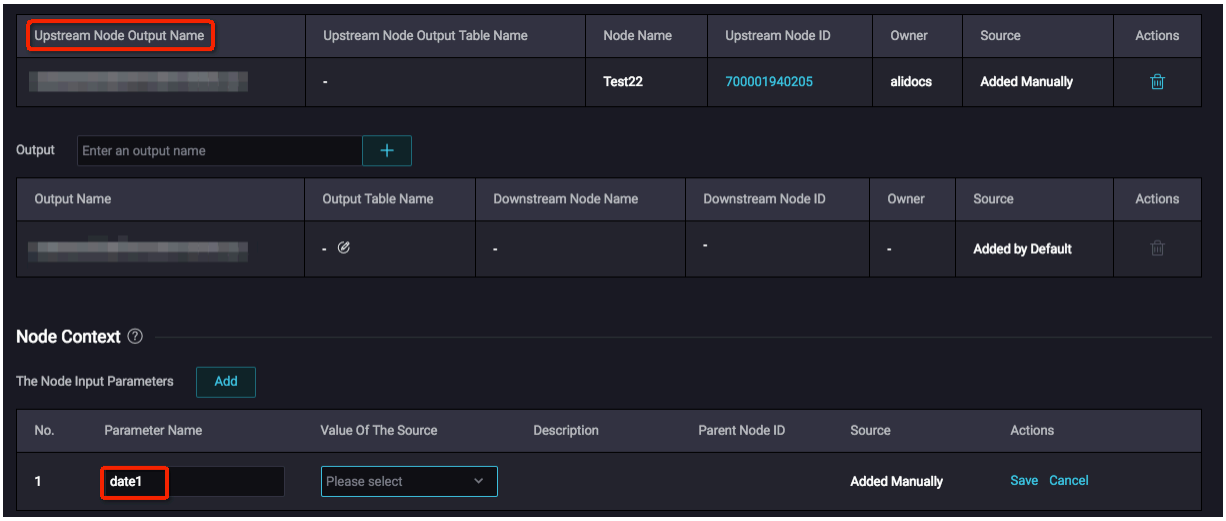


After node test22 is submitted configure the downstream node test223.

Note:

Please ensure the Dependencies > Upstream Node Output Name in test223 similar to Dependencies > Output Name in test22.

Enter the parameter name of test22 `date1` in the Node Context > The Node Input Parameter > Parameter Name, and there will be options available in theValue of the drop-down menu. Choose the specific source and click Save.



1.6.7 Create instances immediately

This topic describes how to immediately create instances from a published node. You can view the instance dependency relationships in the O&M center.

Instance creation methods

You can choose the following methods to create instances from a published node.

- **Next day**

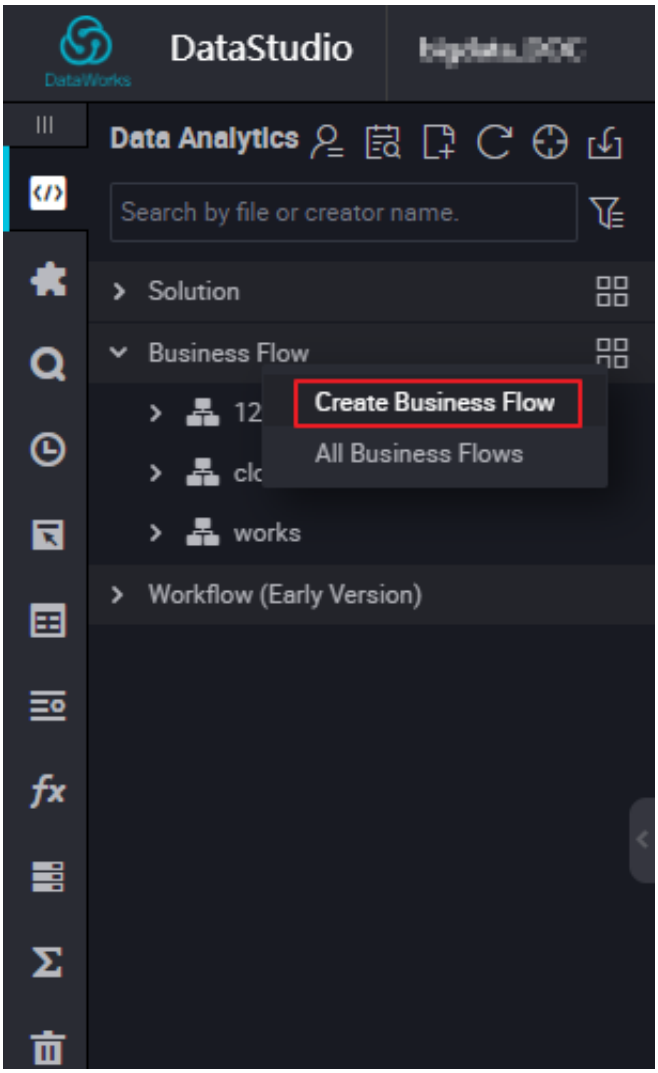
  If you choose this method, the nodes published before 23:30 create instances the following day. The nodes published after 23:30-00:00 create instances three days later.

- **Immediately after publishing**

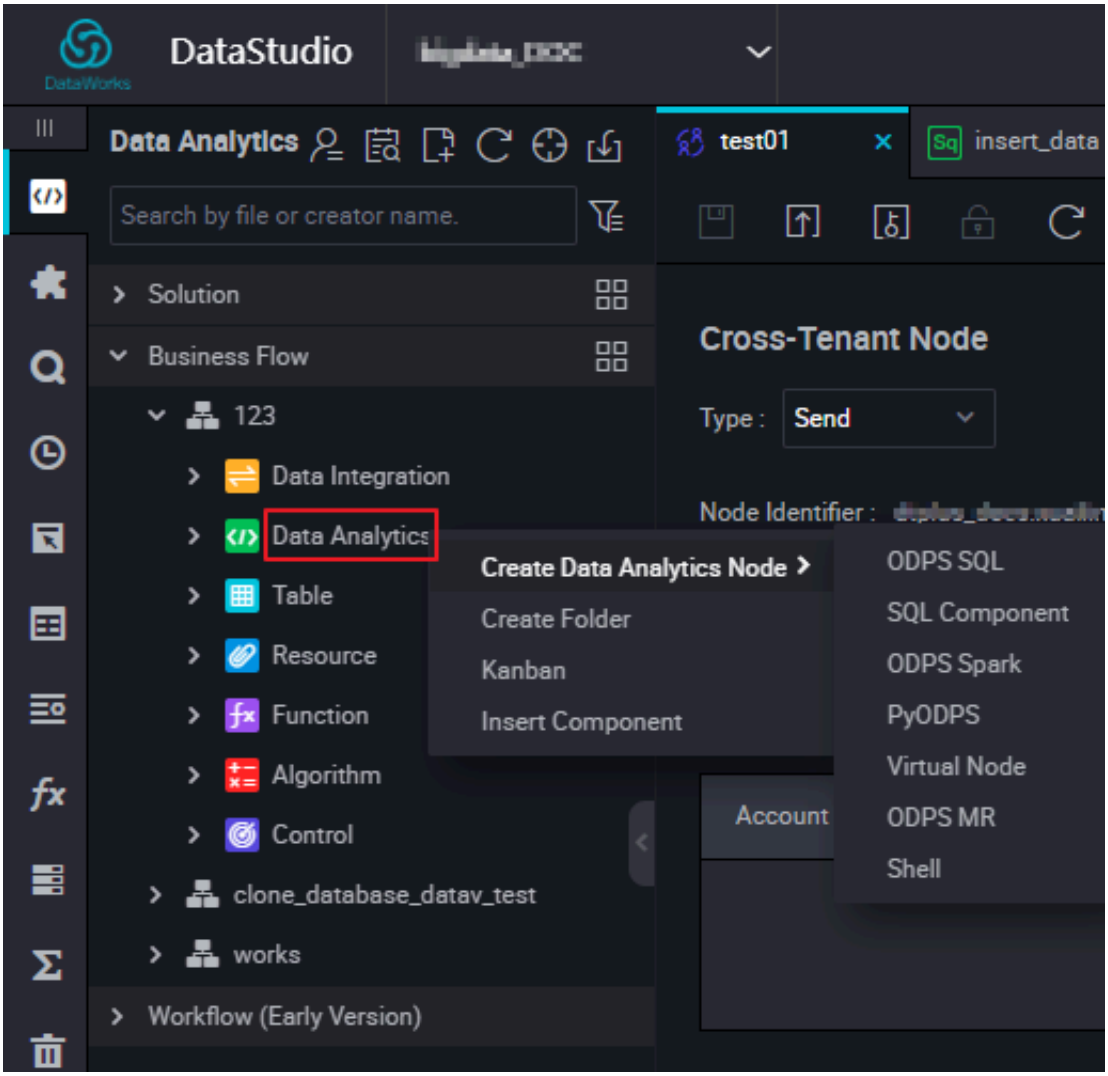
  If you choose this method, the nodes create instances immediately after they are published.

Creating an instance for creating nodes after the node is published

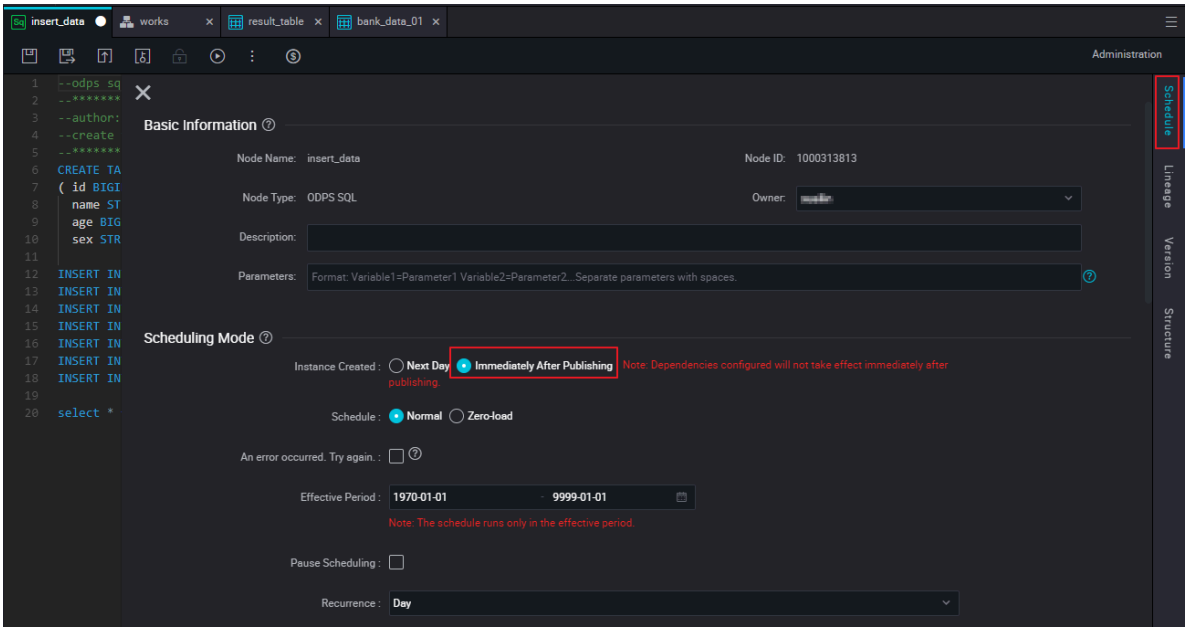
1. On the DataStudio page, create a Business Flow.



2. Create a node in the created business flow. The following example uses an ODPS SQL node.



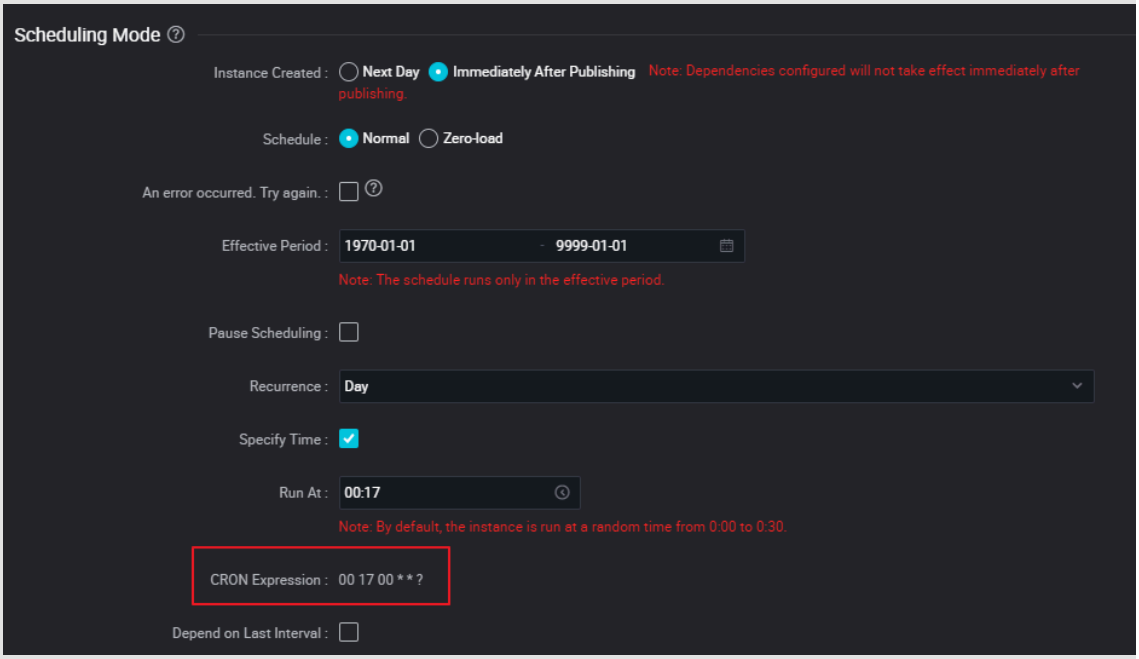
3. Double-click the node, edit the code, and click Schedule on the right-navigation pane of the page. Then set the instance creation method to Immediately After Publishing.



Note:

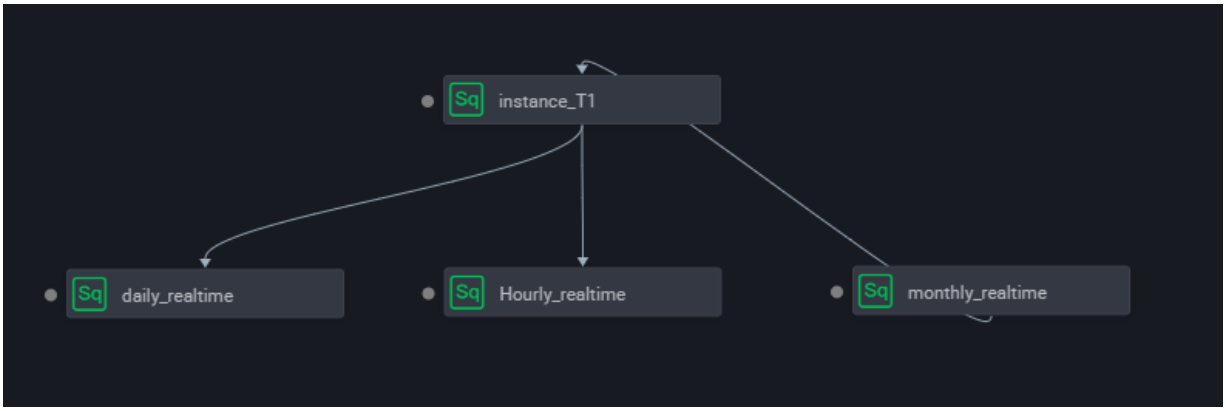
- You can publish the node any time. However, both unpublished and published nodes will not create instances during the time period 23:30 to 24:00.
- After an Immediately After Publishing Node is published, you must wait 10 minutes to create instances.
- If you change the instance creation method from Next Day to Immediately After Publishing to republish the node. Only the instances that have been run are retained. After the node is republished, it will pend 10 minutes before deleting instances that have not been run, and then create now instances.
- An Immediately After Publishing node determines whether to create new instances based on the CRON Expression. If the expression changes, the node then creates new instances. Therefore, if you need to republish Immediately

After Publishing node to create a new instance, you must change the CRON Expression of the node.



Scenarios

The Immediately After Publishing method typically uses the following scenario: The predecessor node uses the Next Day method to create instances. The successor nodes all use the Immediately After Publishing method to create instances. The following figure shows the dependency relationships between these nodes:



This scenario includes the following situations:

1. If the upstream and downstream nodes are new nodes added to today, this means the upstream node must wait until the following day to create the first instance:
   - Daily run upstream nodes: The instance created by the daily run upstream node today does not have an upstream node. If the dependency type is set to custom,

the instance created by the upstream node will depend on an instance created by another node.

- Once a minute or hourly run upstream nodes: If the dependency type is set to upstream-downstream dependency and the upstream node is not once a minute or hourly run node, only the first created instance will not have an upstream node.
- Weekly or monthly run upstream nodes: If the node dependency type is self-dependent, only the first instance created by this node does not have an upstream node.

> Note:
> A daily run upstream node will not create the first instance until the following day. Therefore, instances created by the downstream nodes today will become independent instances without upstream nodes and cannot be run. If the dependency type of the downstream nodes is set to self-dependent, the instances created the next day will depend on those independent instances. As a result, the task is isolated and cannot be run.

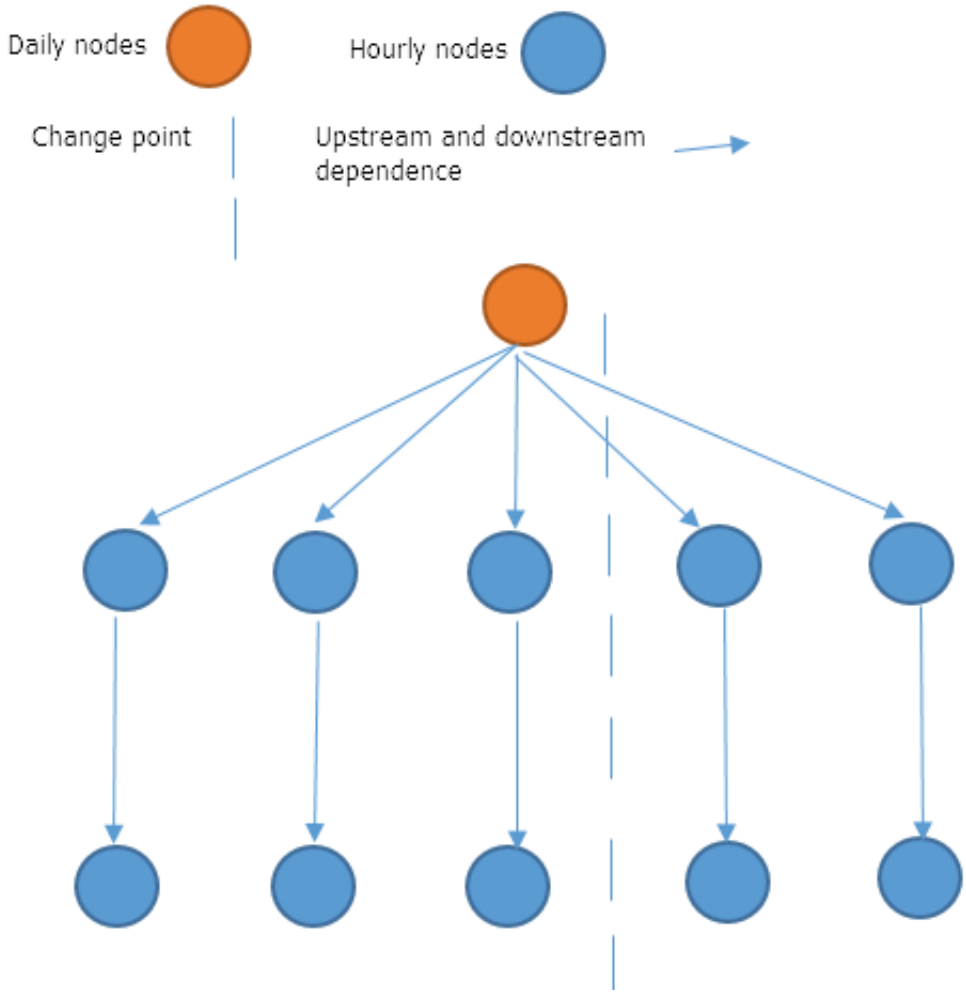
- Conclusion: When the upstream and downstream nodes are new nodes added to the current day and the dependency type is set to self-dependent, the instance created first will not have an upstream instance. As a result, the task cannot run successfully.

2. If the upstream node has created a predecessor instance, and the successor nodes are added Immediately After Publishing nodes:

- Daily run downstream nodes: The instance created by this node today will depend on the existing upstream instance. If the dependency type is set to self-dependent, all instances created by this node will have an upstream instance.
- Once a minute or hourly run downstream nodes:The instance created by this node on the current day will depend on the existing upstream instance. If the dependency type is set to self-dependent, only the first instance does not have an upstream instance.
- Weekly or monthly run downstream nodes: Despite of the set dependency type, the instance created by this node will depend on the existing upstream instance.
- Conclusion:To successfully run a self-dependent node , make sure the node can successfully run on the day before.

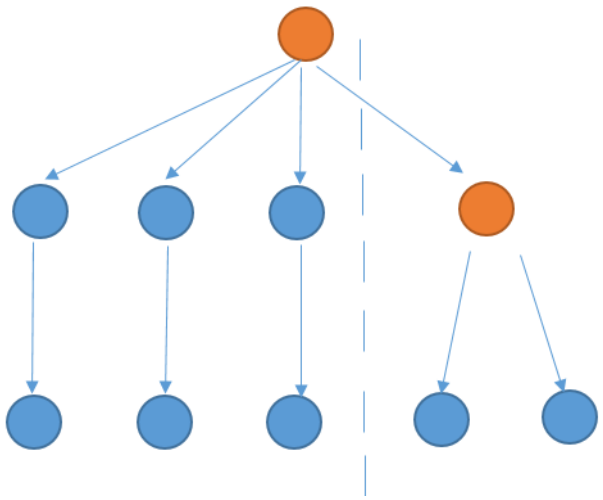
3. If the daily run upstream node has a created instance, and an hourly run upstream node is changed to a daily run node that uses the Immediately After Publishing method:

   - Nodes before modification: Both the upstream and downstream nodes are hourly run nodes that use the Next Day method.



   - Update: The hourly run node that depends on the upstream node is changed to a daily run node that uses the Immediately After Publishing method.
   - Instance creation and dependencies after modification: The dotted line in the preceding figure indicates the time when the node is submitted and republished. The node will delete all instances that are created 10 minutes after the node is republished, and create a new daily run instance. The hourly run successor nodes of the republished node will depend on the newly created daily run instance. If the republished node dependency type is set to self-dependent,

the newly created instance will depend on the instance created by the Next Day node.

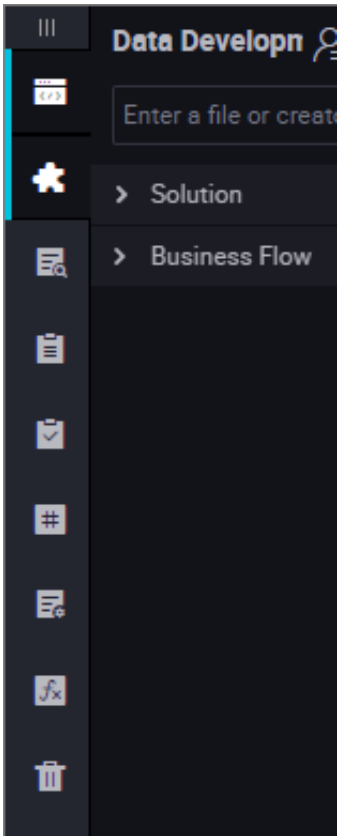


- Instances after modification: Before the node is published, it creates hourly run instances. After the node is republished, it creates daily run instances.
- Conclusion: The dependencies of the republished node remain unchanged. Only the instance created on the current day is affected.

1.7 Configuration management

1.7.1 Overview of configuration management

Configuration management can configure the DataStudio interface, including code , folder, theme, add and delete modules, and more. You can enter the configuration management page by clicking the option in the lower-left corner of data development .

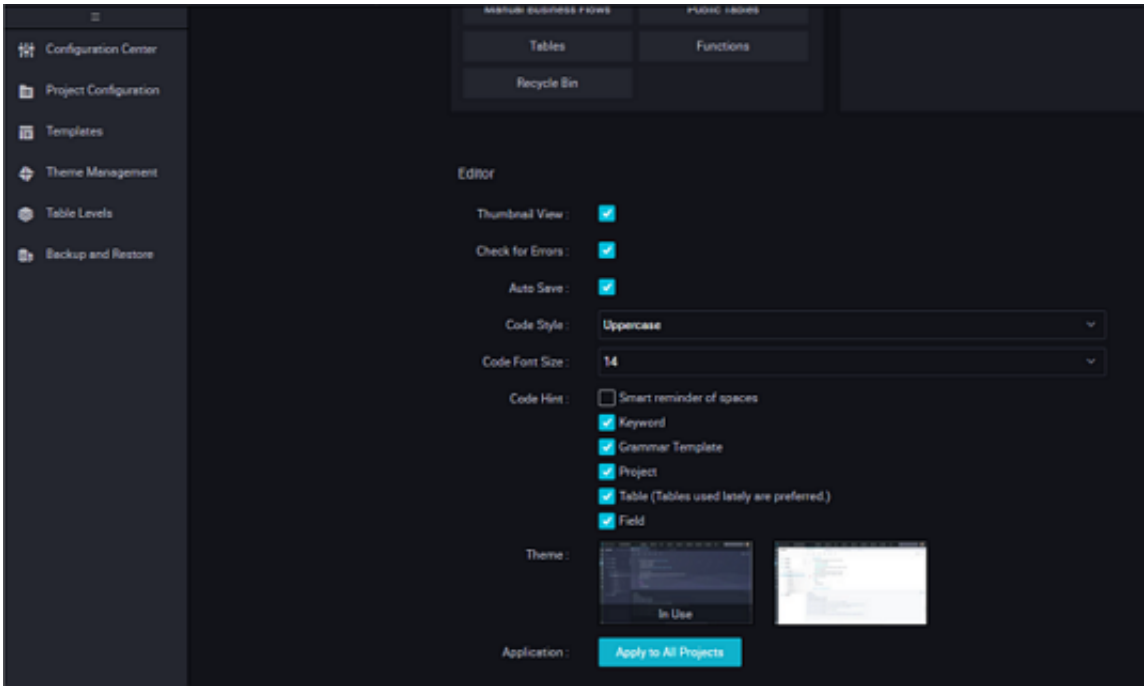


Configuration management is separated into five modules. For more information, see the following documents:

- #unique_84
- #unique_78
- #unique_85
- #unique_86
- #unique_87

1.7.2 Configuration center

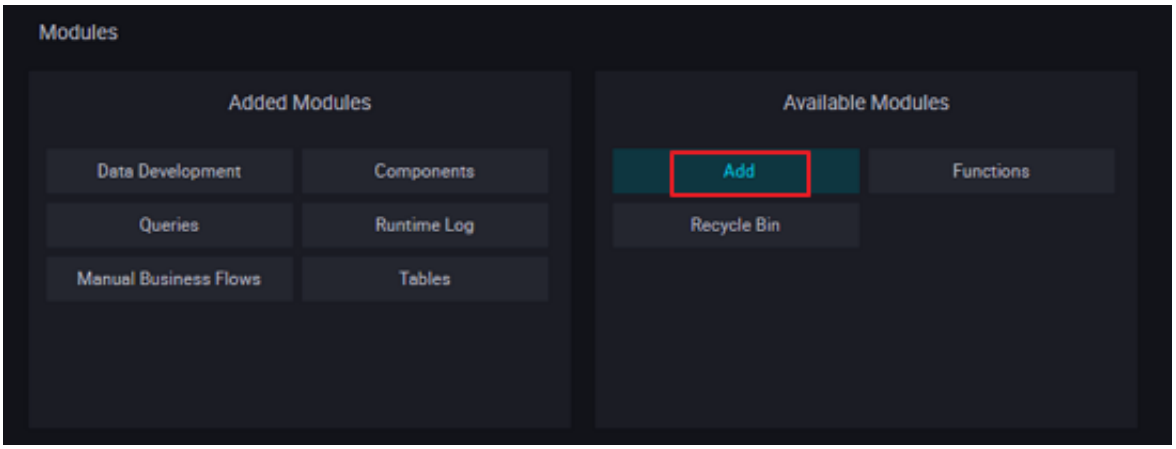
The configuration center sets the common features, including module management and editor management.



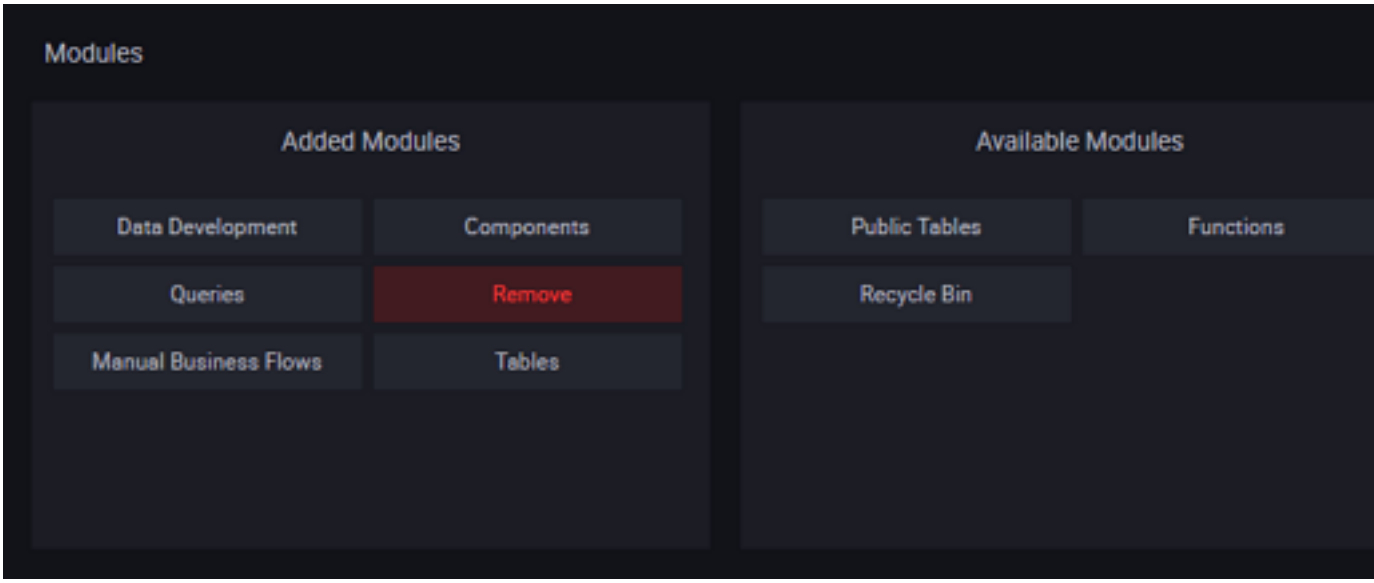
Module management

Module management can add and delete modules in the left-side navigation pane function module of the DataStudio interface, you can click filter to display functional modules on the left-side, you can also sort the module functions by dragging and dropping.

When you move the cursor over the module you want to add, the module turns blue and displays Add.



When the cursor moves over the module that needs to be removed, the module turns red and displays Remove.



Note:

The template management filtering takes effect immediately in the current project, if you want it to take effect for all projects, click Apply Settings To All Projects.

Editor management

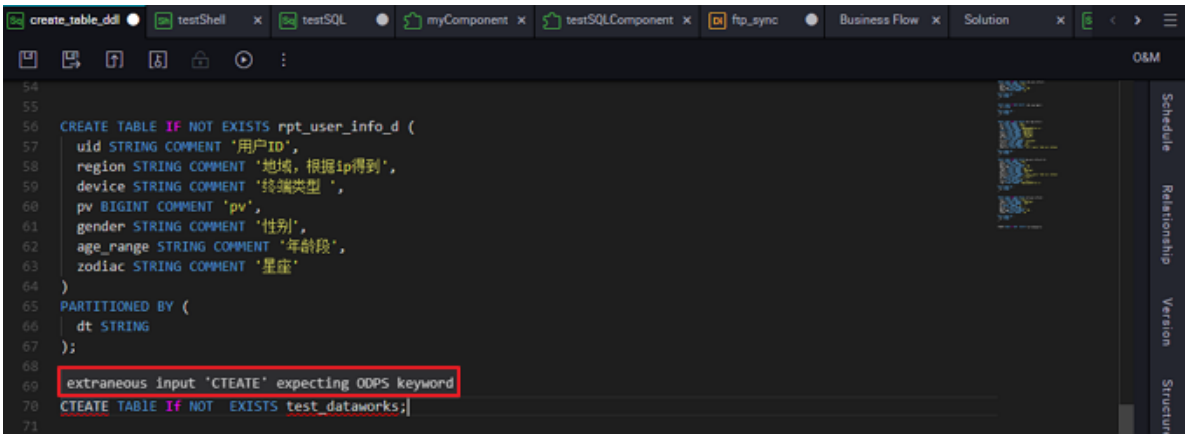
The editor is the setting for code and keywords, and the settings takes effect in real time without the need to refresh the interface.

- Thumbnail view

  The current interface code is displayed on the right side of the code, and the shaded area in the figure is in the displayed area. When the code is longer, you can move the cursor up and down to switch the displayed code area.

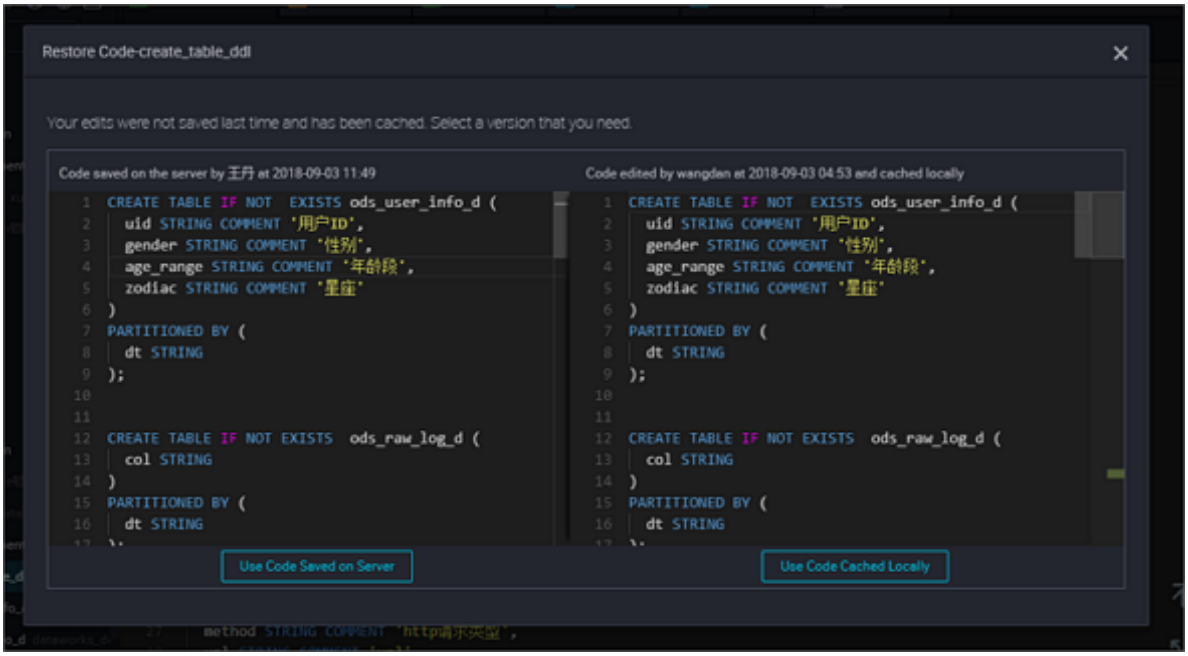
- Check for errors

  Check the error statement in the current code. When the cursor is placed in the red error code area, an error-specific field condition is displayed.



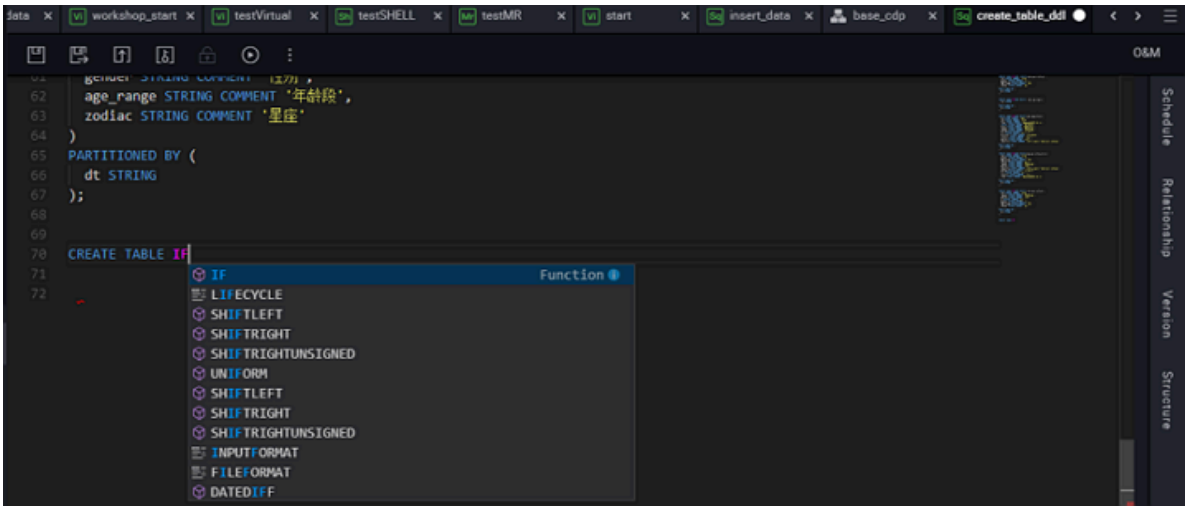
- Auto save

  Automatically cache the currently edited code to avoid the page from crashing and losing edited code that has not been saved. You can choose Use Server-Saved Code in the left-side navigation pane or Use Locally Cached Code in the right-side navigation pane.



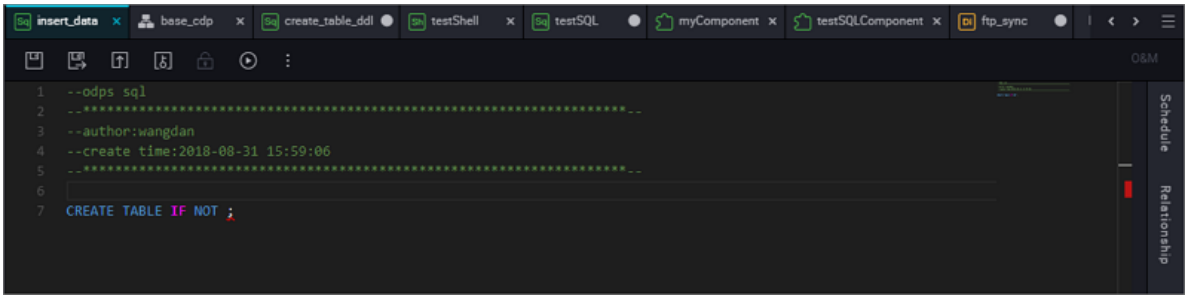
- Code style

  You can select a favorite code style of either uppercase or lowercase. You can enter keywords or use the keyword association shortcut to enter the required keywords.



- Code font size

  The code font size supports a minimum font size of 12 and a maximum font size of 18. You can change the setting based on your code writing habits and volume.



- Code Hint

  Code prompts are used during code entry, and intelligent prompt displays are separated into the following sections.

  - Space Smart Tip: Add a space after selecting associated keywords, tables, and fields.
  - Keywords: The prompt code supports the keywords entered.
  - Syntax templates: The syntax templates are supported.
  - Project: The associated project name.
  - Table: The required table for association.
  - Field: The smart prompt for table fields.

- Theme

  The theme style is the DataStudio interface style setting, which currently supports both black and white.

- Application

  Apply the above template management and editor management settings to all currently existing projects.

1.7.3 Project configuration

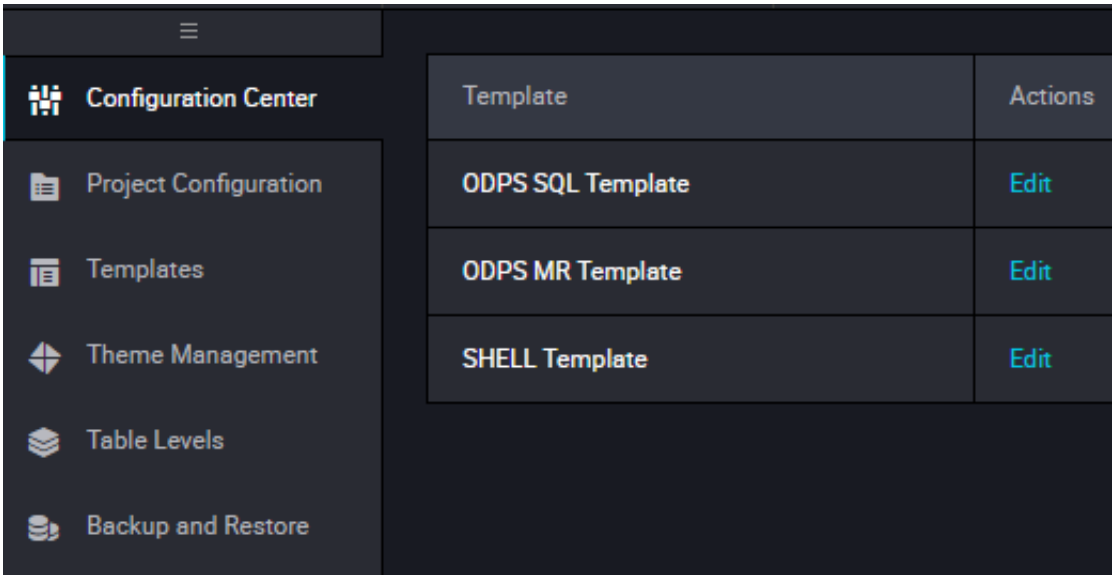
Project configuration includes the following four configuration items: partition date format, partition field naming, temporary table prefix, and upload table (import table) prefix.

- Partition Date Format: By default, this is the display format of the code parameters . You can modify the parameters format based on requirements.
- Partition field naming: The default field name of the partition.
- Temporary table prefix: The fields that begin with "t_" are identified as temporary tables by default.
- Upload table (import table) prefix: The table name prefix when the DataStudio interface uploads the table.

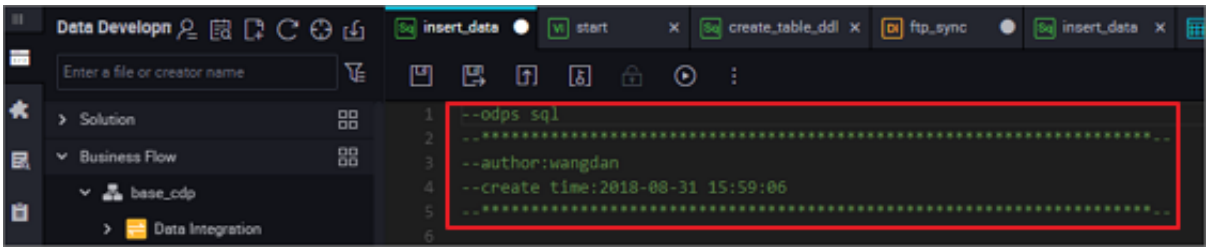
1.7.4 Templates

By default, the template management is the content that is displayed in front of the code after the node is created. The project administrator can modify the template display style as required.

Currently, the title is set for the ODPS SQL template, the ODPS MR template, the ODPS PL template, the PERL template, and the SHELL template.

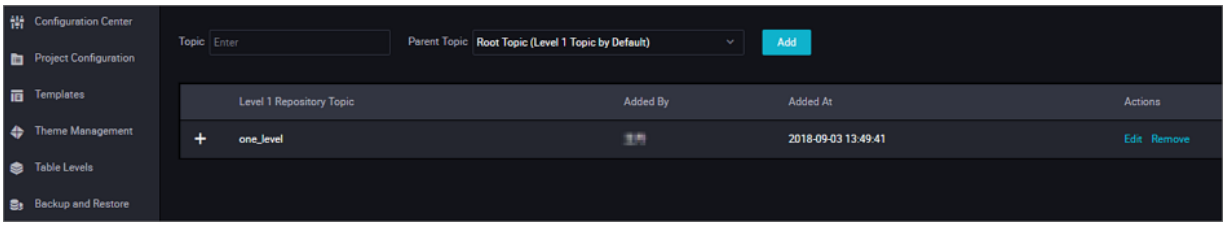


The following is an example of the SQL node template display style:



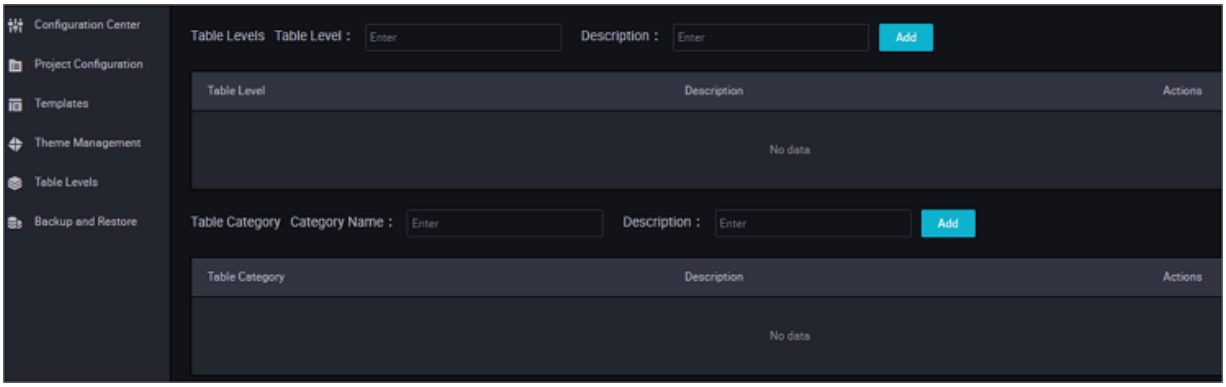
1.7.5 Theme management

This topic is an overview of theme management. There are many tables in table management, where the tables are stored under the second-level sub-Folder according to the selected topics. These folders are summarized in the table, which is the theme. The administrator can add multiple themes based on project requiremen ts, classify and organize the tables according to their purpose and name.



1.7.6 Table levels

This topic is a description of table levels. Table levels is the physical level design of a table. Based on the importance of the table in the project, the table is separated to prevent issues from when a problem occurs in a table, the impact on the project cannot be accurately located, which affects normal online operation.



There is no default hierarchy for the project. The project owner or administrator needs to be added manually according to the purpose and project requirements.

1.7.7 Back up and restore data

This topic describes how to back up and restore data. When you back up your data, your resources are also backed up at the same time. For more information about resources, see Resource.

Note:

- Only workspace administrators can export backups and restore data from backups on the Config Center page. For more information about how to open the Config Center page, see Overview of configuration management.
- Workflows of earlier versions cannot be backed up. We recommend that you use the latest version for data analytics.

You can create both full backups and incremental backups for a workspace. You can select Alibaba Cloud Version 2, Apsara Stack Version 3.6.1 or Later, or Apsara Stack Version 3.6 or Earlier as the version of each backup.

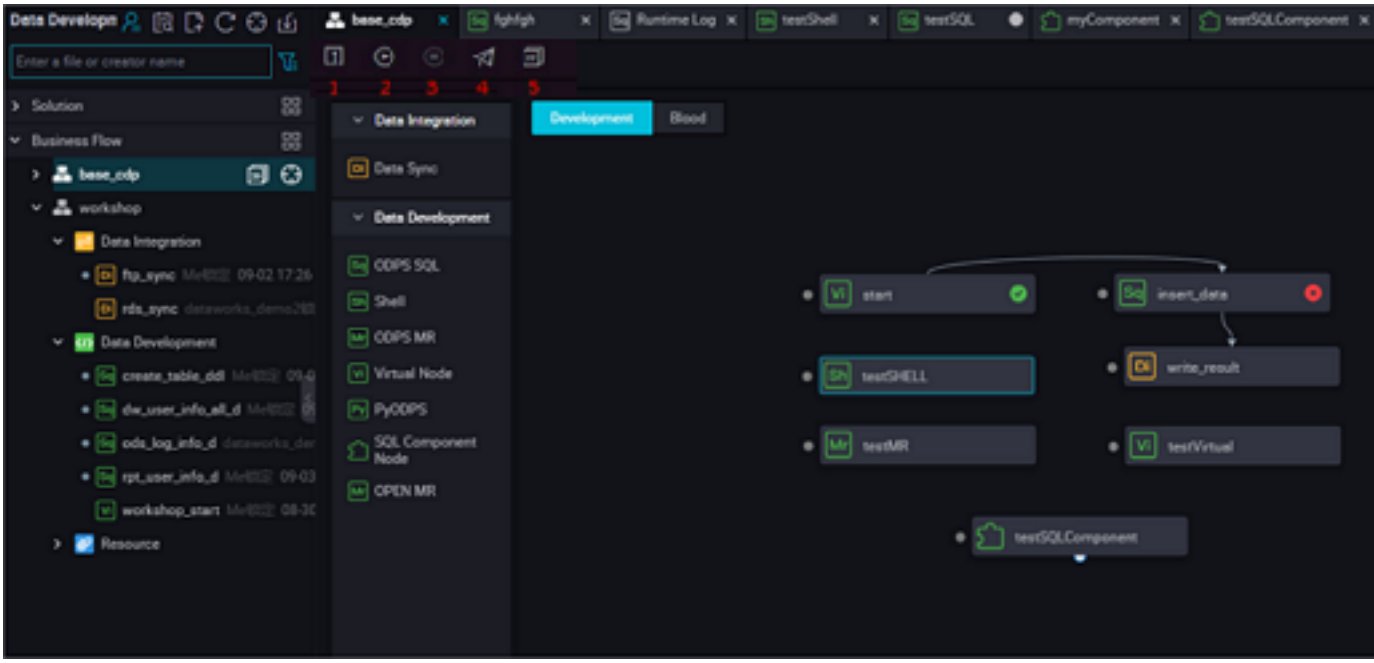
Note:

- You can download backup files in XML format.
- You can restore a workspace from backup files. However, errors may occur during restoration. We recommend that you use full backups whenever possible.

1.8 Manual business flow

1.8.1 Manual business flow overview

In a Manual Business Flow, all created nodes must be manually triggered and cannot be executed by scheduling. Therefore, it is unnecessary to configure the parent node dependency and local node output for nodes in a manual business flow.



The functions of the manual business flow interface are described below:

No.	Function	Description
1	Submit	Submits all nodes in the current manual business flow.
2	Run	Runs all nodes in the current manual business flow. Because the dependency does not exist among manual tasks, these tasks will run concurrently.
3	Stop run	Stops a running node.
4	Publish	Goes to the task publish interface, where you can publish some or all submitted nodes , but does not publish to the production environment.
5	Go to O&M	Goes to the O&M center.
6	Reload	Reloads the current manual business flow interface.

No.	Function	Description
7	Auto layout	Automatically sequence the nodes in the current manual business flow.
8	Zoom-in	Zoom-in the interface.
9	Zoom-out	Zoom-out the interface.
10	Query	Query a node in the current manual business flow .
11	Full screen	Shows nodes in the current manual business flow in full-screen mode.
12	Parameters	Configures the parameters. The priority of a flow parameter is higher than that of a node parameter . If a parameter key matches a parameter, the business flow parameter is configured preferenti ally.
13	Operation records	Views the operation history of all nodes in the current manual business flow.
14	Version	Views the submission and published records of all nodes in the current manual business flow.

1.8.2 Resource

This topic is an overview of resource in Manual Business Flow. Resource is a unique concept in MaxCompute, which supports uploading and submitting the Manual Business Flow, and must be available if you want to use MaxCompute UDFs or MaxCompute MR.

- ODPS SQL UDF: After compiling a UDF, you must upload the compiled JAR package to ODPS. When running this UDF, ODPS automatically downloads the JAR package , extracts the user code, and runs the UDF. The process of uploading the JAR package is creating a resource in ODPS. The JAR package is a type of ODPS resource .
- ODPS MapReduce: After compiling a MapReduce program, you must upload the compiled JAR package as a resource to ODPS. When running a MapReduce job, the MapReduce framework automatically downloads this JAR resource and extracts the user code.

Similarly, you can upload text files, ODPS tables, and various compressed packages, such as .zip, .tgz, .tar.gz, .tar, and .jar as different types of resources to ODPS. Then, you can read or use these resources when running UDFs or MapReduce.

The ODPS provides reading and using resources for APIs. The following types of ODPS resources are available:

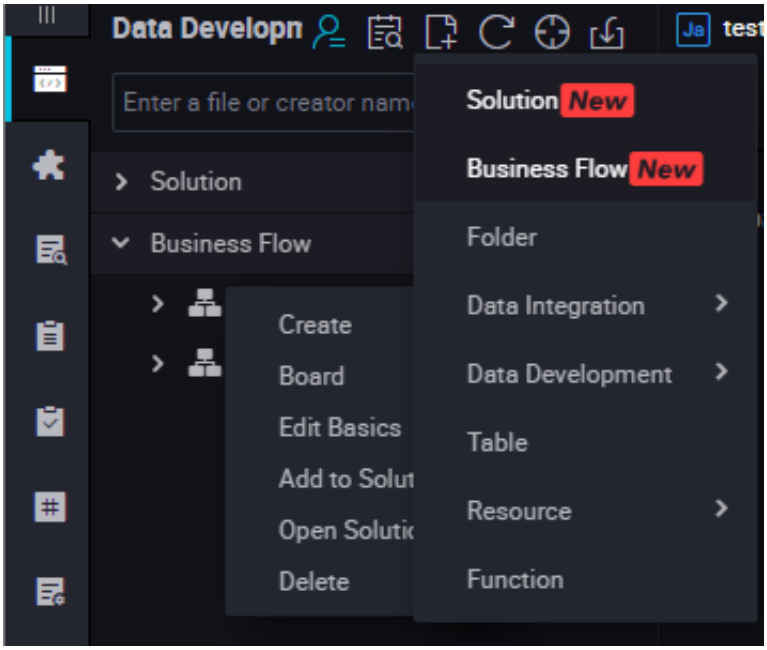
- File
- Archive: The compression type is identified by the extension in the resource name. The following compressed file types are supported: .zip, .tgz, .tar.gz, .tar, and .jar.
- JAR: The compiled Java JAR packages.

In DataWorks, you can add a resource by creating a resource. Currently, DataWorks supports the addition of three types of resources in a visual manner, including JAR, Python, and file resources. The created new entries are the same, but the differences are as follows:

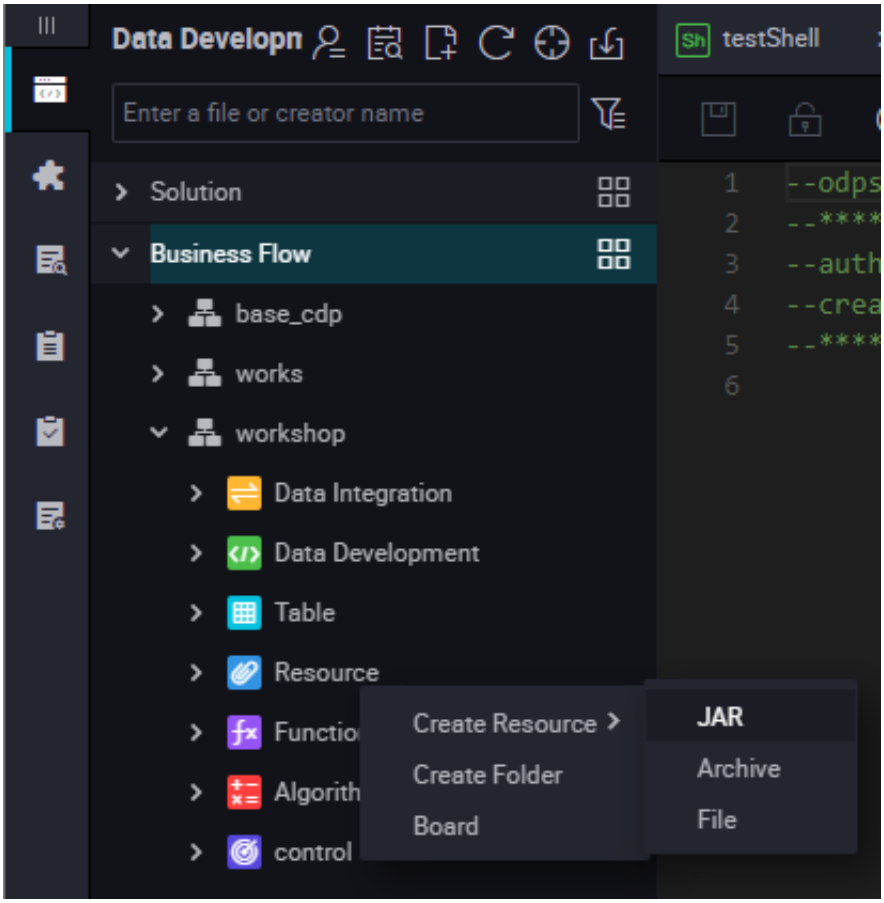
- JAR resource: You need to compile the Java code in the offline Java environment, compress the code into a JAR package, and upload the package as the JAR resource to MaxCompute.
- Small files: These resources are edited on DataWorks.
- File resource: When creating file resources, you need to select big files. You can also upload local resource files.

Create a resource instance

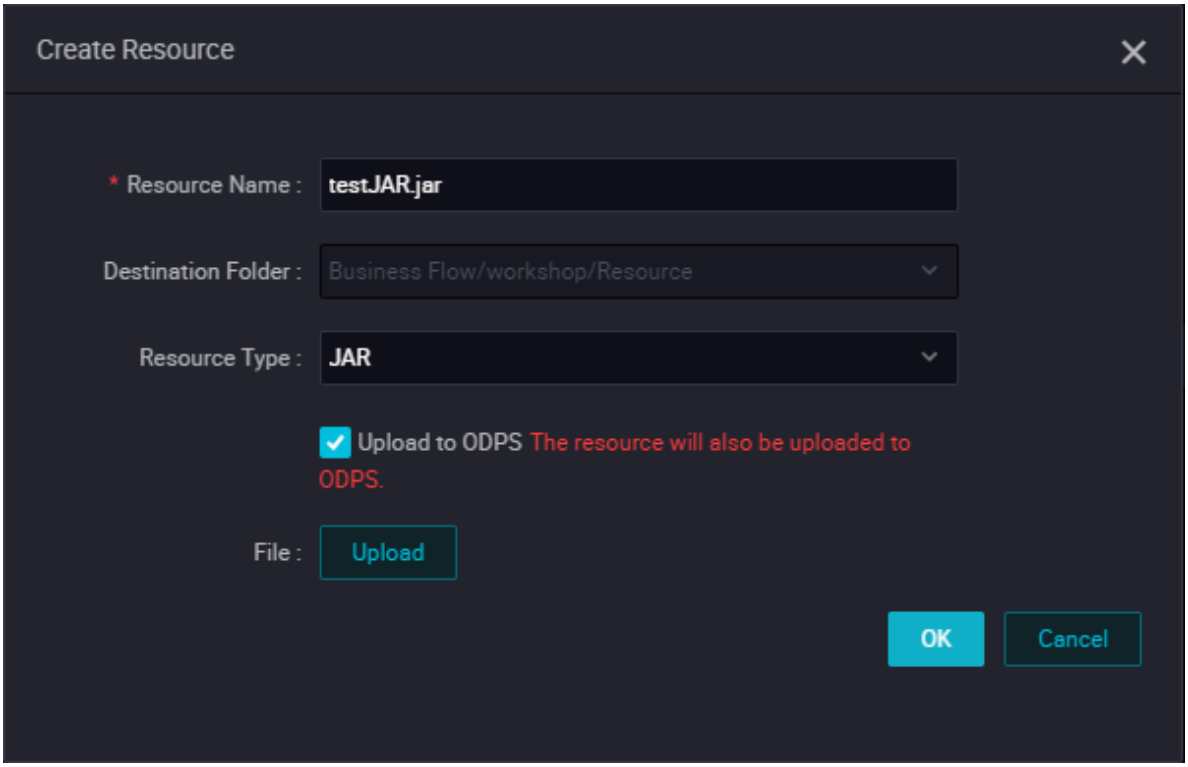
1. Click Manual Business Flow in the left-side navigation pane, and select Create Business Flow.



2. Right-click Resource, and select Create Resource > JAR.



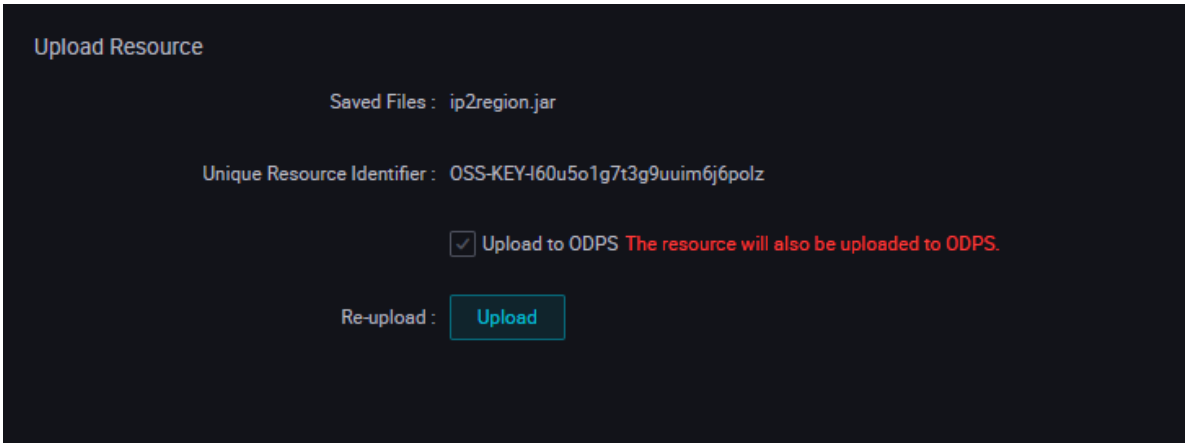
3. The Create Resource dialog box is displayed. Enter the resource name according to the naming convention, set the resource type to JAR, select a local JAR package to upload, and click Submit to submit the package in the development environment.



Note:

- If this JAR package has been uploaded to the ODPS client, you must deselect Upload to ODPS resource. Otherwise, an error will be reported during upload.
- The resource name is not necessarily the same as the uploaded file name.
- Naming convention for a resource name: A string of 1 to 128 characters, including letters, numbers, underscores (_), and periods (.). The name is case insensitive. If the resource is a ()wewweJAR resource, the file extension is .jar.

4. Click Submit to submit the resource to the development scheduling server.



5. Release a node task

   For more information about the operation, see #unique_19.

1.8.3 Function

Register the UDF

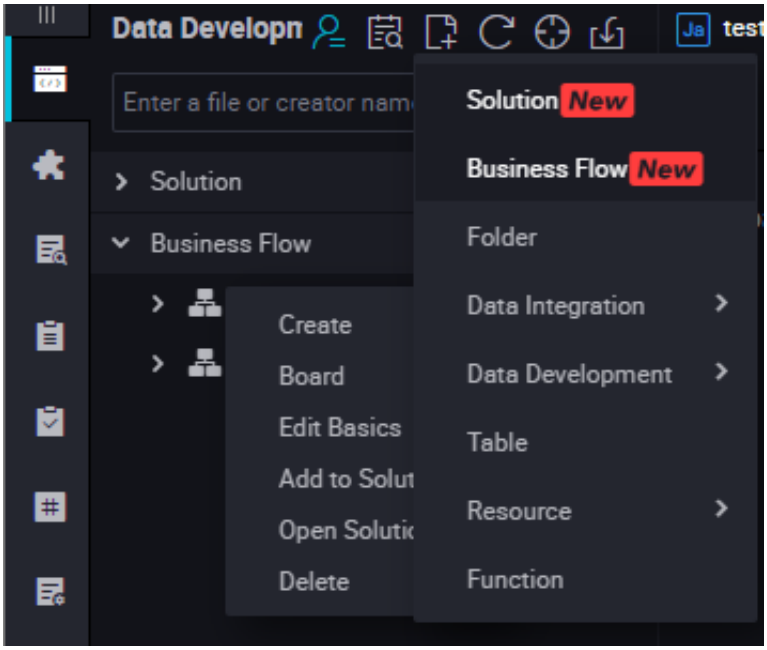
MaxCompute supports UDFs. For more information, see UDF overview.

DataWorks provides the visual GUI to register functions for replacing the MaxCompute command line `add function .`

Currently, the Python and Java APIs supports the implementation of UDF. To compile a UDF program, you can upload the UDF code by Adding resources and then register the UDF.

UDF registration procedure

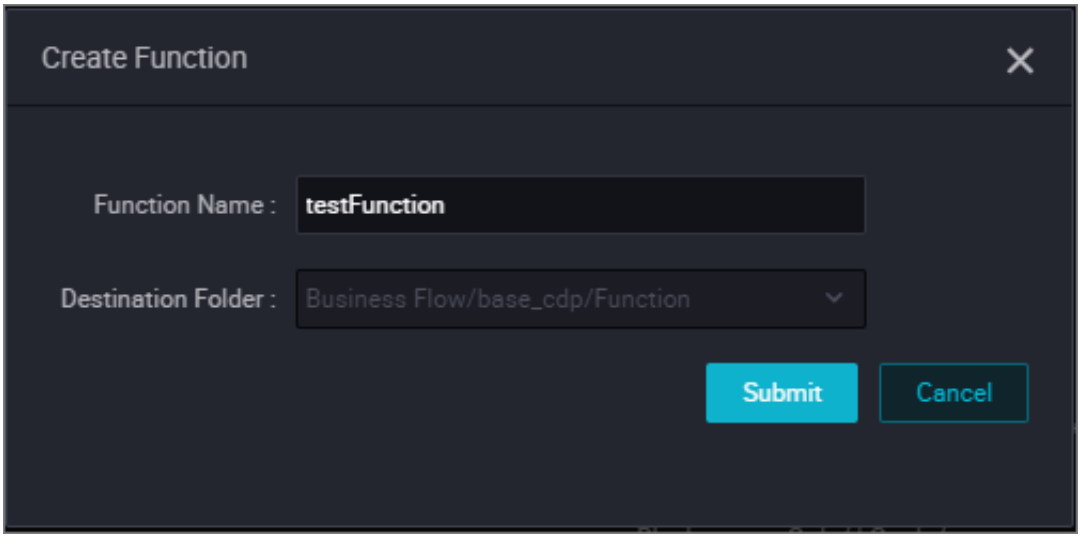
1. Click Manual Business Flow in the left-side navigation pane, select Create Business Flow.



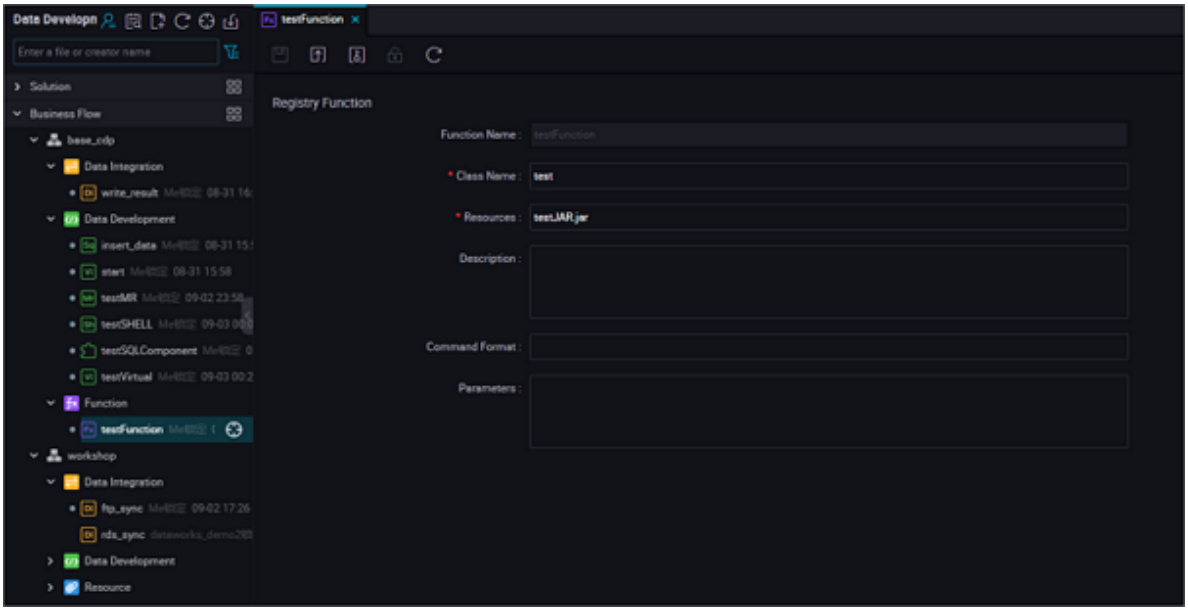
2. In the offline Java environment, you can edit the program, and compress the program into a JAR package. Then create a JAR resource, submit and publish the program.

   You can also create a Python resource. You can compile and save the Python code, and submit the code, and then publish the code. For more information, see Create Resources.

3. Select Function > Create Function, and enter the new function configuration, and then click Submit.



4. Edit the function configuration.



   - Class Name: The name of the main class that implements the UDF. When the resource is Python, the typical writing style is: Python resource name.Class name ('.py' is not required in the resource name).
   - Resources: The name of the resource in the second step. If there are multiple resources, separate them with commas (,).
   - Description: The UDF description. It is optional.

5. Submit the job.

   After the configuration is completed, click Save in the upper-left corner of the page or press Ctrl+S to submit (and unlock) the node to the development environment.

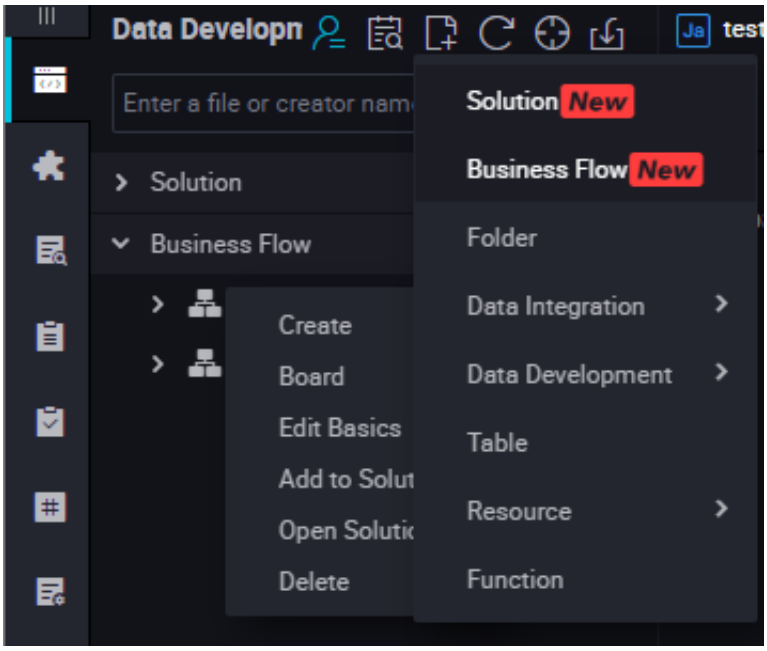
6. Publish a node task

   For more information about publishing a node task, see #unique_19.

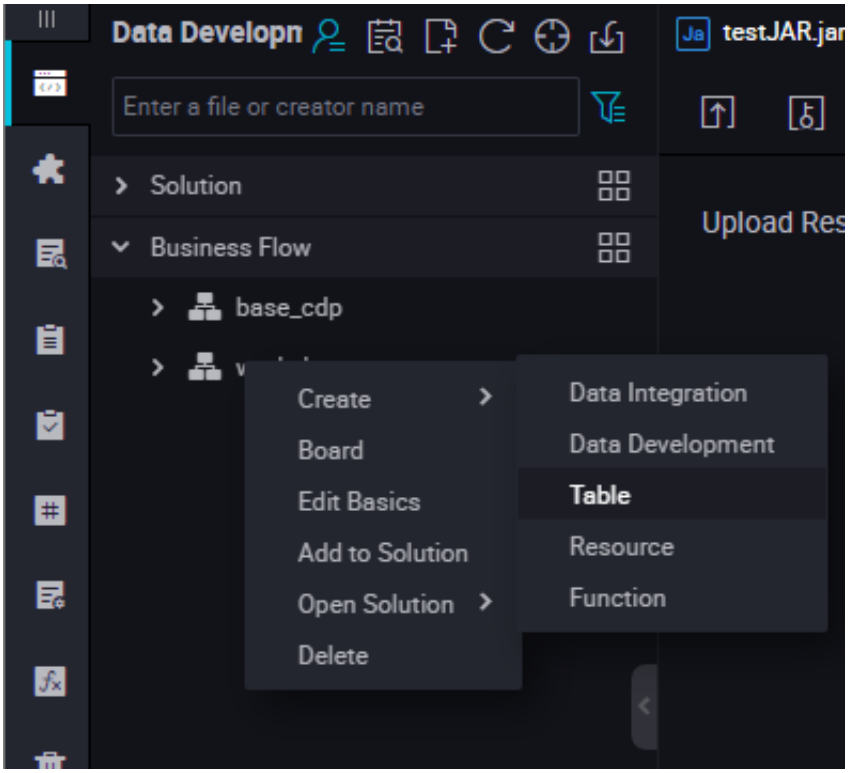
1.8.4 Table

Create a table

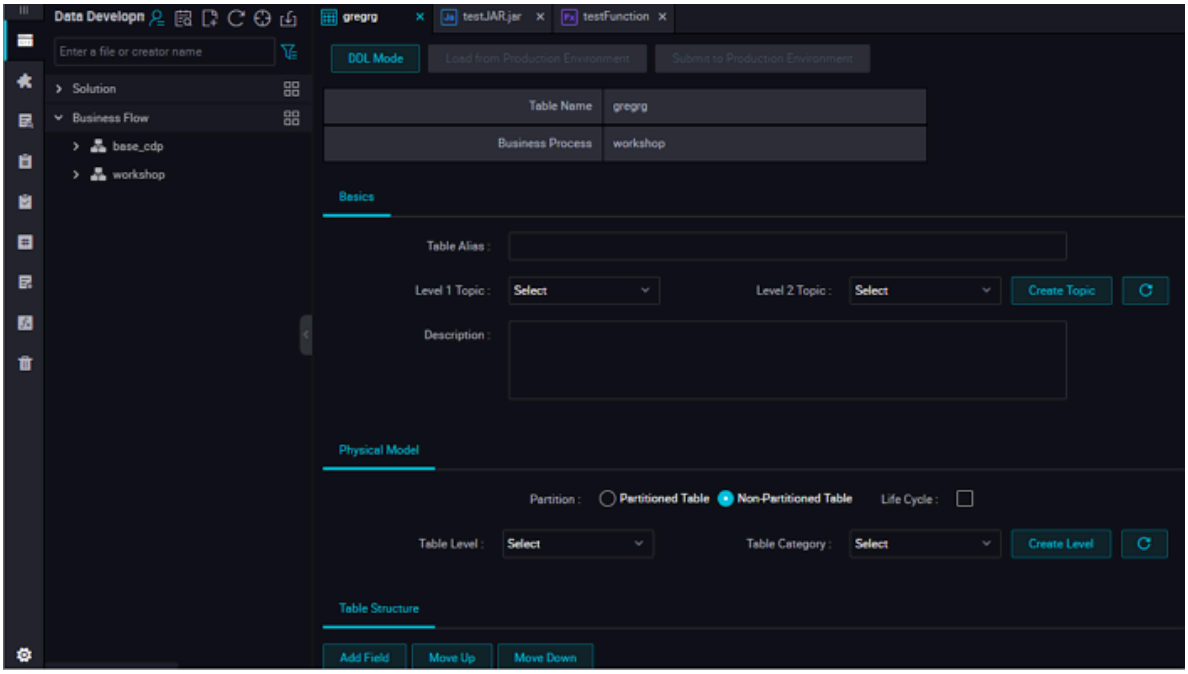
1. Click Manual Business Flow, and select Create Business Flow.



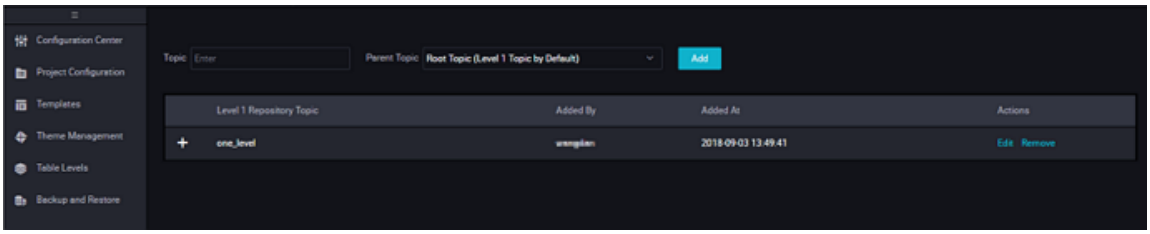
2. Right-click Table, and select Create Table.



3. Set basic attributes.

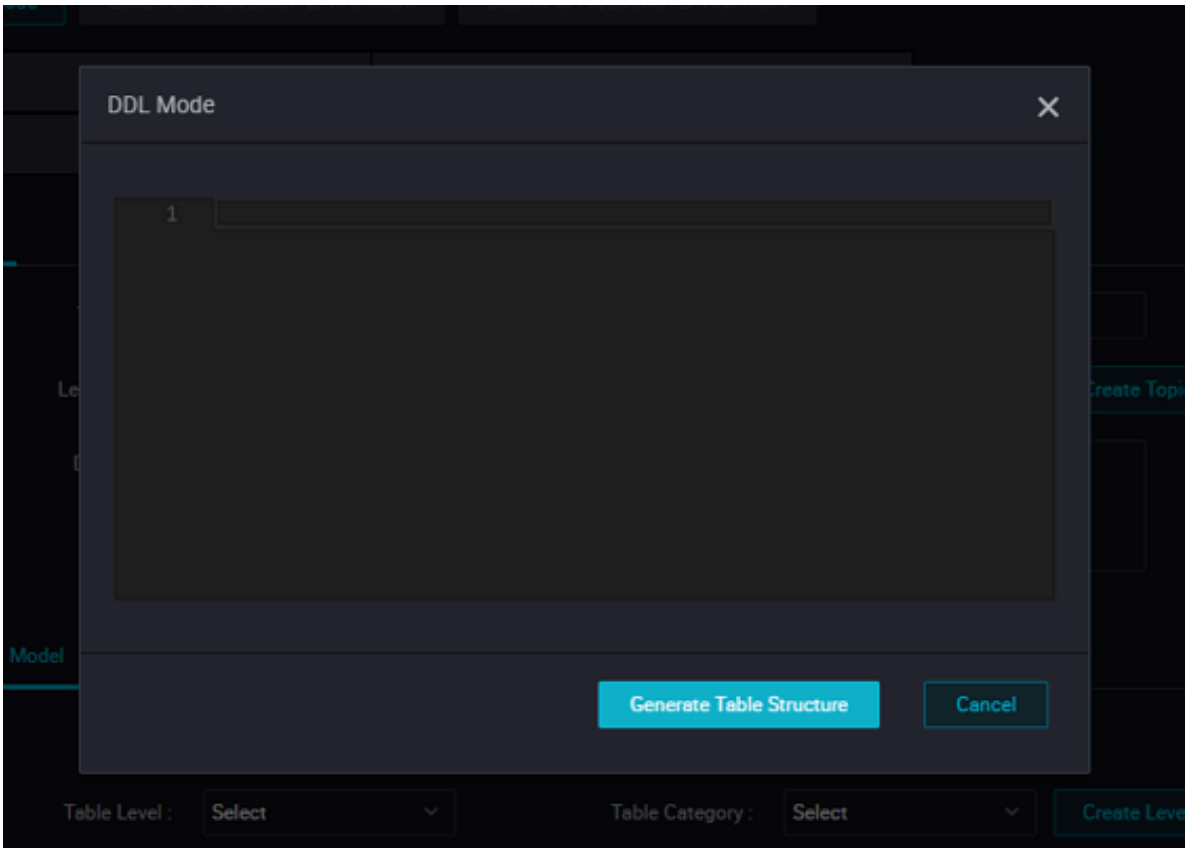


- Chinese Name: The Chinese name of the created table.
- Level-1 Topic: The name of the level-1 target folder of the created table.
- Level-2 Topic: The name of the level-2 target folder of the created table.
- Description: The description of the created table.
- Click Create Topic. On the displayed Topic Management page, create level-1 and level-2 topics.



4. Create a table in DDL mode

   Click DDL Mode. In the displayed dialog box, enter the standard table creation statements.



   After editing the table creation SQL statements, click Generate Table Structure to automatically enter information in the Basic Attributes, Physical Model Design, and Table Structure Design areas.

5. Create a table on the GUI

   If creating a table in DDL mode is not applicable, you can create the table on the GUI by performing the following settings.

   · Physical model design

   - Partition Type: It can be set to Partitioned Table or Non-partitioned Table.
   - Life Cycle: The life cycle function of MaxCompute. Data in the table (or partition) that is not updated within a period specified by the Life Cycle (unit: day) will be cleared.
   - Level: It can be set to DW, ODS, or RPT.
   - Physical Category: It can be set to Basic Business Layer, Advanced Business Layer, or Other. Click Create Level. On the displayed Level Management page, create level.

   · Table structure design

   - English Field Name: The English name of a field can contain letters, numbers , and underscores (_).
   - Chinese Name: The abbreviated Chinese name of a field.
   - Field Type: The MaxCompute data type, which can only be String, Bigint, Double, Datetime, or Boolean.
   - Description: The detailed description of a field.
   - Primary Key: Select this parameter to indicate the field is the primary key or a field in the joint primary key.
   - Click Add Field to add a column for a new field.
   - Click Delete Field to delete a created field.

      Note:
     If you delete a field from the created table and submit the table again, you must drop the current table and create one with the same name. This operation is not allowed in the production environment.

   - Click Move Up to adjust the field order of the created table. However, to adjust the field order of a created table, you must drop the current table and create

one with the same name. This operation is not allowed in the production environment.

- Click Move Down, so the operation is the same as that of Move Up.
- Click Add Partition to create a partition for the current table. To add a partition to the created table, you must drop the current table and create one with the same name. This operation is not allowed in the production environment.
- Click Delete Partition to delete a partition. To delete a partition from a created table, you must drop the current table and create one with the same name. This operation is not allowed in the production environment.
- Action: You can confirm to submit a new field, delete a field, and edit more attributes.

  More attributes include information related to the data quality, which is provided for the system to generate the verification logic. They are used in scenarios, such as data profiling, SQL scan, and test rule generation.

  ■ 0 Allowed: If it is selected, the field value can be zero. This option is applicable only to Bigint and Double fields.

  ■ Negative value allowed: If it is selected, the field value can be a negative number. This option is applicable only to Bigint and Double fields.

  ■ Security level: It can be set to Non-sensitive, Sensitive, or Confidential.

```
C :  Customer   data ,  B :  Company   data ,  S :  Business
  data
C1 – C2 ,  B1 ,  and   S1   are   non - sensitive   data .
C3 ,  B2 - B4 ,  S2 ,  and   S3   are   sensitive   data .
C4 ,  S4 ,  and   B4   are   confidenti  al   data .
```

  ■ Unit: The amount unit, which can be in dollars or cents. This option is not required for fields unrelated to the amount.

  ■ Lookup table name/key value: It is applicable to enumerated value-type fields, such as the member type and status. You can enter the name of the dictionary table (or dimension table) corresponding to the field . For example, the name of the dictionary table corresponding to the member status is dim_user_status. If you use a globally unique dictionary table, enter the corresponding key_type of the field in the dictionary

table. For example, the corresponding key value of the member status is TAOBAO_USER_STATUS.

- Value range: The maximum and minimum values of the current field. It is applicable only to Bigint and Double fields.
- Regular expression verification: The regular expression used by the current field. For example, if a field is a mobile phone number, the value can be limited to an 11-digit number through regular expression (or more strict limitations).
- Maximum length: The maximum number of characters of the field value. It is applicable only to string fields.
- Date precision: The precision of the date, which can be set to Hour, Day, Month, or others. For example, the precision of month_id in the monthly summary table is Month, although, the field value is 2014-08-01 (it seems that the precision is Day). It is applicable to date values of the datetime or string type.
- Date format: The format is applicable only to the date values of the string type. The format of the date value stored in the field is similar to yyyy-mm-dd hh:mm:ss.
- KV primary separator/secondary separator: It is applicable to a large field (of the string type) combined with KV pairs. For example, if a product expansion attribute has a value similar to "key1:value1;key2:value2;key3:value3;...", the semicolon (;) is the primary separator of the field that separates the KV pairs, and the colon (:) is the secondary separator that separates the key and value in a KV pair.

· Partition field design: This option is displayed only when the Partition Type in the Physical Model Design area is set to Partitioned Table.
· Field type: We recommend that you use the string type for all fields.
· Date partition format: If a partition field is a date (although its data type may be string), and select or enter a date format, such as yyyymmmdd.
· Date partition granularity: For example, Day, Month, or Hour.

Submit a table

After editing the table structure information, submit the new table to the development environment and production environment.

- Click Load from Development Environment. If the table has been submitted to the development environment, this button is highlighted. After you click the button, the information of the created table in the development environment overwrites the information on the current page.
- Click Submit to Development Environment, the system checks whether all required items on the current editing page are completely set. If any omission exists, an alarm is reported to prevent you from submitting the table.
- Click Load from Production Environment, to submit the detailed information of the table to the production environment. Information on the current page will be overwritten.
- Click Create in Production Environment, to create the table in the project of the production environment.

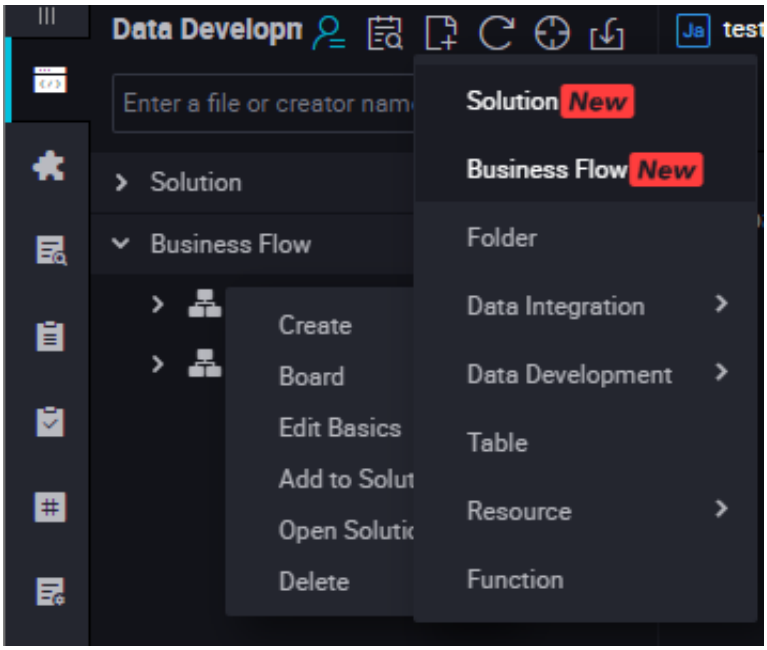
1.9 Manual task node type

1.9.1 ODPS SQL node

The ODPS SQL adopts a syntax similar to that of SQL, and is applicable to a distributed scenario, where the amount of data is massive (TB-level) with low real-time requirement. It is an OLAP application oriented to throughput. ODPS SQL is recommended if a business needs to handle thousands or tens of thousands of transactions because it takes a long time to complete the process from preparation to submission of a job.

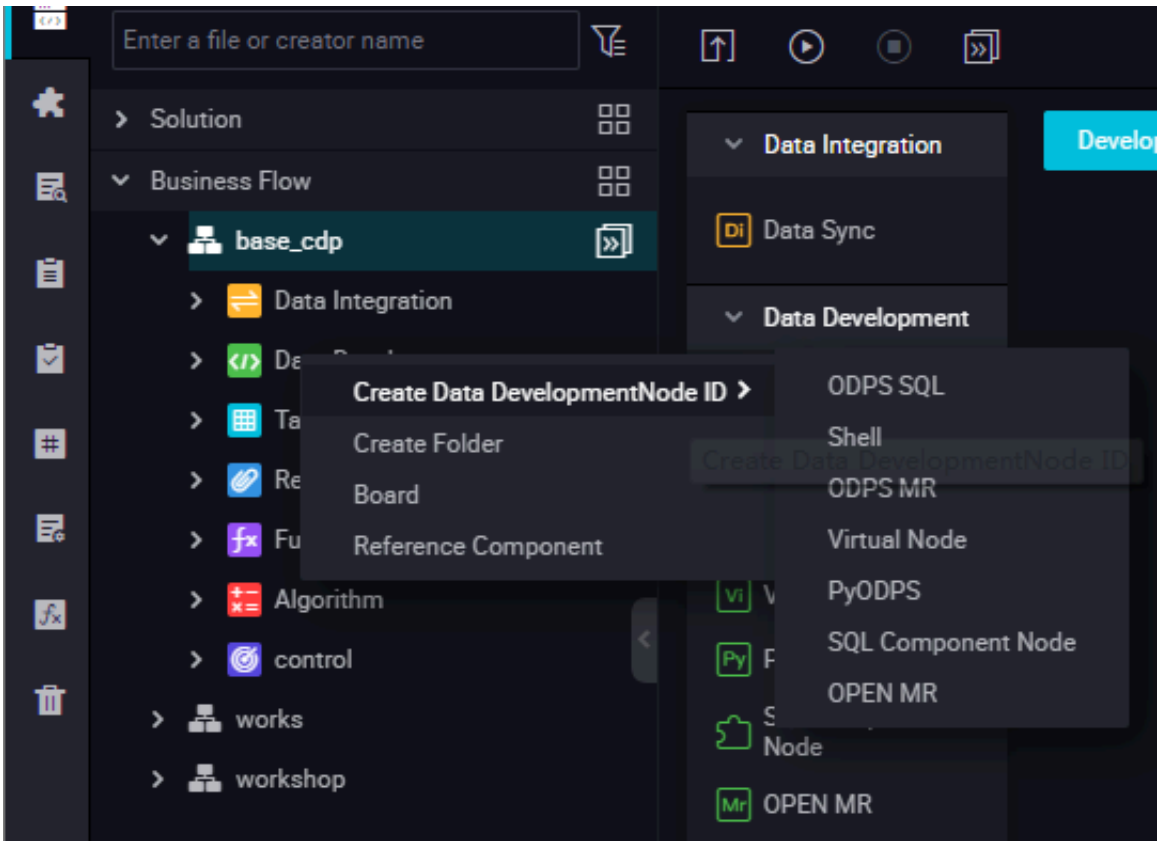
1. Create a business flow.

   Click Manual Business Flow in the left-side navigation pane, and select Create Business Flow.



2. Create ODPS SQL node.

   Right-click Data Development, and select Create Data Development Node > ODPS SQL.



3. Edit the node code.

   For more information about the syntax of the SQL statements, see MaxCompute SQL statements.

4. Node scheduling configuration.

   Click the Schedule on the right of the node task editing area to go to the Node Scheduling Configuration page. For more information, see Scheduling configuration.

5. Submit the node.

   After the configuration is completed, click Save in the upper-left corner of the page or press Ctrl+S to submit (and unlock) the node to the development environment.

6. Publish a node task.

   For more information about the operation, see Release management.

7. Test in the production environment.

   For more information about the operation, see #unique_102.

1.9.2 PyODPS node

This topic describes the PyODPS node functions. DataWorks provides the PyODPS task type and integrates the Python SDK of MaxCompute. You can edit the Python code to operate MaxCompute on a PyODPS node of DataWorks.

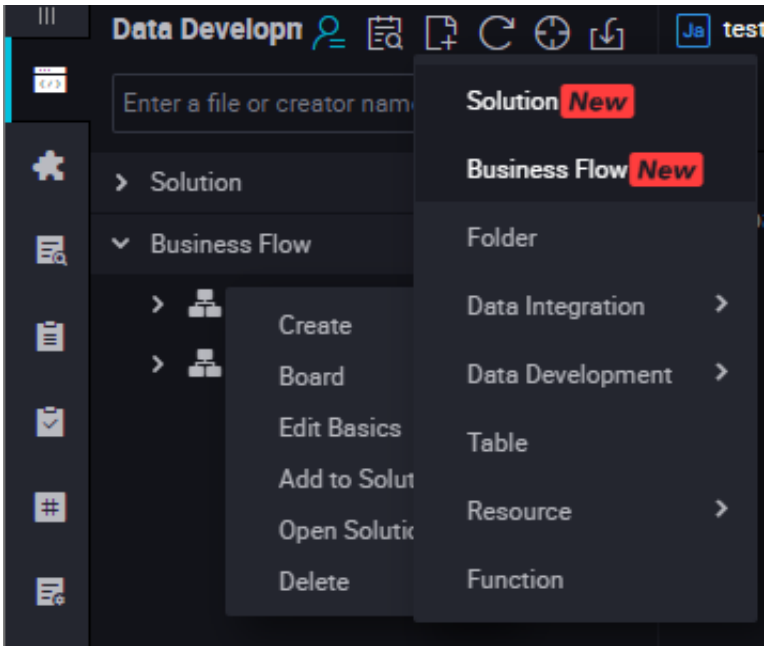
Create a PyODPS Node

MaxCompute provides the Python SDK, which can be used to operate MaxCompute.

To create a PyODPS node, perform the following steps:

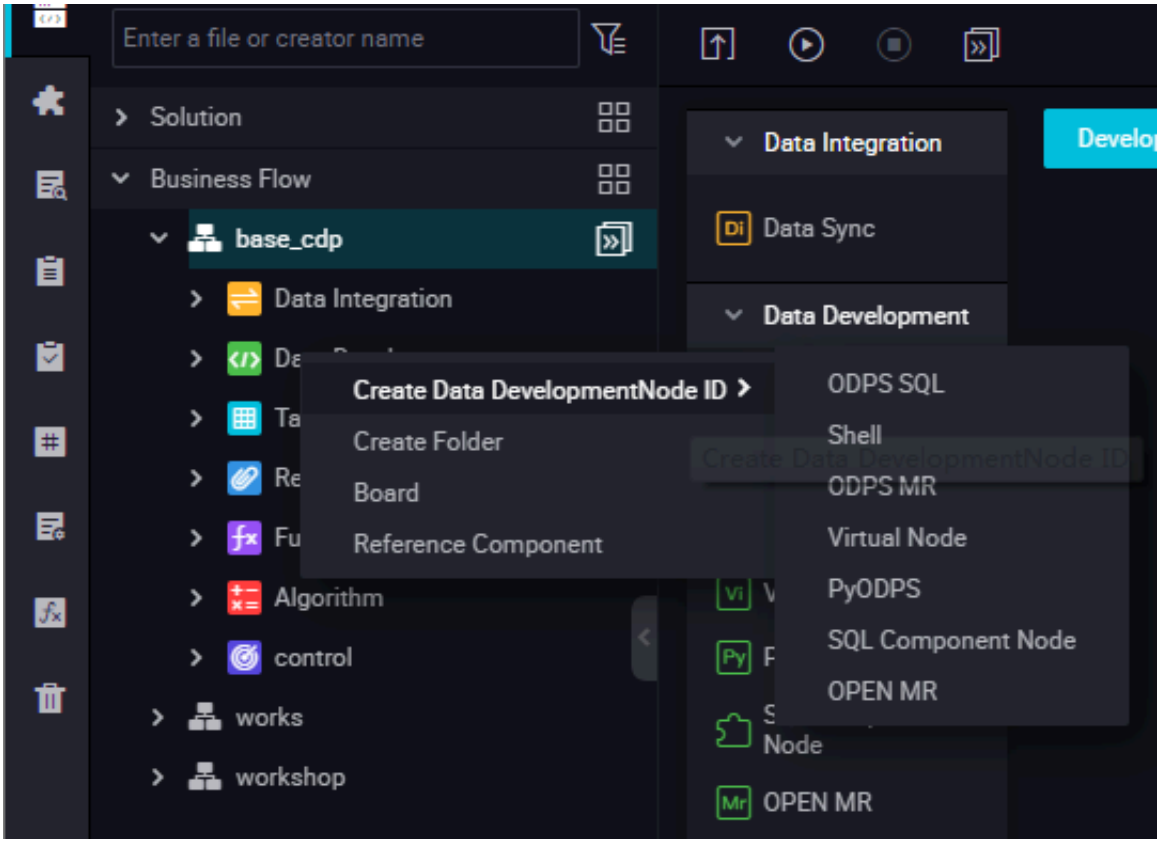
1. Create a business flow

   Click Manual Business Flow in the left-side navigation pane, and select Create Business Flow.



2. Create a PyODPS node.

   Right-click Data Development, and select Create Data Development Node > PyODPS.



3. Edit the PyODPS node.

   a. ODPS portal

      On DataWorks, the PyODPS node contains a global variable odps or o, which is the ODPS entry. You do not need to manually define an ODPS entry.

```
print ( odps . exist_tabl  e (' PyODPS_iri  s '))
```

   b. Run the SQL statements

      PyODPS supports ODPS SQL query and can read the execution result. The return value of the execute_sql or run_sql method is the running instance.

Note:

Not all commands that can be executed on the ODPS console are SQL statements that are accepted by ODPS. You need to use other methods to call non DDL/DML statements. For example, use the run_security_query method to call the GRANT or REVOKE statements, and use the run_xflow or execute_xflow method to call PAI commands.

```
o . execute_sq  l (' select  *  from   dual ') #  Run   the
  SQL   statements   in   synchronou  s   mode .  Blocking
continues   until   execution   of   the   SQL   statement   is
  completed .
instance  =  o . runsql (' select  *  from   dual ') #  Run
the   SQL   statements   in   asynchrono  us   mode .
print ( instance . getlogview  _address ()) #  Obtain   the
logview   address .
instance . waitforsuc  cess () #  Blocking   continues   until
execution   of   the   SQL   statement   is   completed .
```

   c. Configure the runtime parameters

      The runtime parameters must be set sometimes. You can set the hints parameter with the parameter type of dict.

```
o . execute_sq  l (' select  *  from   PyODPS_iri  s ',  hints
={' odps . sql . mapper . split . size ':  16 })
```

      After you add sql.settings to the global configuration, related runtime parameters are added upon each running.python.

```
from   odps   import   options
options . sql . settings  = {' odps . sql . mapper . split . size
':  16 }
```

```
o . execute_sq  l (' select  *  from   PyODPS_iri  s ') # " hints
"  is   added   based   on   the   global   configurat  ion .
```

d. Read the SQL statement execution results

The instance that runs the SQL statement can directly perform the open_reader operation. In one case, the structured data is returned as the SQL statement execution result.

```
with   odps . execute_sq  l (' select  *  from   dual ').
open_reade  r ()  as   reader :
for   record   in   reader : #  Process   each   record .
```

In another case, desc may be executed in an SQL statement. In this case, the original SQL statement execution result is obtained through the reader.raw attribute.

```
with   odps . execute_sq  l (' desc   dual '). open_reade  r ()
as   reader :
print ( reader . raw )
```

Note:
User-defined scheduling parameters are used in data development. If a PyODPS node is directly triggered on the page, the time must be clearly specified. The time of a PyODPS node cannot be directly replaced like that of an SQL node.

4. Node scheduling configuration.

Click the Schedule on the right of the node task editing area to go to the node scheduling configuration page. For more information, see Scheduling configuration.

5. Submit the node.

After the configuration is completed, click Save in the upper left corner of the page or press Ctrl+S to submit (and unlock) the node to the development environment.

6. Publish a node task.

For more information about the operation, see Release management.

7. Test in the production environment.

For more information about the operation, see #unique_102.

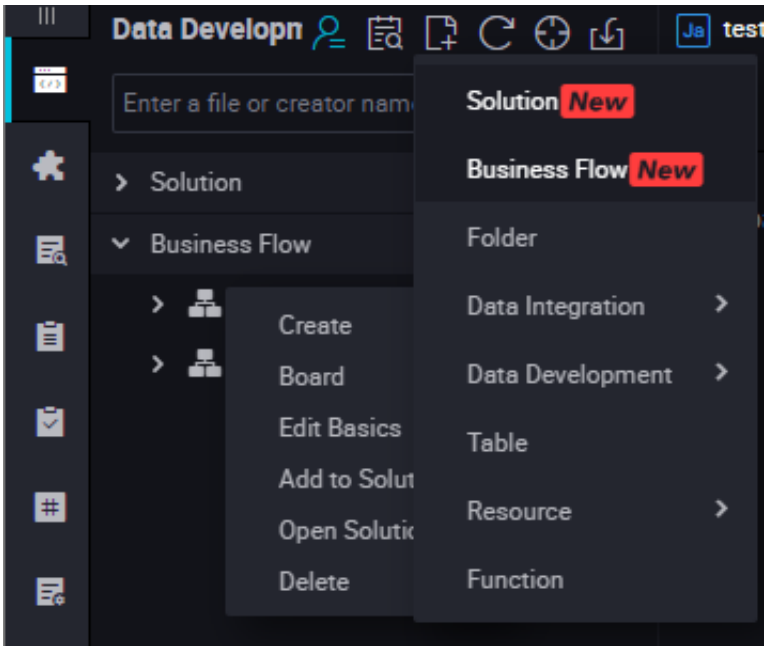
1.9.3 Manual data intergration node

Currently, the data intergration task supports the following data sources: MaxCompute, MySQL, DRDS, SQL Server, PostgreSQL, Oracle, MongoDB, DB2, Table

Store, OTSStream, OSS, FTP, Hbase, LogHub, HDFS, and Stream. For details about more supported data sources, see #unique_29.

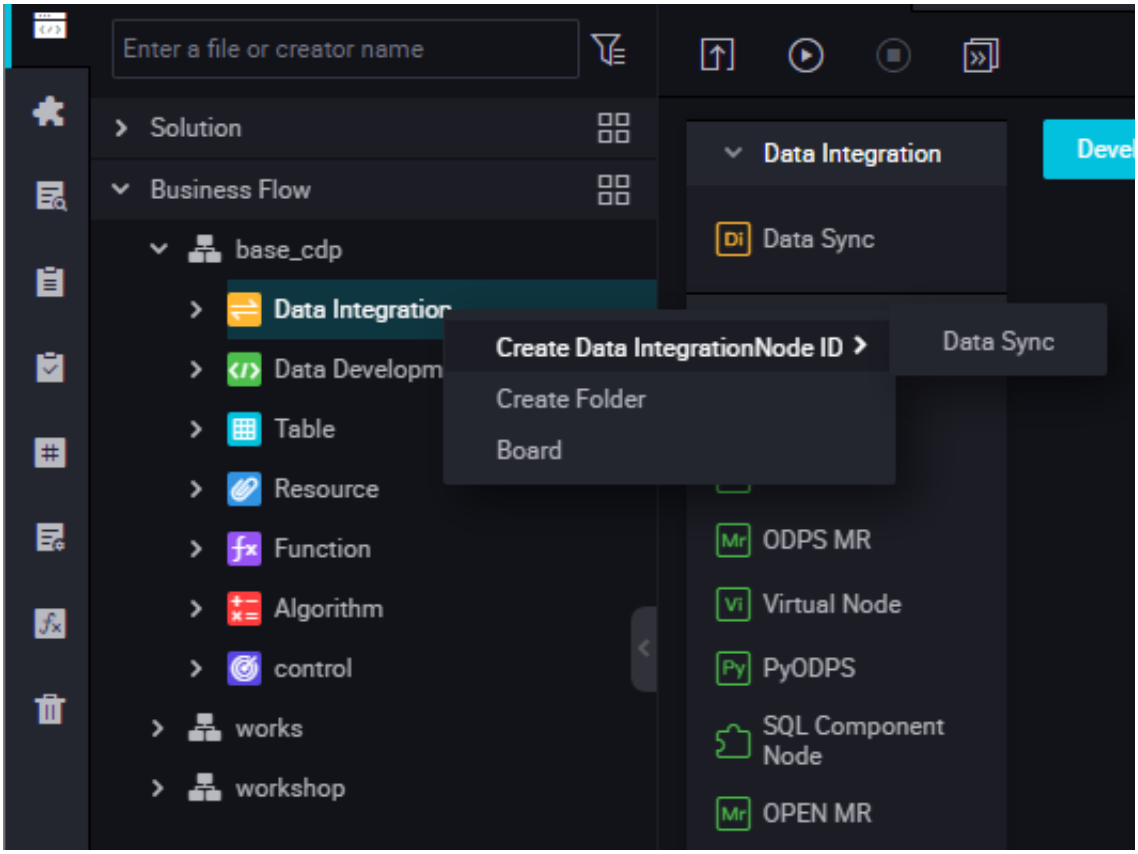
1. Create a business flow

   Click Manual Business Flow in the left-side navigation pane, select Create Business Flow.



2. Create a data intergration node

   Right-click Data Integration, and select Create Data Data Integration Node > Data Integration.



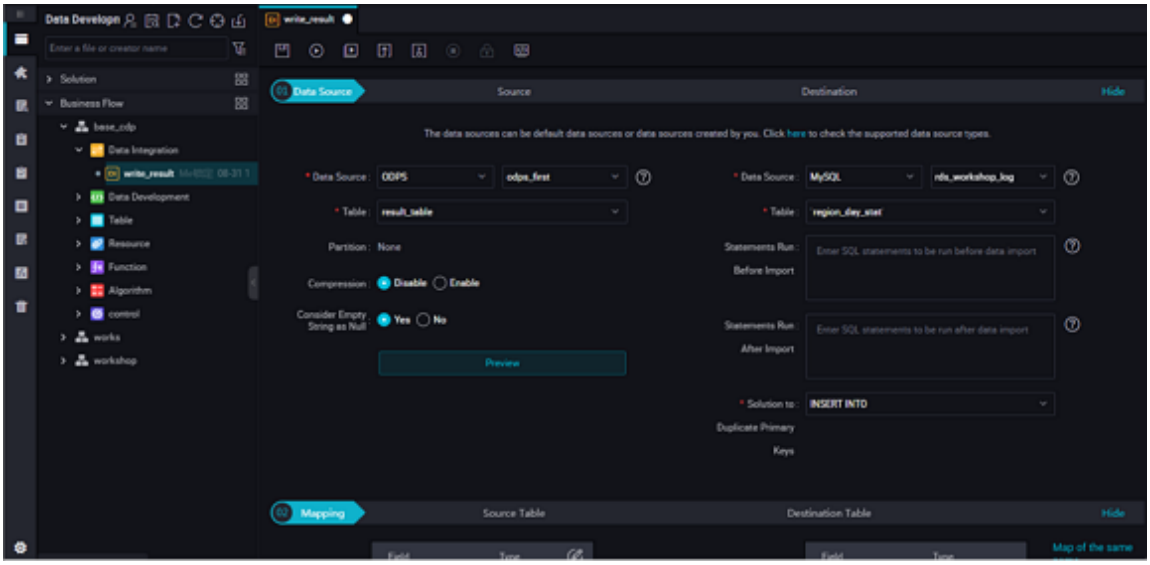
3. Configure a intergration task

   You can enter the source table name and target table name to complete a simple task configuration.

   After you enter a table name, a list of objects that match the table name is automatically displayed(Currently, only exact match is supported. Therefore, you must enter the correct and complete table name), Some objects are not supported by the current intergration center and are marked Not supported. You can move the mouse over an object. The detailed information about the object, such as the database, IP address, and owner of the table, is automatically displayed. The information helps you select an appropriate table object. After selecting an object,

click the object. The column information is automatically filled in. You can edit columns, for example, moving, deleting, or adding column.

a. Configure intergration tables.



b. Edit the data source.

Generally, you do not need to edit the content of the source table unless necessary.

- Click Insert on the right of a column to insert a new column.
- Click Delete on the right of a column to delete the column.

c. Edit the data destination.

Generally, you do not need to edit the field information of the destination table unless necessary (for example, you need to import data of only some columns).

> Note:
> If the destination is an ODPS table, columns cannot be deleted. In configuration of a intergration task, the field settings of the source table matches those of the destination table in one-to-one relationship by page instead of by field name.

d. Incremental intergration and full intergration.

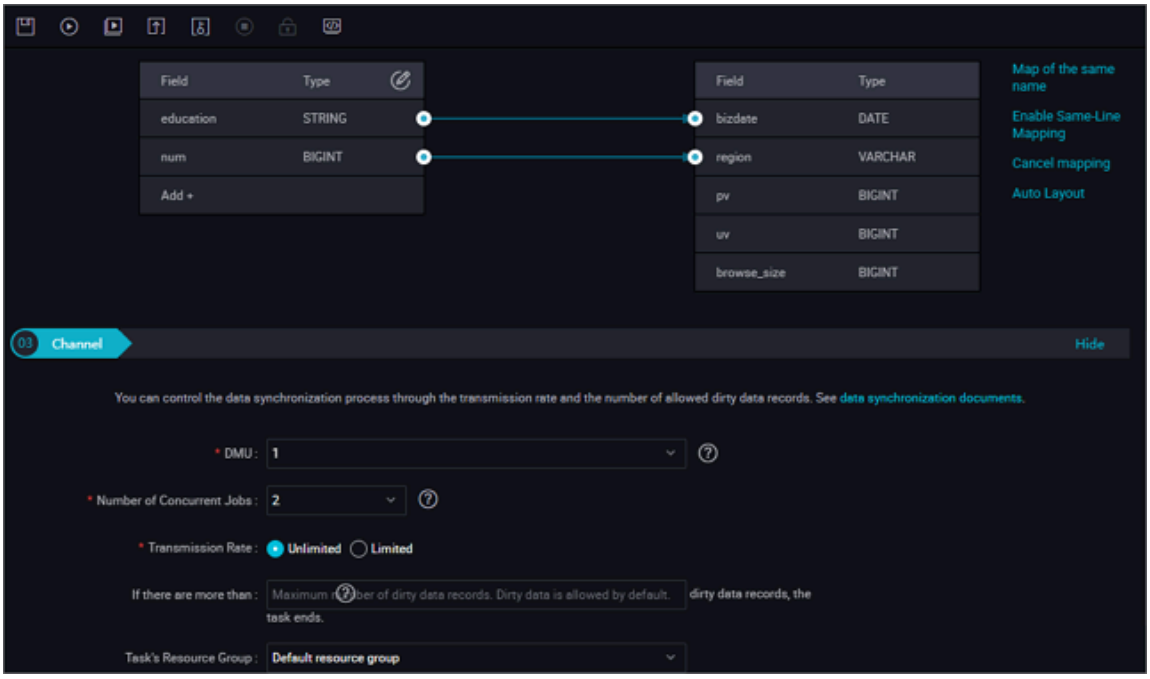
- Shard format for incremental intergration: ds=${bizdate}
- Shard format for full intergration: ds=*

> Note:
> If multiple shards need to be synchronized, the intergration center supports simple regular expressions.

- For example, if you need to synchronize multiple shards, but it is difficult to write regular expressions, use the following method: `ds = 20180312 | ds = 20180313 | ds = 20180314 ;`
- If you need to synchronize shards in the same range, the intergration center supports an extended syntax similar to the following: `/* query */ ds >= 20180313 and ds < 20180315 ;` If this method is used, you must add /query/.
- The variable bizdate must be defined in the following parameter: `- p "- Dbizdate =$ bizdate - Denv_path =$ env_path - Dhour =$ hour "`. If you need to customize a variable, for example, `pt =${ selfVar }`, also define the variable in the parameter, for example, `- p "- Dbizdate =$ bizdate - Denv_path =$ env_path - Dhour =$ hour - DselfVar = xxxx "`.

e. Field mapping.

Fields are mapped based on the locations of fields in the source table and destination table, instead of based on the field names and types.



Note:

If the source table is an ODPS table, fields cannot be added during data intergration. If the source table is not an ODPS table, fields can be added during data intergration.

f. Tunnel control.

Tunnel control is used to control the speed and error rate when you select a intergration task.

- DMU: Data migration unit, which measures the resources (including the CPU, memory, and network) consumed during data integration.
- Concurrent job count: Maximum number of threads used to concurrently read data from or write data into the data storage media in a data intergration task.
- intergration speed: Maximum speed of the intergration task.
- Maximum error count: It is used to control the amount of dirty data, and is set by yourself based on the amount of synchronized data when the field types of the source table do not match those of the destination table. It indicates the maximum dirty data count allowed. If it is set to 0, no dirty data is allowed; if it is not specified, dirty data is allowed.
- Task resource group: To select a resource group where the current intergrati on node is located, you can add or modify the resource group on the data integration page.

4. Node scheduling configuration.

Click the Schedule on the right of the node task editing area to go to the node scheduling configuration page. For more information, see Scheduling configuration.

5. Submit a node task.

After the configuration is completed, click Save in the upper left corner of the page or press Ctrl+S to submit (and unlock) the node to the development environment.

6. Publish a node task.

For more information about the operation, see Release management.

7. Test in the production environment.

For more information about the operation, see #unique_32.

1.9.4 ODPS MR node

MaxCompute supports MapReduce programming APIs. You can use the Java API provided by MapReduce to write MapReduce programs for processing data in MaxCompute. You can create ODPS MR nodes and use them in Task Scheduling.

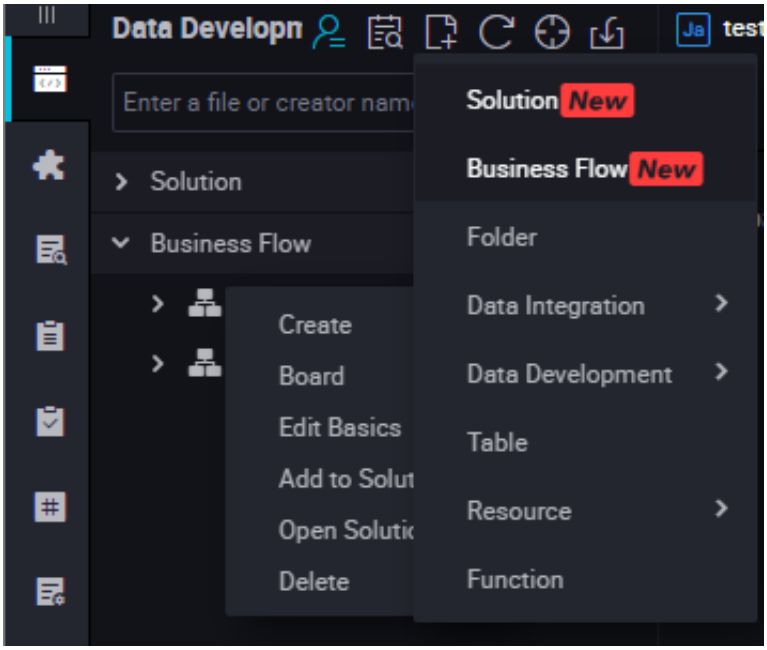
For how to edit and use the ODPS MR, see the examples in the MaxCompute documentation WordCount examples.

To use an ODPS MR node, you must first upload and release the resource to be used, and then create the ODPS MR node.

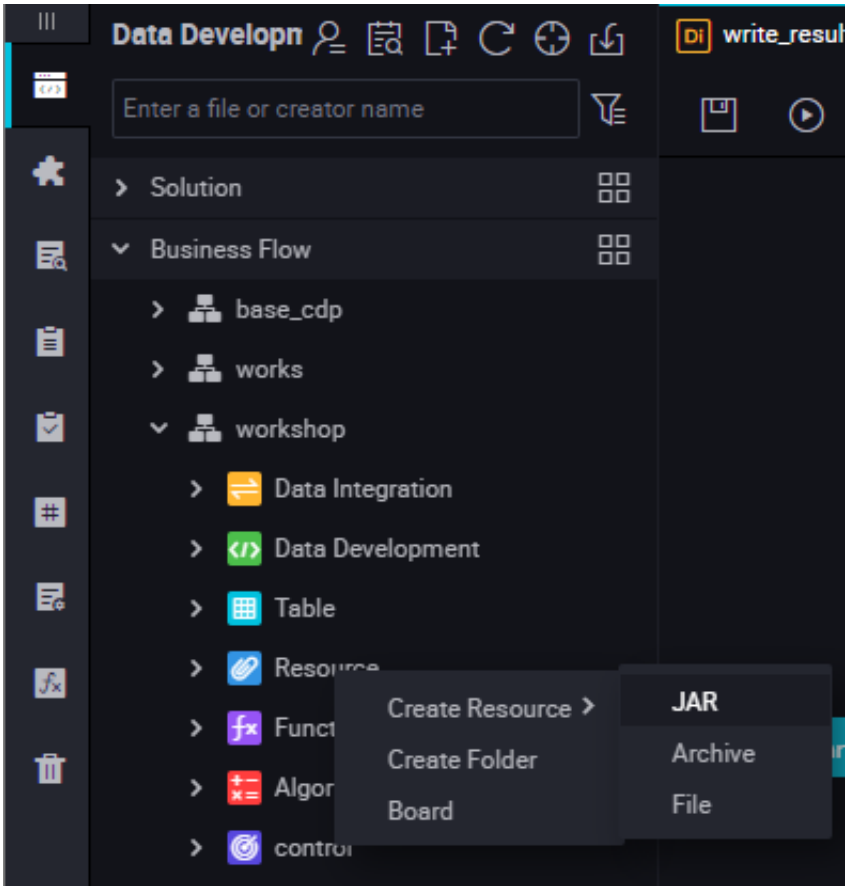
Create a resource instance

1. Create a business flow

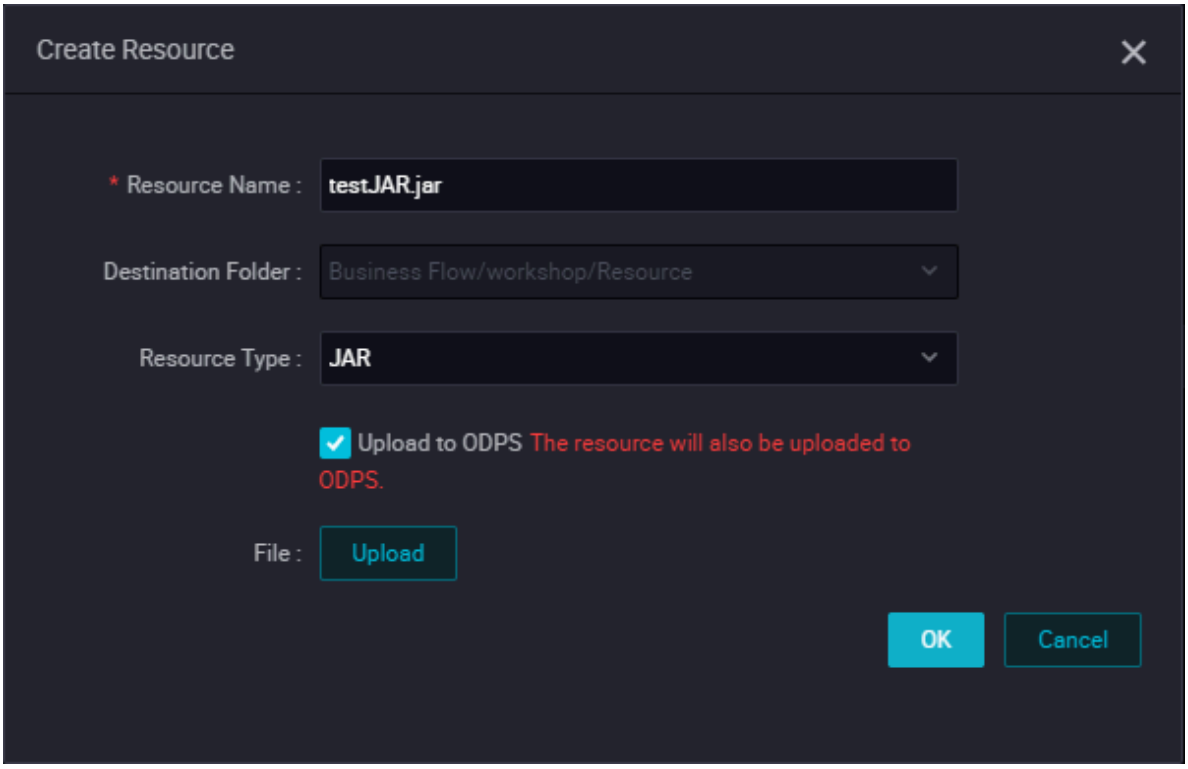
   Click Manual Business Flow in the left-side navigation pane, select Create Business Flow.



2. Right-click Resource, and select Create Resource > jar.



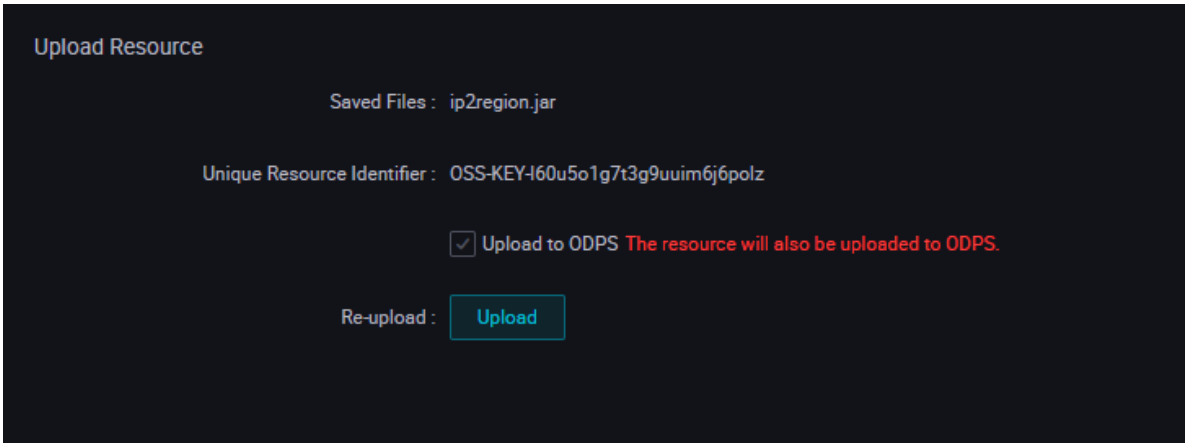
3. Enter the resource name in the Create Resource according to the naming convention, set the resource type to jar, select a local jar package to the uploaded.



Note:

- If this jar package has been uploaded on the ODPS client, you must deselect Uploaded as the ODPS resource. In this upload, the resource will also be uploaded to ODPS. Otherwise, an error will be reported during the upload process.
- The resource name is not necessarily the same as the name of the uploaded file.
- Naming convention for a resource name: a string of 1 to 128 characters, including letters, numbers, underlines, and dots. The name is case insensitive. If the resource is a jar resource, the extension is .jar. If the resource is a Python resource, the extension is .py.

4. Click Submit to submit the resource to the development scheduling server.



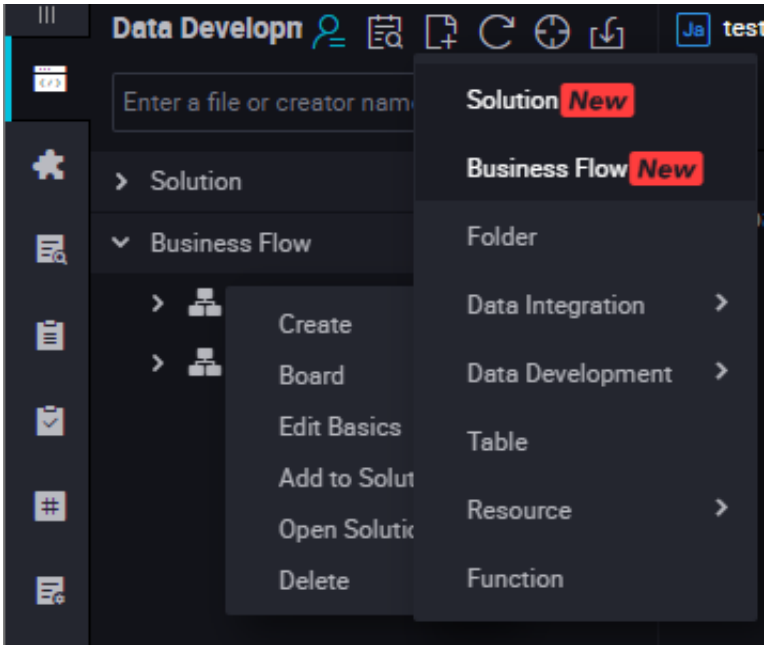
5. Publish a node task.

   For more information about the operation, see Release management.

Create an ODPS MR node

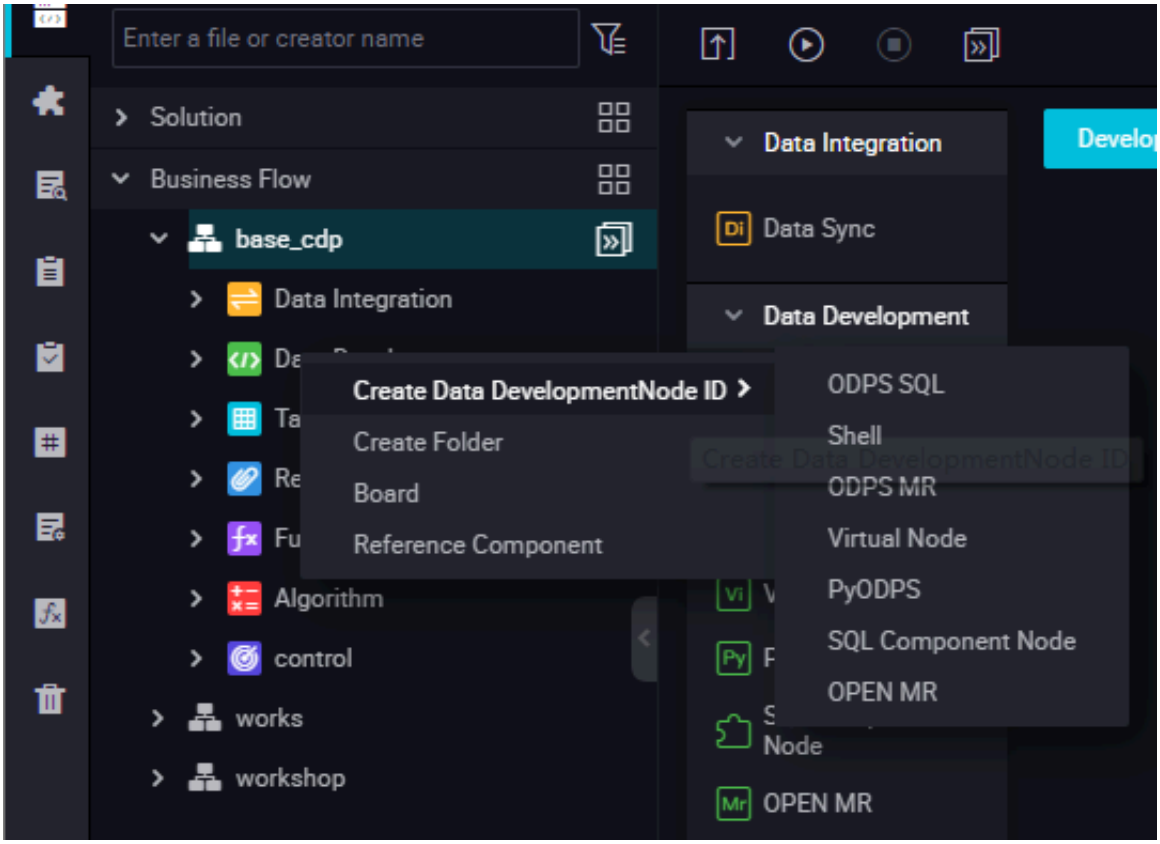
1. Create a business flow

   Click Manual Business Flow in the left-side navigation pane, select Create Business Flow.

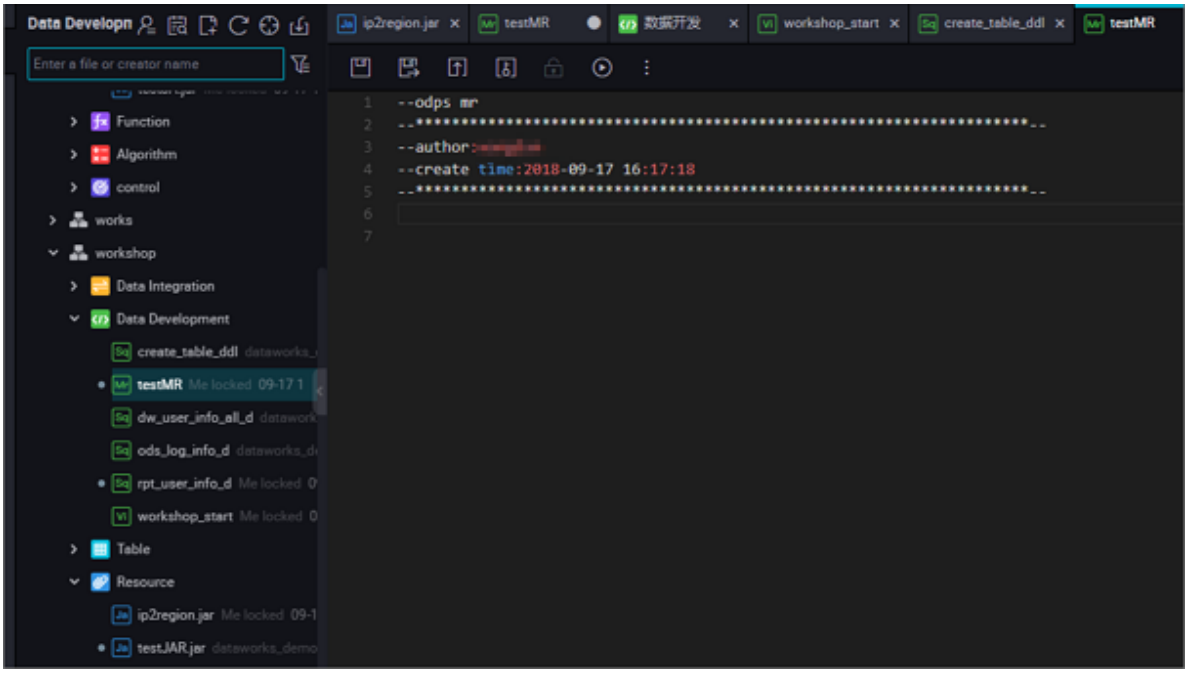


2. Create an ODPS MR node.

   Right-click Data Development, and select Create Data Development Node > ODPS MR.



3. Edit the node code.Double click the new ODPS MR node and enter the following interface.



Node code editing example:

```
jar  -resources  base_test.jar  -classpath  ./base_test.jar
  com.taobao.edp.odps.brandnormalize.WordNormalizeW
ordAll
```

The code is described below:

- `-resources base_test.jar` : indicates the file name of the referenced jar resource.
- `-classpath` : jar package path, you can right-click the Reference resource and obtain this path.

  > **Note:**
  > Double click the new ODPS MR node and enter the jar resource after entering the ODPS MR node interface.

- `com.taobao.edp.odps.brandnormalize.WordNormalizeW ordAll` : indicates the main class in the jar package that is called during execution. It must be consistent with the main class name in the jar package.

When one MR calls multiple jar resources, classpath must be written as follows: `-classpath ./xxxx1.jar,./xxxx2.jar` , that is, two paths must be separated by a comma.

4. Node scheduling configuration.

   Click the Schedule on the right of the node task editing area to go to the node scheduling configuration page. For more information, see Scheduling configuration.

5. Submit the node.

   After the configuration is completed, click Save in the upper left corner of the page or press Ctrl+S to submit (and unlock) the node to the development environment.

6. Publish a node task.

   For more information about the operation, see Release management.

7. Test in the production environment.

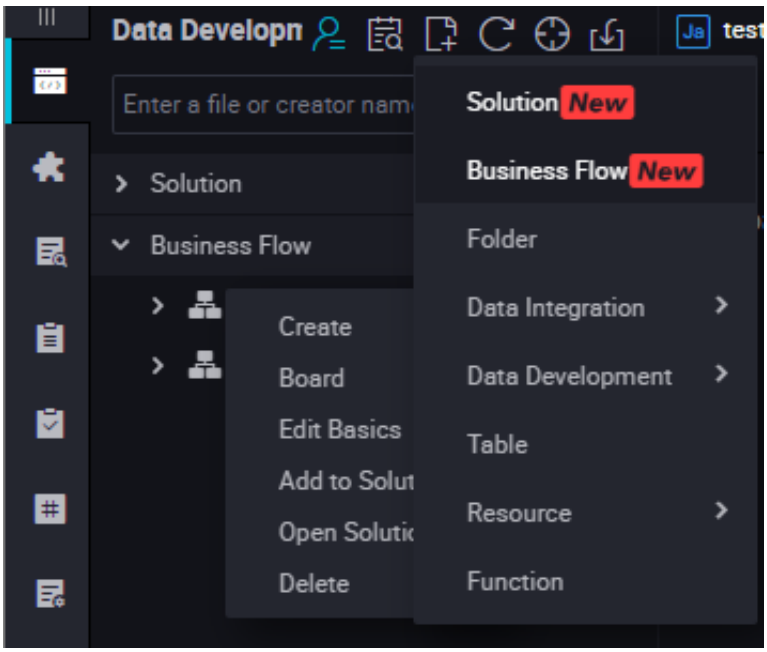
   For more information about the operation, see #unique_102.

1.9.5 SQL component node

Procedure

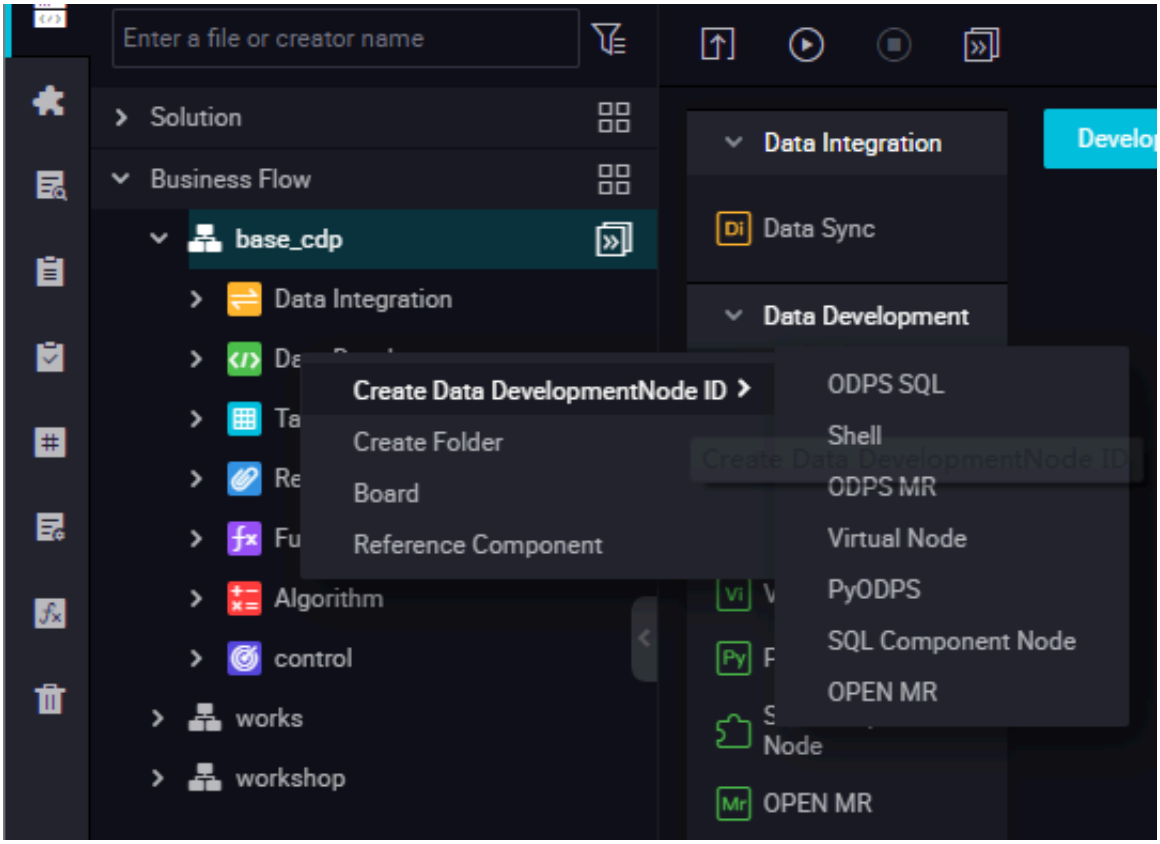
1. Create Business Flow

   Click Manual Business Flow in the left-side navigation pane, select Create Business Flow.



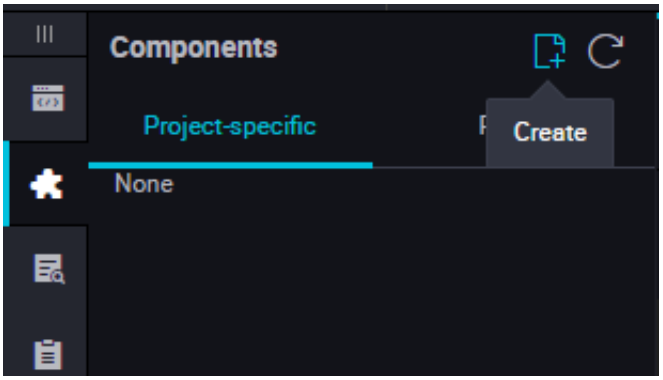
2. Create an SQL component node

   Right-click Data Development, and select Create Data Development Node > SQL Component Node.

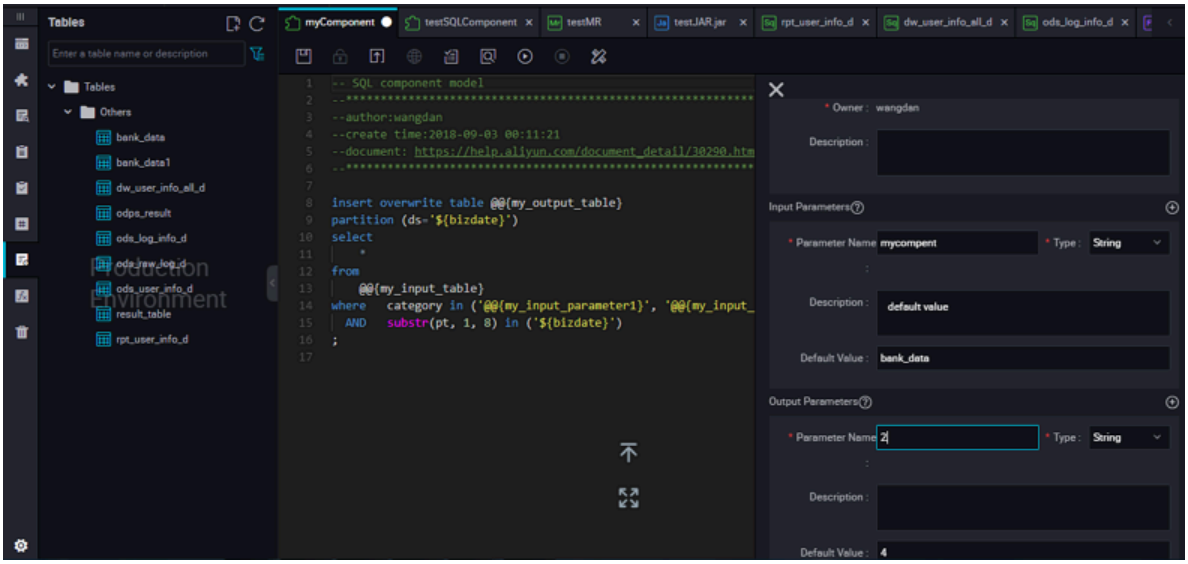


3. To improve the development efficiency, data task developers can use components contributed by project members and tenant members to create data processing nodes.
   - Components created by members of the local project are located under Project Components.
   - Components created by tenant members are located under Public Components.

   When create a node, set the node type to the SQL component node type, and specify the name of the node.



   Specify parameters for the selected component.



   Enter the parameter name, and set the parameter type to Table or String.

   Specify three get_top_n parameters in sequence.

   Specify the following input table for the parameters of the Table type: test_project.test_table.

4. Node scheduling configuration.

   Click the Schedule Configuration on the right of the node task editing area to go to the node scheduling configuration page. For more information, see Scheduling configuration.

5. Submit a node.

   After the configuration is completed, click Save in the upper left corner of the page or press Ctrl+S to submit (and unlock) the node to the development environment.

6. Publish a node task.

   For more information about the operation, see Release management.

7. Test in a production environment.

   For more information about the operation, see #unique_102.

Upgrade the version of an SQL component node.

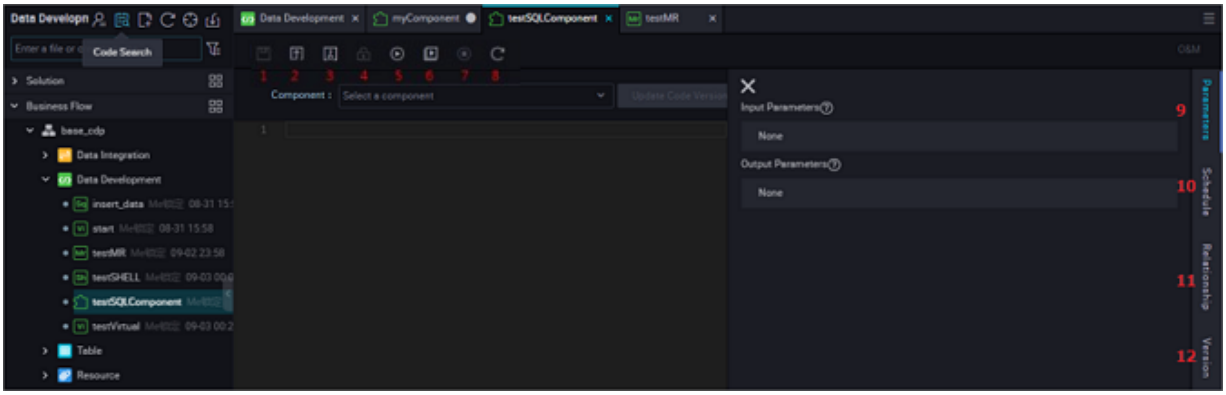
After the component developer release a new version, the component users can choose whether to upgrade the use instance of the existing component to the latest version of the used component.

With the component version mechanism, developers can continuously upgrade components and component users can continuously enjoy the improved process execution efficiency and optimized business effects after upgrade of components.

For example, user A uses the v1.0 component developed by user C, and the component owner C upgrades the component to V.2.0. After the upgrade, user A can still use the v1.0 component, but will receive the upgrade reminder. After comparing the new code with the old code, user A finds that the business effects of the new version are better than those of the old version, and therefore can determine whether to upgrade the component to the latest version.

To upgrade an SQL component node developed based on the component template , you only need to select Upgrade, check whether parameter settings of the SQL component node are still effective in the new version, make some adjustments based on the instructions of the new version component, and then submit and release the node like a common SQL component node.

Interface functions



The interface features are described below:

No.	Feature	Description
1	Save	Click it to save settings of the current component.
2	Submit	Click it to submit the current component to the development environment.
3	Submit and Unlock	Click it to submit the current node and unlock the node to edit the code.
4	Steallock Edit	Click it to steallock edit the node if you are not the owner of the current component.
5	Run	Click it to run the component locally in the development environment.
6	Advanced Run (with Parameters)	Click it to run the code of the current node using the parameters configured for the code. Note: Advanced Run is unavailable to a Shell node.
7	Stop Run	Click it to stop a running component.
8	Re-load	Click it to refresh the interface and restore the last saved status. Unsaved content will be lost. Note: If cache is enabled in the configuration center, after the interface is refreshed, you are notified of the code that is cached but not saved. In this case, select the version that you need.

No.	Feature	Description
9	Parameter Settings	Click it to view the component information, input parameter settings, and output parameter settings.
10	Attributes	Click it set the owner, description, parameters, and resource group of the node.
11	Kinship	Click it to view the map of kinship between SQL component nodes and the internal kinship map of each SQL component node.
12	Version	Click it to view the submission and release records of the current component.

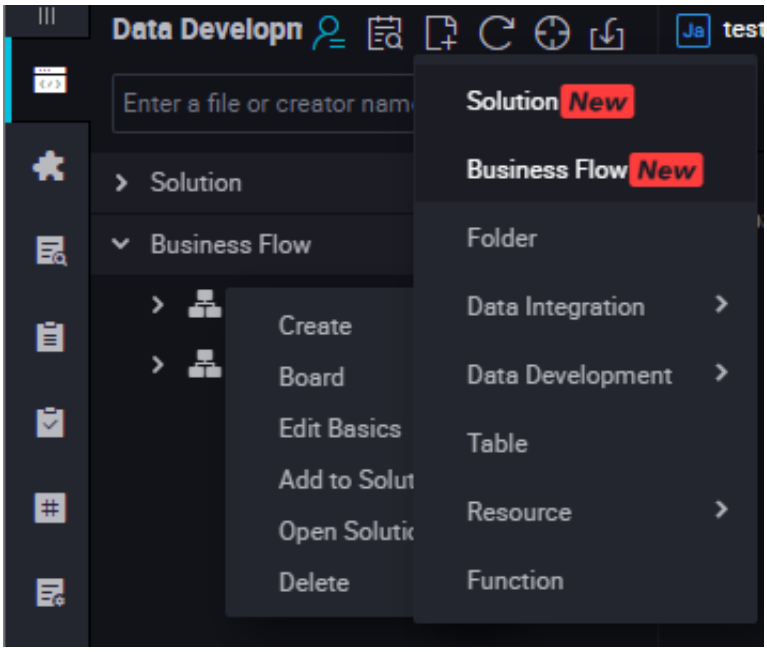
1.9.6 Virtual node

A virtual node is a control node that does not generate any data. Generally, it is used as the root node for overall planning of nodes in the workflow.

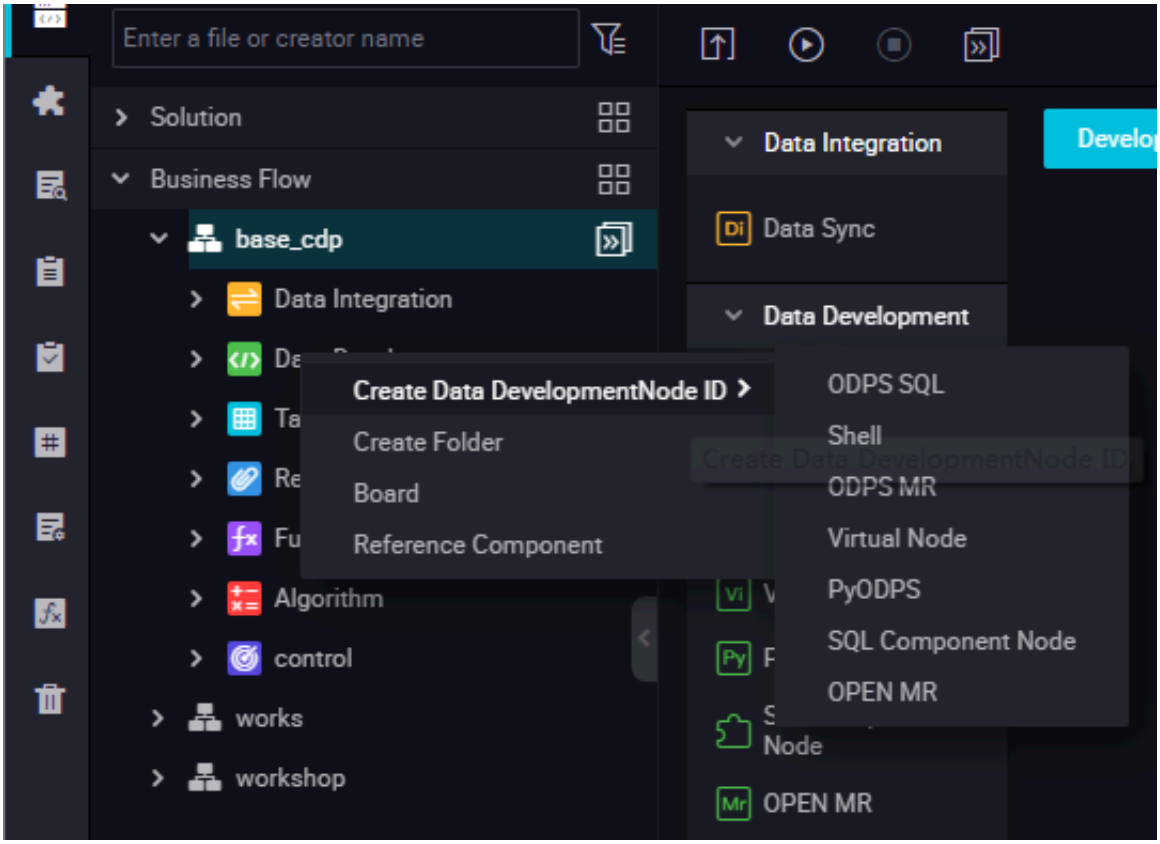
Create a virtual node task

1. Create a business flow

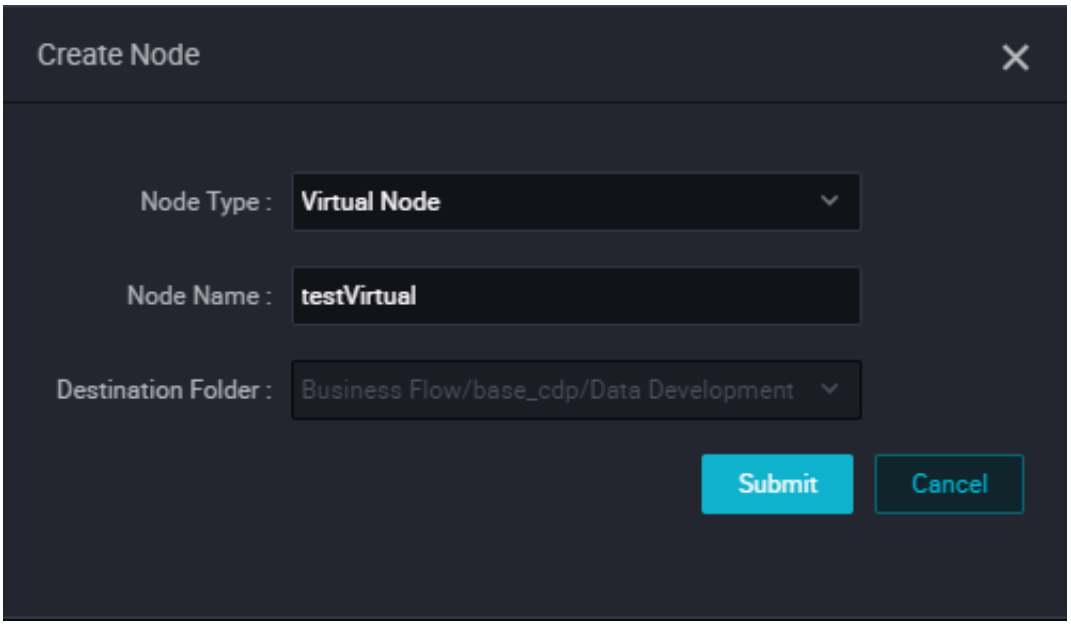
   Click Manual Business Flow in the left-side navigation pane, select Create Business Flow.



2. Create a virtual node. Right-click Data Development, and select Create Data Development Node > Virtual Node.



3. Set the node type to Virtual Node, enter the node name, select the target folder, and click Submit.



4. Edit the node code: You do not need to edit the code of a virtual node.

5. Node scheduling configuration.

   Click the Schedule on the right of the node task editing area to go to the node scheduling configuration page. For more information, see Scheduling configuration.

6. Submit the node.

   After the configuration is completed, click Save in the upper left corner of the page or press Ctrl+S to submit (and unlock) the node to the development environment.

7. Publish a node task.

   For more information about the operation, see Release management.

8. Test in the production environment.

   For more information about the operation, see #unique_102.

1.9.7 SHELL Node

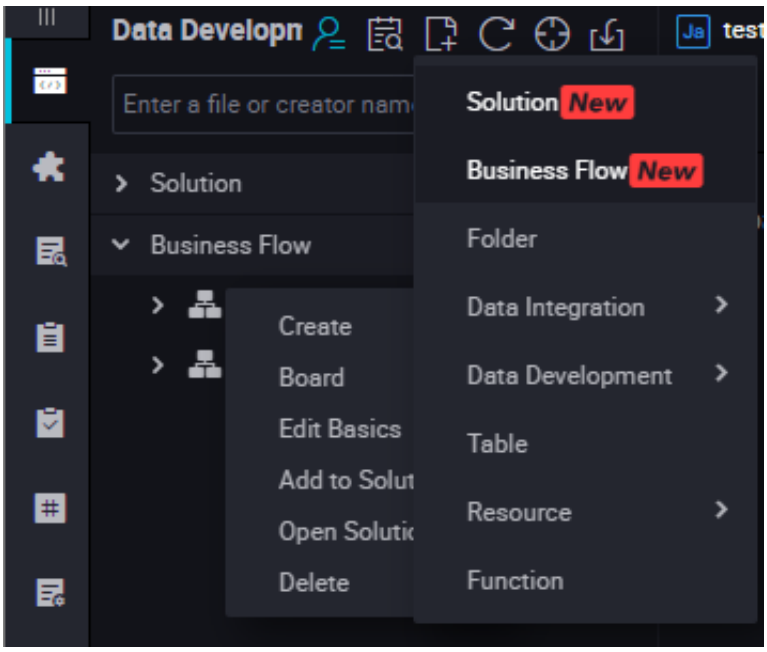
SHELL tasks support standard SHELL syntax but not interactive syntax. SHELL task can run on the default resource group. If you want to access an IP address or a

domain name, add the IP address or domain name to the whitelist by choosing Project Configuration.

Procedure

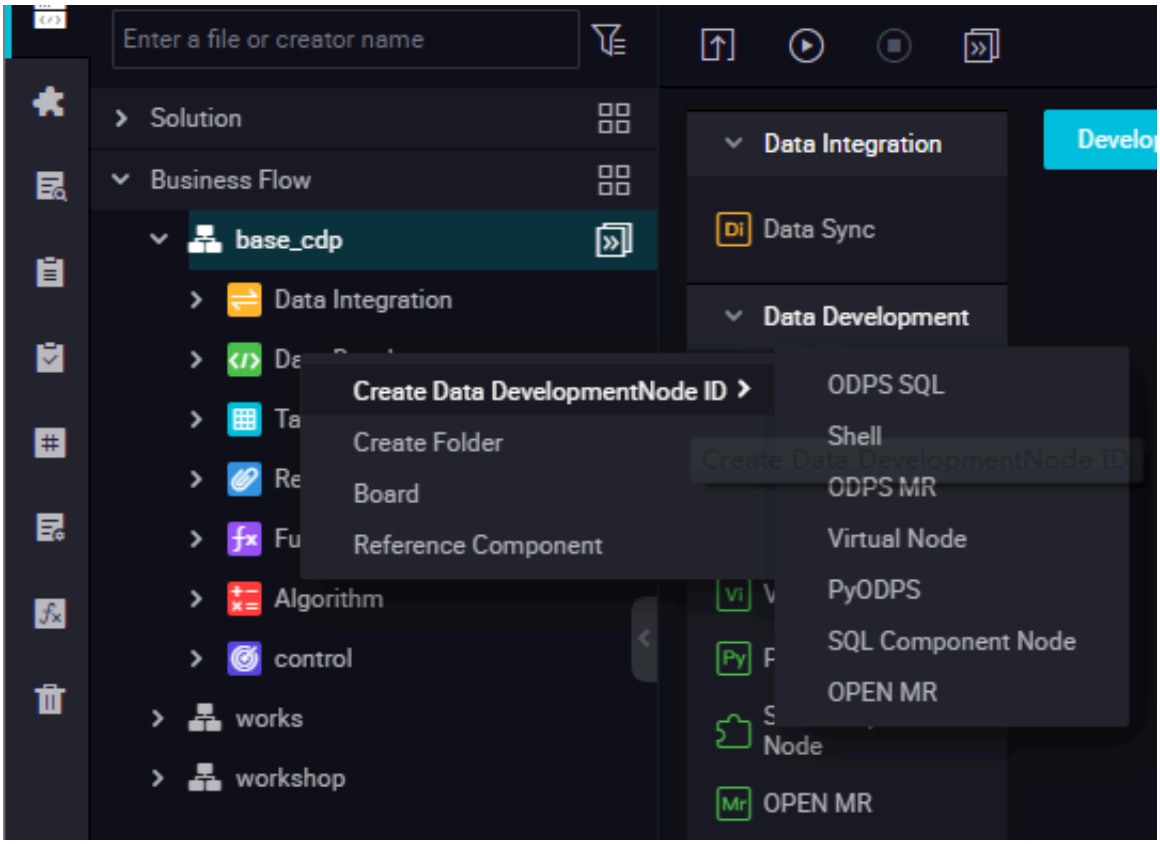
1. Create Business Flow

   Click Manual Business Flow in the left-side navigation pane, select Manual Business Flow.



2. Create a SHELL node.

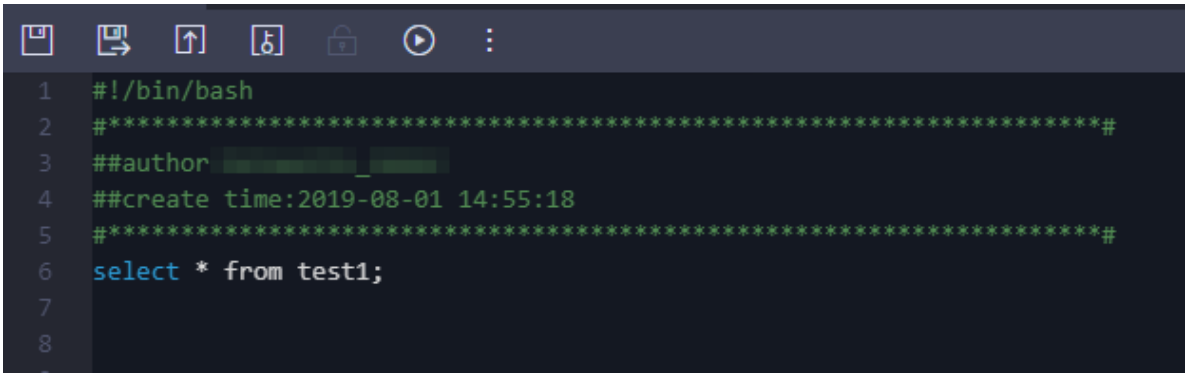
   Right-click Data Development, and select Create Data Development Node > SHELL.



3. Set the node type to SHELL, enter the node name, select the target folder, and click Submit.

4. Edit the node code.

   Go to the SHELL node code editing page and edit the code.



   If you want to call the System Scheduling Parameters in a SHELL statement, compile the SHELL statement as follows:

```
echo "$1 $2 $3"
```

   Note:
   Parameter 1 Parameter 2... Multiple parameters are separated by spaces. For more information on the usage of system scheduling parameters, see #unique_42.

5. Node scheduling configuration.

   Click the Schedule on the right of the node task editing area to go to the node scheduling configuration page. For more information, see Scheduling configuration.

6. Submit the node.

   After the configuration is completed, click Save in the upper left corner of the page or press Ctrl+S to submit (and unlock) the node to the development environment.

7. Release a node task.

   For more information about the operation, see Release management.

8. Test in the production environment.

   For more information about the operation, see #unique_102.

Use cases

Connect to a database using SHELL

- If the database is built on Alibaba Cloud and the region is China (Shanghai), you must open the database to the following whitelisted IP addresses to connect to the database.

  10.152.69.0/24,10.153.136.0/24,10.143.32.0/24,120.27.160.26,10.46.67.156,120.27.160.81,10.46.64.81,121.43.110.160,10.117.39.238,121.43.112.137,10.117.28.203,118.178.84.74,10.27.63.41,118.178.56.228,10.27.63.60,118.178.59.233,10.27.63.38,118.178.142.154,10.27.63.15,100.64.0.0/8

  Note:
  If the database is built on Alibaba Cloud but the region is not China (Shanghai), we recommend that you use the Internet or buy an ECS instance in the same region of the database as the scheduling resource to run the SHELL task on a custom resource group.

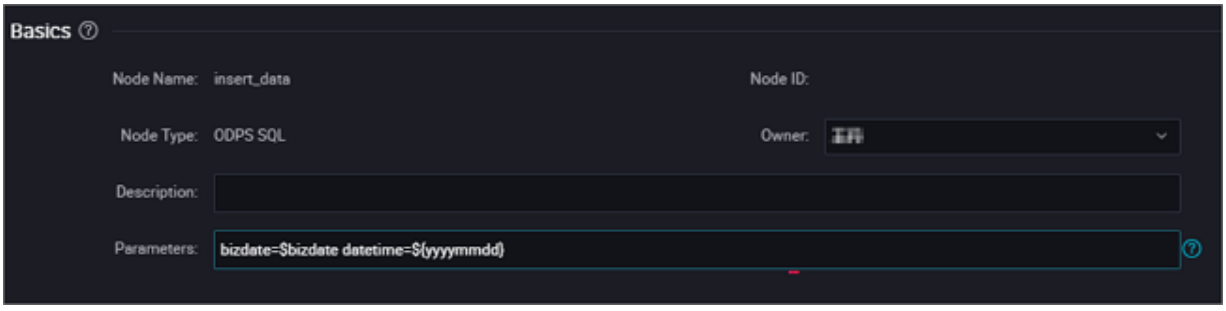
- If the database is built locally, we recommend that you use the Internet connection and open the database to the preceding whitelisted IP addresses.

  Note:
  If you are using a custom resource group to run the SHELL task, you must add the IP addresses of machines in the custom resource group to the preceding whitelist.

1.10 Manual task parameter settings

1.10.1 Basic Attributes

The figure below shows the basic attribute configuration interface:



- Node Name: It is the node name that you enter when creating a workflow node. To modify a node name, right-click the node on the directory tree and choose Rename from the short-cut menu.

- Node ID: It is the unique node ID generated when a task is submitted, and cannot be modified.
- Node ID: It is the unique node ID generated when a task is submitted, and cannot be modified.
- Owner: It is the node owner. The owner of a newly created node is the current logon user by default. To modify the owner, click the input box, and enter the owner name or directly select another user.

  Note:
  When you select another user, the user must be a member of the current project.
- Description: It is generally used to describe the business and purpose of the node.
- Parameter: It is used to assign value to a variable in the code during task scheduling.

  For example, when a variable "pt=${datetime}" is used to indicate the time in the code, you can assign a value to the variable here. The assigned value can use the scheduling built-in time parameter "datetime=$bizdate".
- Resource Group: It specifies the resource group for running the node.

Parameter value assignment formats for various node types

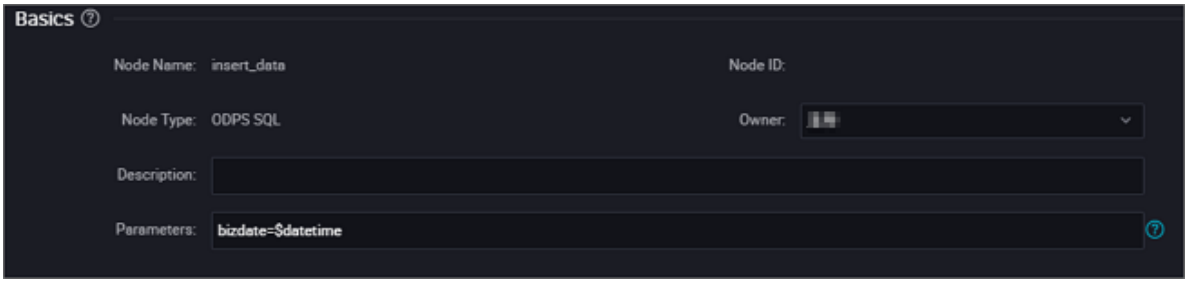
- ODPS SQL, ODPSPL, ODPS MR, and XLIB types: `Variable name 1 = Parameter 1 Variable name 2 = Parameter 2 ...`, Multiple parameters are separated by spaces.
- SHELL type: `Parameter 1 Parameter 2 ...`, Multiple parameters are separated by spaces.

Some frequently-used time parameters are provided as built-in scheduling parameters. For more information about these parameters, see #unique_42.

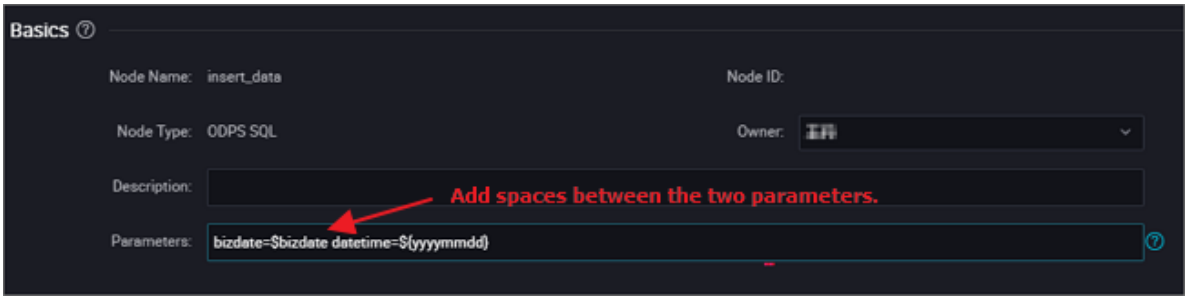
1.10.2 Configure manual node parameters

To ensure that tasks can dynamically adapt to environment changes when running automatically at the scheduled time, DataWorks provides the parameter configuration feature. Pay special attention to the following issues before configuring parameters.

- No space can be added on either side of the equation mark "=" of a parameter.
  Correct: bizdate=$bizdate



- Multiple parameters (if any) must be separated by spaces.



System parameters

DataWorks provides two system parameters, which are defined as follows:

- ${bdp.system.cyctime}: It is defined as the scheduled run time of an instance. Default format: yyyymmddhh24miss.
- ${bdp.system.bizdate}: It is defined as the business date on which an instance is calculated. Default business data is one day before the running date, which is displayed in default format: yyyymmdd.

According to the definitions, the formula for calculating the runtime and business date is as follows: `Runtime = Business date - 1`.

To use the system parameters, directly reference '${bizdate}' in the code without setting system parameters in the editing box, and the system will automatically replace the reference fields of system parameters in the code.

> **Note:**
> The scheduling attribute of a periodic task is configured with a scheduled runtime. Therefore, you can backtrack the business date based on the scheduled runtime of an instance and retrieve the values of system parameters for the instance.

Example

Set an ODPS_SQL task that runs every hour between 00:00 and 23:59 every day. To use system parameters in the code, perform the following statement.

```
insert   overwrite   table   tb1   partition ( ds  =' 20180606 ')
select
c1 , c2 , c3
from  (
select  *  from   tb2
where   ds  ='${ bizdate }');
```

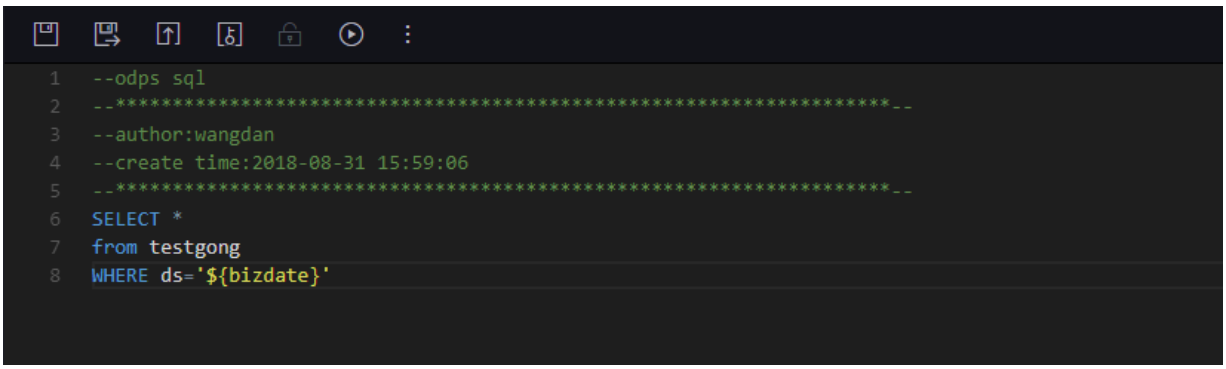
Configure scheduling parameters for a non-Shell node

Note:
The name of a variable in the SQL code can contain only a-z, A-Z, numbers, and underlines. If the variable name is "date", the value "$bizdate" is automatically assigned to this variable, and you do not need to assign the value in the scheduling parameter configuration. Even if another value is assigned, this value is not used in the code because the value "$bizdate" is automatically assigned in the code by default.

For a non-Shell node, you need to first add ${variable name} (indicating that the function is referenced) in the code, then input a specific value to assign the value to the scheduling parameter.

For example, for an ODPS SQL node, add ${variable name} in the code, and then configure the parameter item "variable name=built-in scheduling parameter" for the node.

For a parameter referenced in the code, you must add the parsed value during scheduling.



Configure scheduling parameters for a Shell node

The parameter configuration procedure of a Shell node is similar to that of a non-Shell node except that rules are different. For a Shell node, variable names cannot be customized and must be named '$1,$2,$3...'.

For example, for a Shell node, the Shell syntax declaration in the code is: $1, and the node parameter configuration in scheduling is: $xxx (built-in scheduling parameter). That is, the value of $xxx is used to replace $1 in the code.

For a parameter referenced in the code, you must add the parsed value during scheduling.

Note:
For a Shell node, when the number of parameters reaches 10, ${10} should be used to declare the variable.

The variable value is a fixed value

Take an SQL node for example. For ${variable name} in the code, configure the parameter item "variable name="fixed value"" for the node.

Code: select xxxxxx type=’ ${type}’

Value assigned to the scheduling variable: type="aaa"

During scheduling, the variable in the code is replaced by type='aaa'.

The variable value is a built-in scheduling parameter

Take an SQL node for example. For ${variable name} in the code, configure the parameter item variable name=scheduling parameter for the node.

Code: select xxxxxx dt=${datetime}

alue assigned to the scheduling variable: datetime=$bizdate

During scheduling, if today is July 22, 2017, the variable in the code is replaced by dt=20170721.

Built-in scheduling parameter list

$bizdate: business date in the format of yyyymmdd NOTE: This parameter is widely used, and is the date of the previous day by default during routine scheduling.

For example: In the code of the ODPS SQL node, pt=${datetime}. In the parameter configuration of the node, datetime=$bizdate. Today is July 22, 2017. When the node is executed today, $bizdate is replaced by pt=20170721.

For example, In the code of the ODPS SQL node, pt=${datetime}. In the parameter configuration of the node, datetime=$gmtdate. Today is July 22, 2017. When the node is executed today, $gmtdate is replaced by pt=20170722.

For example, In the code of the ODPS SQL node, pt=${datetime}. In the parameter configuration of the node, datetime=$bizdate. Today is July 1, 2017. When the node is executed today, $bizdate is replaced by pt=20130630.

For example, In the code of the ODPS SQL node, pt=${datetime}. In the parameter configuration of the node, datetime=$gmtdate. Today is July 1, 2017. When the node is executed today, $gmtdate is replaced by pt=20170701.

$cyctime: scheduled time of the task. If no scheduled time is configured for a daily task, cyctime is 00:00 of the current day. The time is accurate to hour, minute, and second, and is generally used for a hour-level or minute-level scheduling task. Example: cyctime=$cyctime.

 Note:
Pay attention to the difference between the time parameters configured using $[] and ${}. $bizdate: business date, which is one day before the current time by default. $cyctime: It is the scheduled time of the task. If no scheduled time is configured for a daily task, the task is executed on 00:00 of the current day. The time is accurate to hour, minute, and second, and is generally used for an hour-level or minute-level scheduling task. If a task is scheduled to run on 00:30, for example, on the current day, the scheduled time is yyyy-mm-dd 00:30:00. If the time parameter is configured using [], cyctime is used as the benchmark for running. For more information about the usage, see the instructions below. The time calculation method is the same with that of Oracle. During data population, the parameter value after replacement will

be the business date + 1 day. For example, if the date of 20140510 is selected as the business date, the cyctime will be replaced by 20140511.

$jobid: ID of the workflow to which a task belongs. Example: jobid=$jobid.

$nodeid: ID of a node. Example: nodeid=$nodeid.

$taskid: ID of a task, that is, ID of a node instance. Example: taskid=$taskid.

$bizmonth: business month in the format of yyyymm.

- If the month of a business date is equal to the current month, $bizmonth = Month of the business date - 1; otherwise, $bizmonth = Month of the business date.
- For example: In the code of the ODPS SQL node, pt=${datetime}. In the parameter configuration of the node, datetime=$bizmonth. Today is July 22, 2017. When the node is executed today, $bizmonth is replaced by pt=201706.

$gmtdate: current date in the format of yyyymmdd. The value of this parameter is the current date by default. During data population, gmtdate that is input is the business date plus 1.

Custom parameter ${…} Parameter description:

- Time format customized based on $bizdate, where yyyy indicates the 4-digit year, yy indicates the 2-digit month, mm indicates the month, and dd indicates the day. The parameter can be combined as expected, for example, ${yyyy}, ${yyyymm}, ${yyyymmdd}, ${yyyy-mm-dd}.
- $bizdate is accurate to year, month, and day. Therefore, the custom parameter ${……} can only represent the year, month, or day.
- Methods for obtaining the period plus or minus certain duration:

  Next N years: ${yyyy+N}

  Previous N years: ${yyyy-N}

  Next N months: ${yyyymm+N}

  Previous N months: ${yyyymm-N}

  Next N weeks: ${yyyymmdd+7*N}

  Previous N weeks: ${yyyymmdd-7*N}

  Next N days: ${yyyymmdd+N}

  Previous N days: ${yyyymmdd-N}

${yyyymmdd}: business date in the format of yyyymmdd. The value is consistent with that of $bizdate.

- Note: The value is consistent with that of $bizdate. This parameter is widely used , and is the date of the previous day by default during routine scheduling. The format of this parameter can be customized, for example, the format of ${yyyy-mm -dd} is yyyy-mm-dd.
- For example: In the code of the ODPS SQL node, pt=${datetime}. In the parameter configuration of the node, datetime=${yyyymmdd}. Today is July 22, 2013. When the node is executed today, ${yyyymmdd} is replaced by pt=20130721.

${yyyymmdd-/+N}: yyyymmdd plus or minus N days

${yyyymm-/+N}: yyyymm plus or minus N month

${yyyy-/+N}: year (yyyy) plus or minus N years

${yy-/+N}: year (yy) plus or minus N years

NOTE: yyyymmdd indicates the business date and supports any separator, such as yyyy-mm-dd. The preceding parameters are derived from the year, month, and day of the business date.

Example:

- In the code of the ODPS SQL node, pt=${datetime}. In the parameter configuration of the node, datetime=${yyyy-mm-dd}. Today is July 22, 2018. When the node is executed today, ${yyyy-mm-dd} is replaced by pt=2018-07-21.
- In the code of the ODPS SQL node, pt=${datetime}. In the parameter configuration of the node, datetime=${yyyymmdd-2}. Today is July 22, 2018. When the node is executed today, ${yyyymmdd-2} is replaced by pt=20180719.
- In the code of the ODPS SQL node, pt=${datetime}. In the parameter configurat ion of the node, datetime=${yyyymm-2}. Today is July 22, 2018. When the node is executed today, ${yyyymm-2} is replaced by pt=201805.
- In the code of the ODPS SQL node, pt=${datetime}. In the parameter configuration of the node, datetime=${yyyy-2}. Today is July 22, 2018. When the node is executed today, ${yyyy-2} is replaced by pt=2018.

In the ODPS SQL node configuration, multiple parameters are assigned values, for example, startdatetime=$bizdate enddatetime=${yyyymmdd+1} starttime=${yyyy-mm -dd} endtime=${yyyy-mm-dd+1}.

Example: (Assume $cyctime=20140515103000)

- $[yyyy] = 2014, $[yy] = 14, $[mm] = 05, $[dd] = 15, $[yyyy-mm-dd] = 2014-05-15, $[hh24:mi:ss] = 10:30:00, $[yyyy-mm-dd hh24:mi:ss] = 2014-05-1510:30:00
- $[hh24:mi:ss - 1/24] = 09:30:00
- $[yyyy-mm-dd hh24:mi:ss -1/24/60] = 2014-05-1510:29:00
- $[yyyy-mm-dd hh24:mi:ss -1/24] = 2014-05-1509:30:00
- $[add_months(yyyymmdd,-1)] = 2014-04-15
- $[add_months(yyyymmdd,-12*1)] = 2013-05-15
- $[hh24] =10
- $[mi] =30

Method for testing the parameter $cyctime:

After the instance runs, right-click the node to check the node attribute. Check whether the scheduled time is the time at which the instance runs periodically.

Result after the parameter value is replaced by the scheduled time minus one hour.

1.11 Component management

1.11.1 Create components

Definition of components

A component is an SQL code process template containing multiple input and output parameters. To handle an SQL code process, one or more source data tables are imported, filtered, joined, and aggregated to form a target table required for new business.

Value of components

In actual businesses, many SQL code processes are similar. The input and output tables in a process have the same or compatible structures but different names. In this case, component developers can abstract such SQL process to an SQL component node, and variable input and output tables in the SQL process to input and output parameters to reuse the SQL code.

When using SQL component nodes, component users only need to select components like their own business flows from the component list, configure specific input and output tables in their own businesses for these components, and generate new SQL component nodes without repeatedly copying the code. This greatly improves the development efficiency and avoids repeated development. Publishing and scheduling of the SQL component nodes after generation is the same as those of common SQL nodes.

Composition of components

Like a function definition, a component consists of the input parameters, output parameters, and component code processes.

Component input parameters

A component input parameter contains the attributes such as the name, type, description, and definition. The parameter type can be table or string.

- A table-type parameter specifies tables to be referenced in a component process. When using a component, the component user can set the parameter to the table required for the specific business.
- A string-type parameter specifies variable control parameters in a component process. For example, if a result table of a specific process only outputs the sales amount of top N cities in each region, the value of N can be specified by the string-type parameter.

  If a result table of a specific process needs to output the total sales amount of a province, a province string-type parameter can be set to specify different provinces and obtain the sales amount of the specified province.
- Parameter description specifies the role of a parameter in a component process.
- Parameter definition is a text definition of the table structure, which is required only for table-type parameters. When this attribute is specified, the component user must provide an input table that is compatible with the field names and types defined by the table parameter so that the component process can run properly . Otherwise, an error is reported when the component process runs because the specified field in the input table cannot be found. The input table must contain the field names and types defined by the table parameter. The fields and types can be in different orders, and the input table can also contain other fields. The

definition is for reference only. It provides guidance for users and does not need to be immediately and forcibly checked.

- The recommended definition format of the table parameter is as follows:

```
Field   1   name   Field   1   type   Field   1   comment
Field   2   name   Field   2   type   Field   2   comment
Field   n   name   Field   n   type   Field   n   comment
```

Example:

```
area_id   string  ‘ Region   ID ’
city_id   string  ‘ City   ID ’
order_amt   double  ‘ Order   amount ’
```

Component output parameters

- A component output parameter contains the attributes such as the name, type, description, and definition. The parameter type can only be table. A string-type output parameter does not have the logical meaning.
- A table-type parameter: specifies tables to be generated from a component process . When using a component, the component user can set the parameter to the result table that the component process generates for the specific business.
- Parameter description: specifies the role of a parameter in a component process.
- Parameter definition: it is a text definition of the table structure. When this attribute is specified, the component user must provide the parameter with an output table that has the same number of fields and compatible type as defined by the table parameter so that the component process can run properly. Otherwise , an error is reported when the component process runs because the number of fields does not match or the type is incompatible. The field names of the output table do not need to be consistent with those defined by the table parameter. The definition is for reference only. It provides guidance for users and does not need to be immediately and forcibly checked.
- The recommended definition format of the table parameter is as follows:

```
Field   1   name   Field   1   type   Field   1   comment
Field   2   name   Field   2   type   Field   2   comment
Field   n   name   Field   n   type   Field   n   comment
```

Example:

```
area_id   string  ‘ Region   ID ’
city_id   string  ‘ City   ID ’
order_amt   double  ‘ Order   amount ’
```

```
rank   bigint  ‘ Rank ’
```

Component process bodies

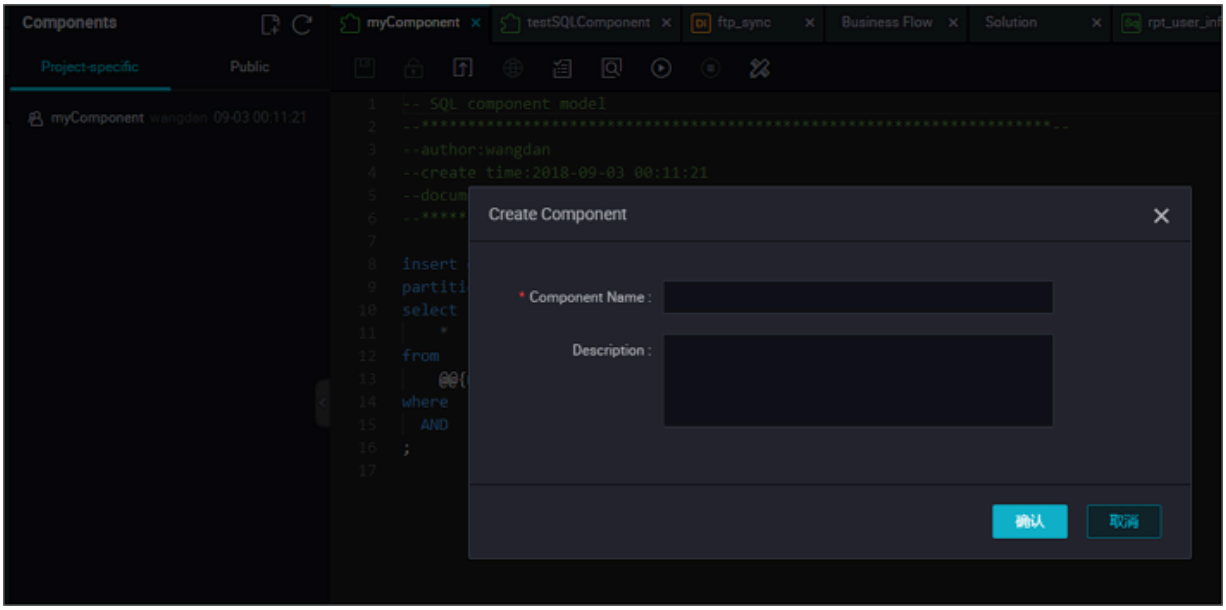
The reference format of the parameters in a process body is as follows: @@{ parameter name}

By compiling an abstract SQL working process, the process body controls the specified input tables based on the input parameters and generates output tables with business value.

Certain skills are required for the development of a component process. Input parameters and output parameters must be well used for the component process code so that different values of input parameters and output parameters can generate correct and runnable SQL code.

Example of creating a component

You can create a component as shown in the following figure.



Source table schema definition

The source MySQL schema definition of the sales data is described in the following table:

Field Name	Field type	Field description
order_id	varchar	Order ID
report_date	datetime	Order date
customer_name	varchar	Customer Name

Field Name	Field type	Field description
order_level	varchar	Order grade
order_number	double	Order quantity
order_amt	double	Order amount
back_point	double	Discount
shipping_type	varchar	Transportation mode
profit_amt	double	Profit amount
price	double	Unit price
shipping_cost	double	Transportation cost
area	varchar	Region
province	varchar	Province
city	varchar	City
product_type	varchar	Product Type
product_sub_type	varchar	Product subtype
product_name	varchar	Product Name
product_box	varchar	Product packing box
shipping_date	Datetime	Transportation date

Business implication of components

Component name: get_top_n

Component description:

In the component process, the specified sales data table is used as the input parameter (table type), the number of the top cities is used as the input parameter (string type), and the cities are ranked by sales amount. In this way, the component user can easily obtain the rank of the specified top N cities in each region.

Definition of component parameters

Input parameter 1:

Parameter name: myinputtable type: table

Input parameter 2:

Parameter name: topn type: string

Input parameter 3:

Parameter name: myoutput type: table

Parameter definition:

area_id string

city_id string

order_amt double

rank bigint

Table creation statement:

```
CREATE TABLE IF NOT EXISTS company_sa les_top_n
(
area STRING COMMENT ' Region ',
city STRING COMMENT ' City ',
sales_amou nt DOUBLE COMMENT ' Sales amount ',
rank BIGINT COMMENT ' Rank '
)
COMMENT ' Company sales ranking '
PARTITIONE D BY ( pt STRING COMMENT '')
LIFECYCLE 365 ;
```

Definition of component process bodies

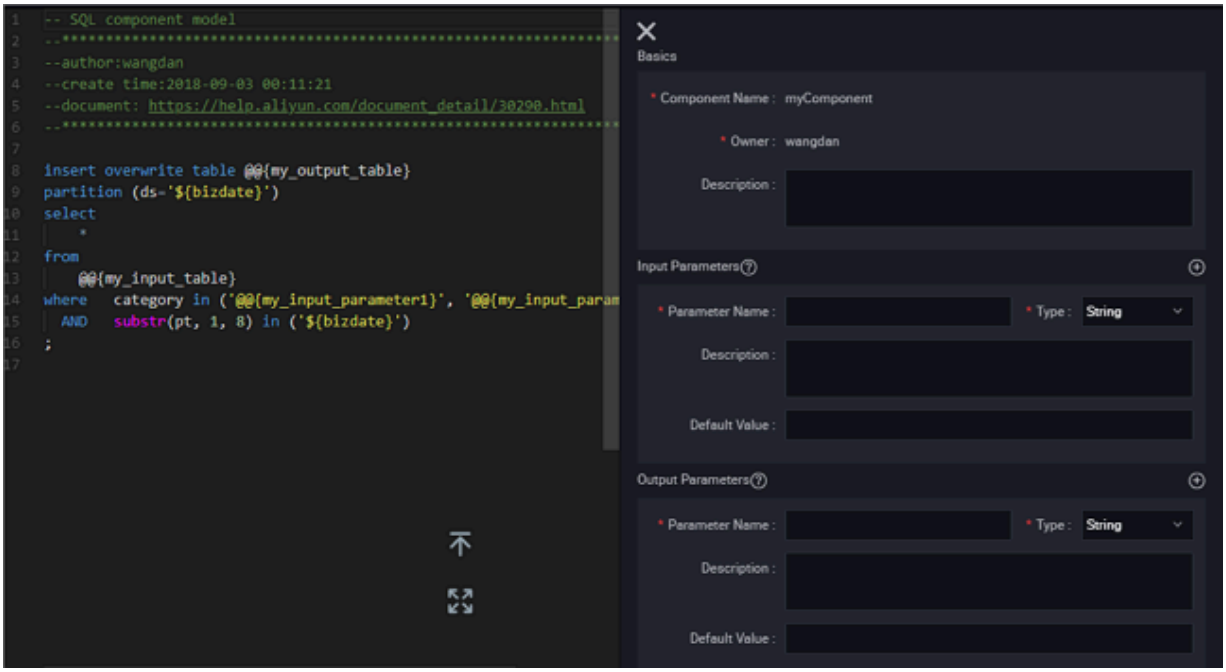
```
INSERT OVERWRITE TABLE @@{ myoutput } PARTITION ( pt ='${
bizdate }')
    SELECT r3 . area_id ,
    r3 . city_id ,
    r3 . order_amt ,
    r3 . rank
from (
SELECT
    area_id ,
    city_id ,
    rank ,
    order_amt_ 1505468133 993_sum as order_amt ,
    order_numb er_1505468 133991_sum ,
    profit_amt _150546813 4000_sum
FROM
   ( SELECT
    area_id ,
    city_id ,
    ROW_NUMBER () OVER ( PARTITION BY r1 . area_id ORDER BY
  r1 . order_amt_ 1505468133 993_sum DESC )
AS rank ,
    order_amt_ 1505468133 993_sum ,
    order_numb er_1505468 133991_sum ,
    profit_amt _150546813 4000_sum
FROM
   ( SELECT area AS area_id ,
     city AS city_id ,
     SUM ( order_amt ) AS order_amt_ 1505468133 993_sum ,
     SUM ( order_numb er ) AS order_numb er_1505468 133991_sum
,
```

```
      SUM ( profit_amt )  AS   profit_amt  _150546813  4000_sum
FROM
   @@{ myinputtab  le }
WHERE
    SUBSTR ( pt ,  1 ,  8 )  IN  ( '${ bizdate }' )
GROUP   BY
    area ,
    city  )
    r1  )  r2
WHERE
    r2 . rank  >=  1   AND   r2 . rank  <= @@{ topn }
ORDER   BY
    area_id ,
    rank   limit   10000 )  r3 ;
```

Sharing scope of components

There are two sharing scopes: project component and public component.

After a component is published, it is visible to users within the project by default. The component developer can click the Publish Component icon to publish a universal global component to the entire tenant, allowing all users in the tenant to view and use the public component. Whether a component is public depends on whether the icon in the following figure is visible:

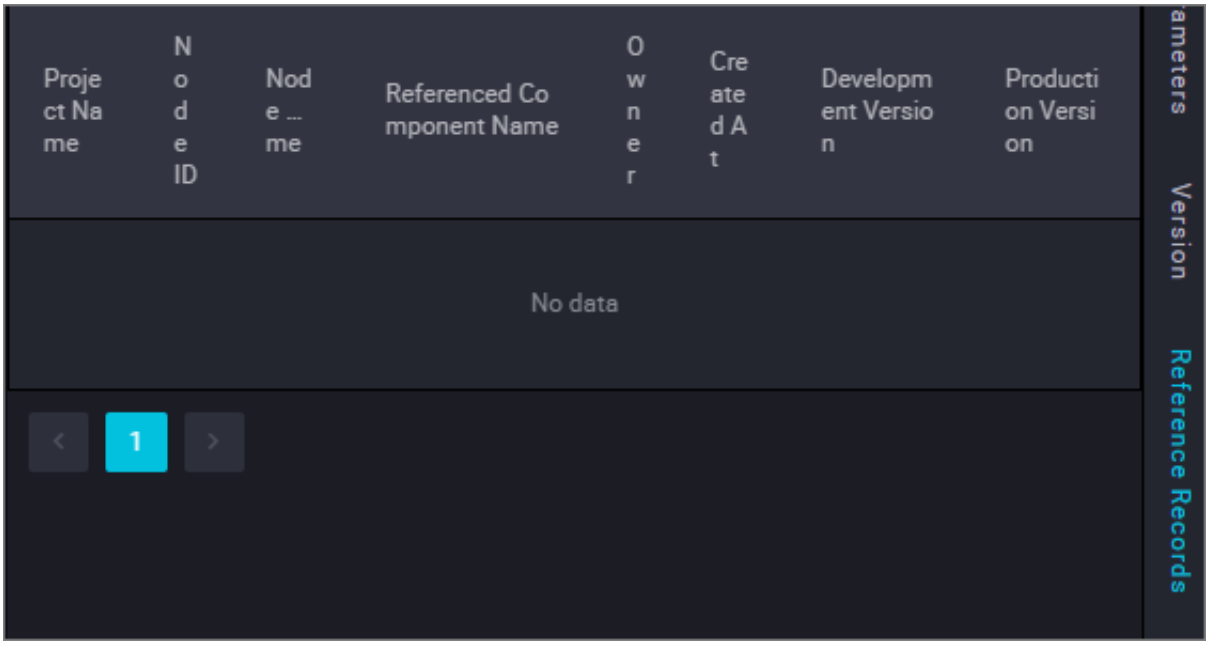


Use of components

How can users use a developed component? For more information, see #unique_114

Reference records of components

The component developer can click the Reference Records tab to view the reference record of a component.



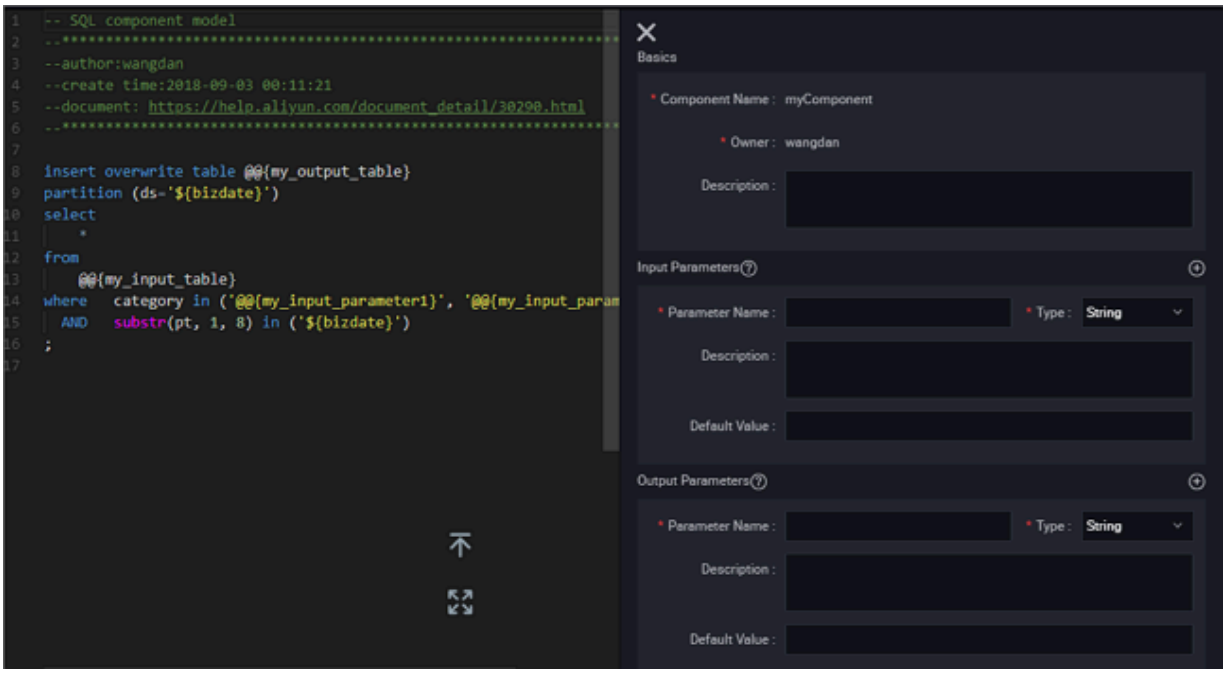
1.11.2 Use components

To improve the development efficiency, data task developers can use components contributed by project and tenant members to create data processing nodes.

- Components created by members of the local project are located under Project Components.
- Components created by tenant members are located under Public Components.

For more information about how to use the components, see #unique_27.

Interface functions



The interface functions are described below:

No.	Function	Description
1	Save	Click it to save settings of the current component.
2	Steallock Edit	Click it to steallock edit the node if you are not the owner of the current component.
3	Submit	Click it to submit the current component to the development environment.
4	Publish Component	Click it to publish a universal global component to the entire tenant, so that all users in the tenant can view and use the public component.
5	Resolve Input and Output Parameters	Click it to resolve the input and output parameters of the current code.
6	Pre-compile	Click it to edit custom and component parameters of the current component.
7	Run	Click it to run the component locally in the development environment.
8	Stop Run	Click it to stop a running component.
9	Format	Click it to sort the current component code by keyword.
10	Parameter settings	Click it to view the component information, input parameter settings, and output parameter settings.
11	Version	Click it to view the submission and release records of the current component.
12	Reference Records	Click it to view the use record of the component.

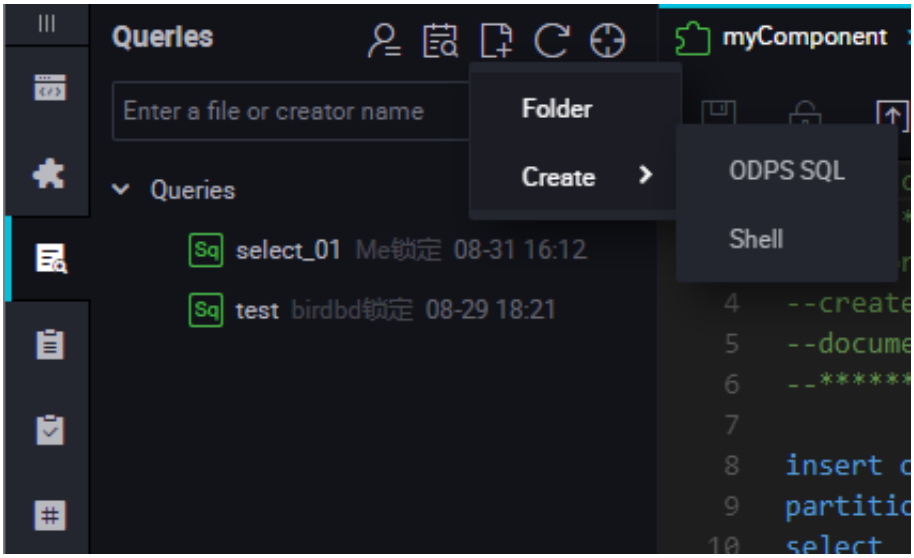
1.12 Queries

Temporary query facilitates you to use the editing code, test whether the actual conditions of the local code meets the expectations, and check the code status. Therefore, temporary query does not support submitting, releasing, and setting the

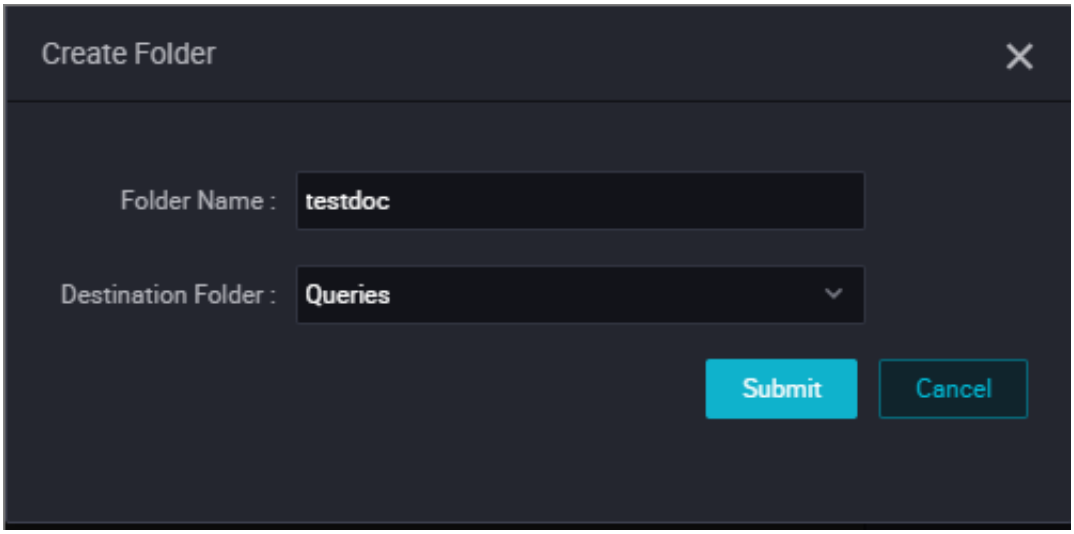
scheduling parameters. To use the scheduling parameters, create a node in Data development or Manual business flow.

Create a folder

1. Click the Queries in the left-hand navigation bar, select folder.



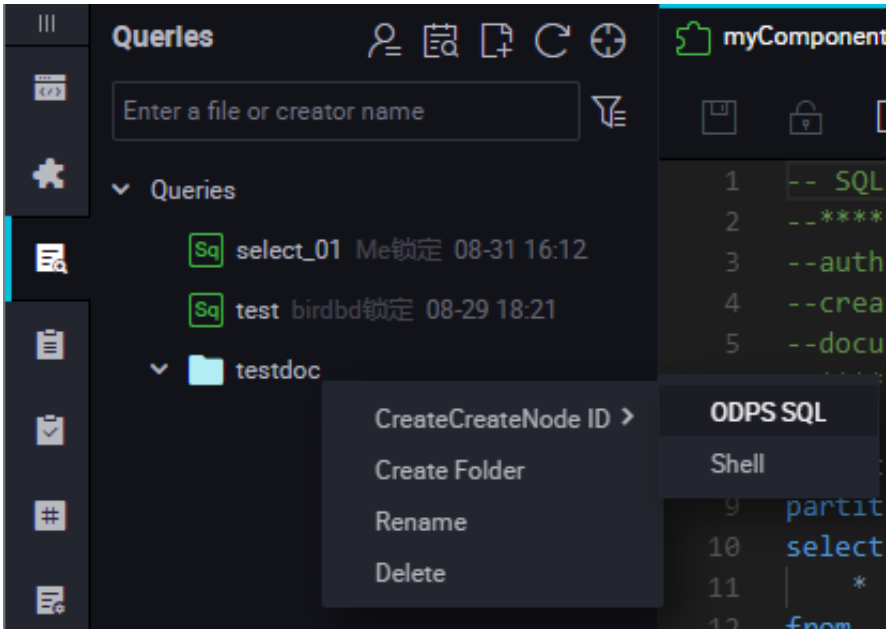
2. Enter the folder name, select the folder directory, and click Submit.



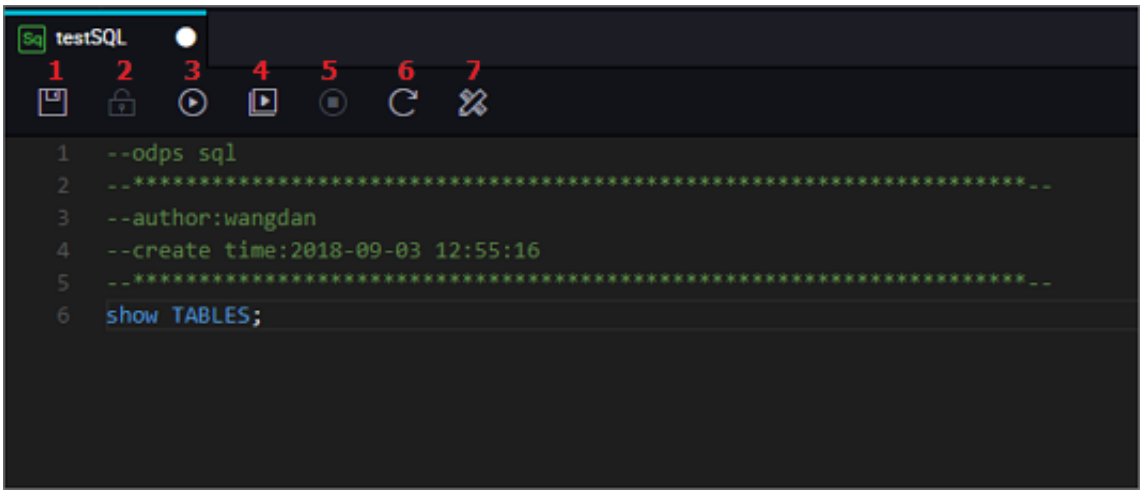
> Note:
> A multi-level folder directory is supported. Therefore, you can store the folder in another folder that has been created.

Create a node

Temporary query only supports the SHELL and SQL nodes.



Take the new ODPS SQL node as an example, right-click the folder name and select Create Node > ODPS SQL.



No.	Function	Description
1	Save	Click it to save the entered code.
2	Steallock Edit	A user other than the node owner can click it to edit the node.
3	Run	Click it to run the code locally (in the development environment).
4	Advanced Run (with Parameters)	Click it to run the code of the current node using the parameters configured for the code. Note: Advanced Run is unavailable to a Shell node.

No.	Function	Description
5	Stop Run	Click it to stop the code that is being run.
6	Reload	Click it to refresh the page, reload, and restore the last saved status. Unsaved content will be lost.  Note: If the cache has been enabled in the configuration center, a message is displayed after page refreshing, indicating that the unsaved code has been cashed. Select a required version.
7	Format	Click it to sort the current node code by keyword format. It is often used when a row of code is too long.

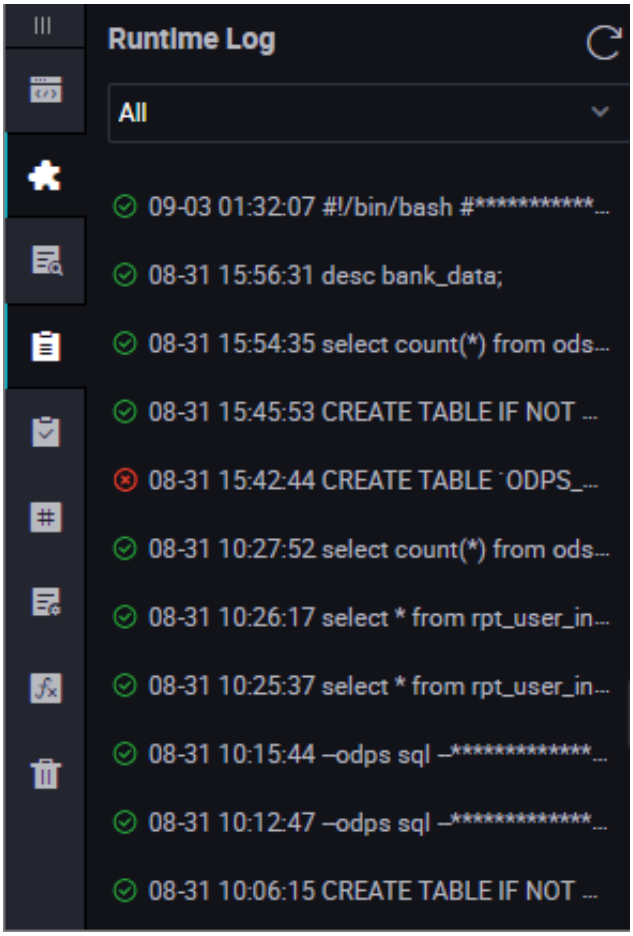
1.13 Running log

The Running Log page displays the record of all tasks that have locally run in the past three days. You can click it to view the task history and filter the running records by task status.

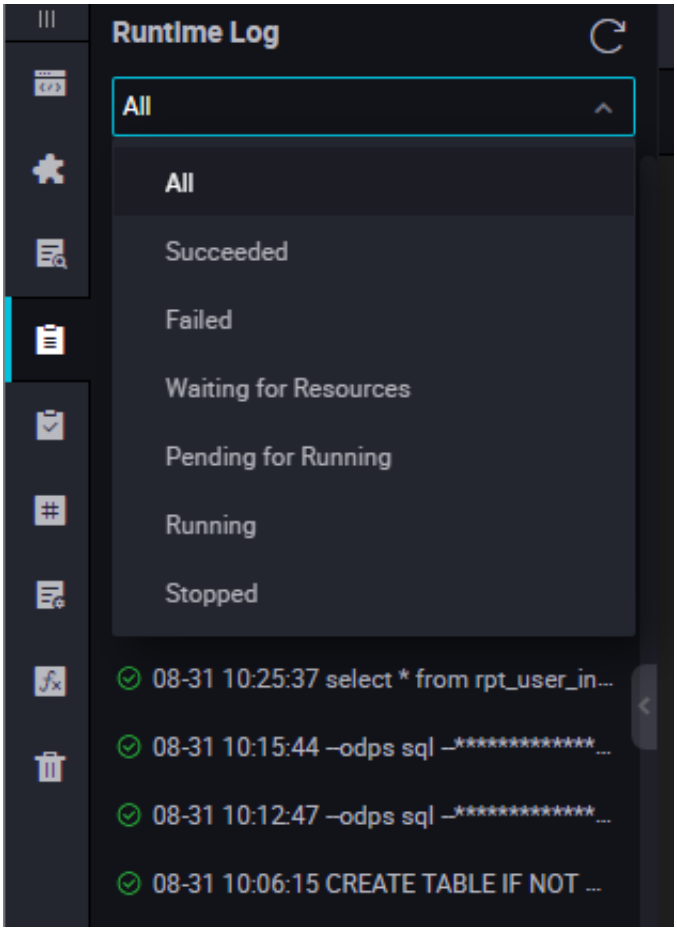
 Note:
The Running Log is only retained for three days.

View the Running Log

1. Click to switch to the Running Log page (tasks in all status are displayed by default).



2. Click the drop-down list box and select the task filter criterion.



3. Click the target running record. The Running Log page displays the log of the running record.

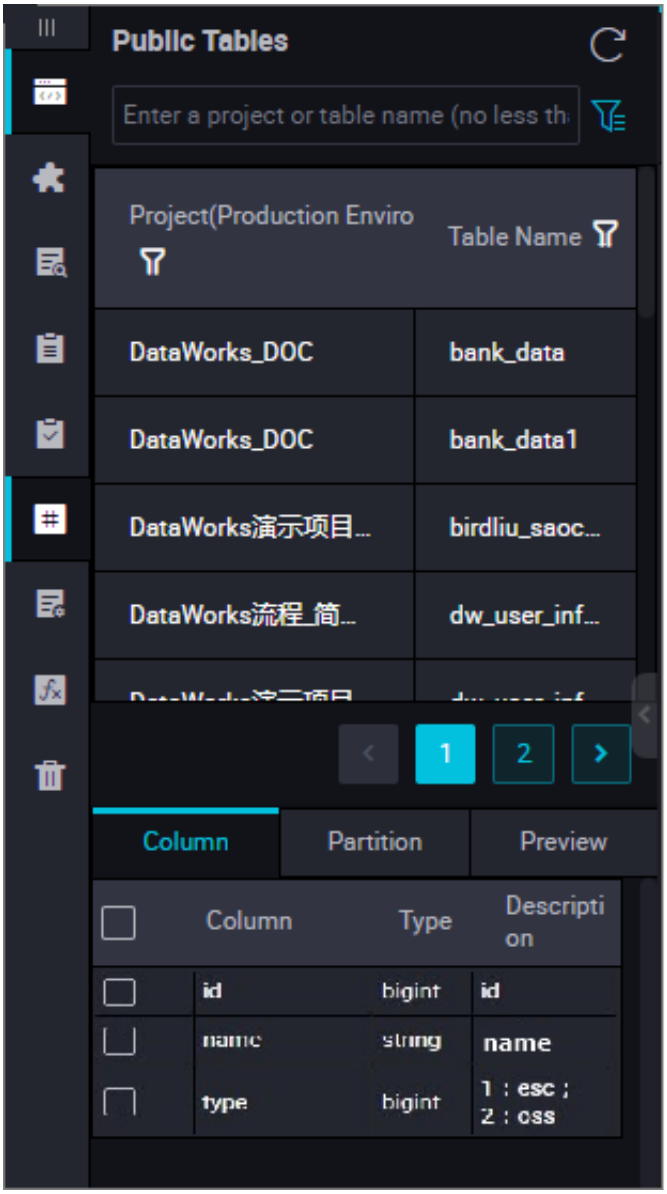
Save the log to a temporary file

To save the SQL statements in the running record, click the Save icon to save the SQL statements that have run to a temporary file.

Enter the file name and directory, and click Submit.

1.14 Public Tables

In the Public Table area, you can view tables created in all projects under the current tenant.



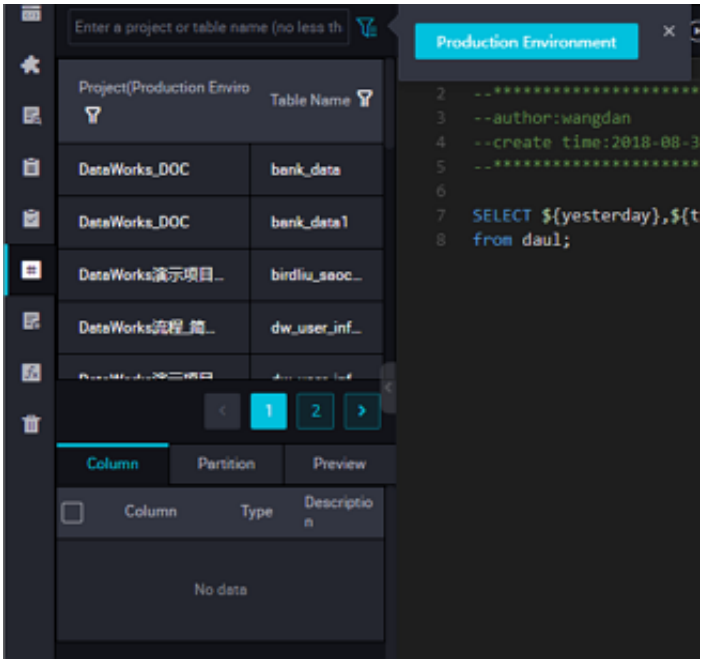
- Project: Project name. The prefix "odps." is added to each project name. For example, if a project name is "test", "odps.test" is displayed.
- Table Name: Name of the table in the project.

Click a table name to view the column and partition information of the table, and preview the table data.

- Column Information: Click it to view the field quantity, field type, and field description of the table.
- Partition Information: Click it to view the partition information and partition quantity of the table. A maximum of 60,000 partitions are allowed. If you have set the life cycle, the actual number of partitions depends on the life cycle.
- Data Preview: Click it to preview data in the current table.

Environment switchover

Similar to Table Management, Public Table supports the development and production environments. The current environment is displayed in blue. After you click an environment to be queried, the corresponding environment is displayed.

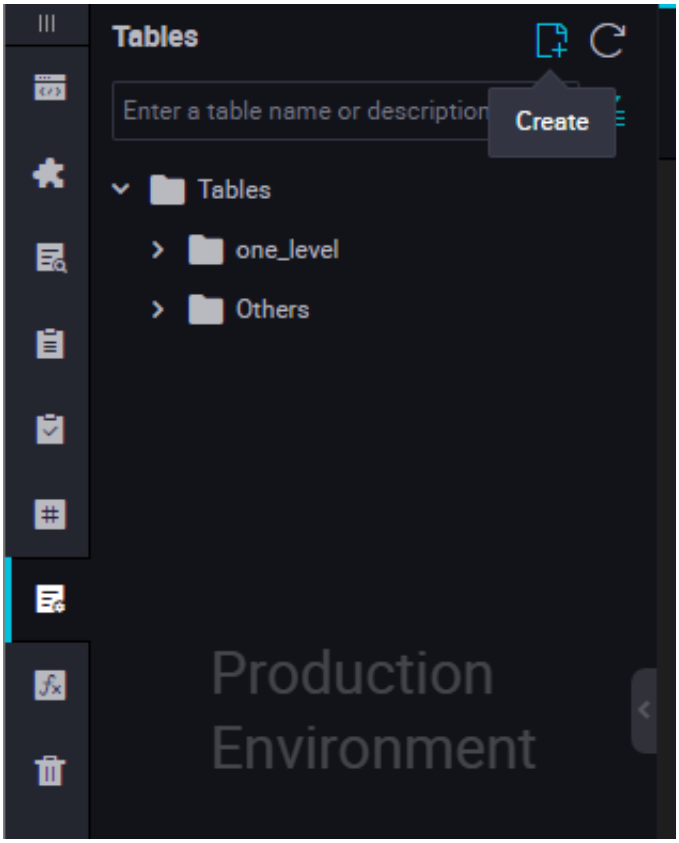


1.15 Table Management

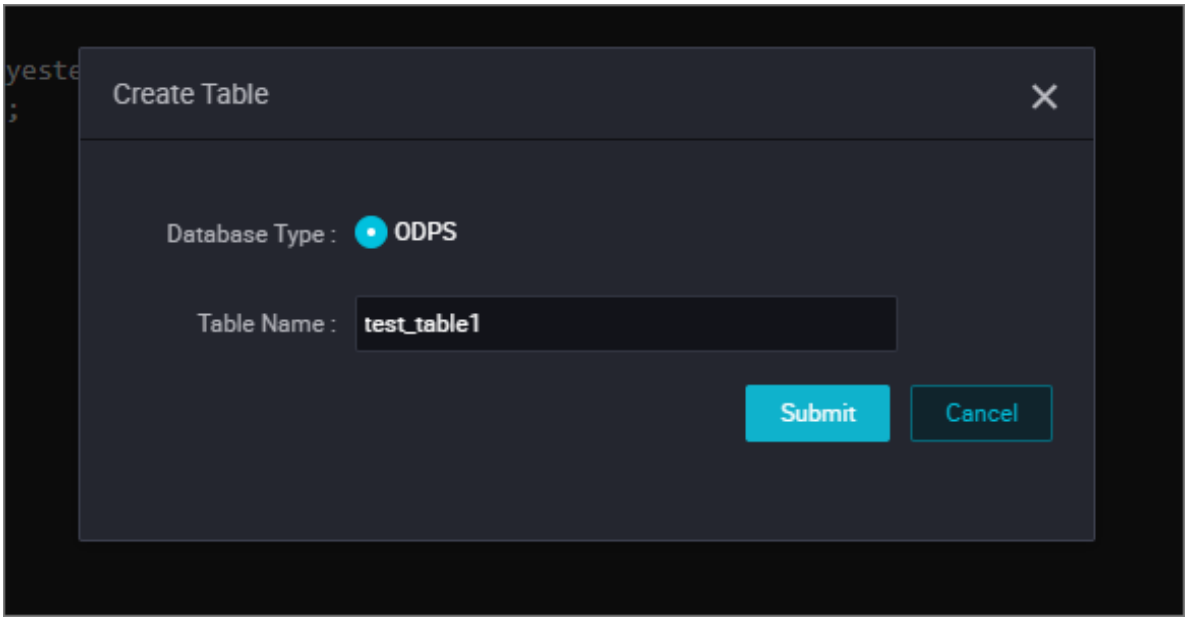
Create a table

1. Click Table Management in the upper left corner of the page.

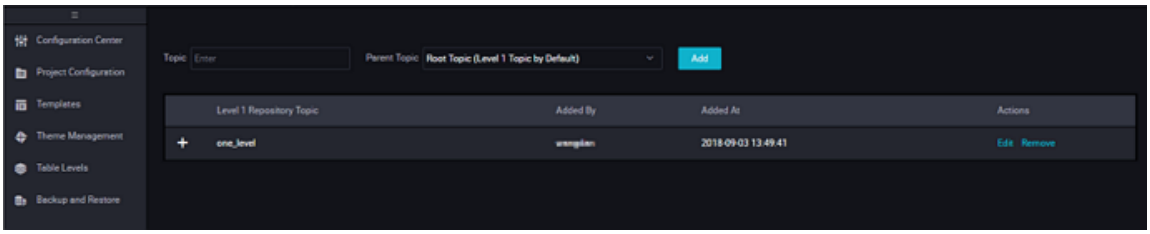
2. Select the + icon to create a table.



3. Enter the table name, only MaxCompute tables are supported currently, click Submit.



4. Set basic attributes.
   - Chinese Name: Chinese name of the table to be created.
   - Level-1 Topic: Name of the level-1 target folder of the table to be created.
   - Level-2 Topic: Name of the level-2 target folder of the table to be created.
   - Description: Description of the table to be created.
   - Click Create Topic. On the displayed Topic Management page, create level-1 and level-2 topics.



5. Create a table in DDL mode.

   Click DDL Mode. In the displayed dialog box, enter the standard table creation statements.

   After editing the table creation SQL statements, click Generate Table Structure. Information in the Basic Attributes, Physical Model Design, and Table Structure Design areas is automatically entered.

6. Create a table on the GUI

   If creating a table in DDL mode is not applicable, you can create the table on the GUI by performing the following settings.

   · Physical model design

   - Table type: It can be set to Partitioned Table or Non-partitioned Table.
   - Life Cycle:Life cycle function of MaxCompute. Data in the table (or partition) that is not updated within a period specified by Life Cycle (unit: day) will be cleared.
   - Level: It can be set to DW, ODS, or RPT.
   - Physical Category: It can be set to Basic Business Layer, Advanced Business Layer, or Other. Click Create Level. On the displayed Level Management page, create a level.

   · Table structure design

   - English Field Name: English name of a field, which may contain letters, digits , and underscores (_).
   - Chinese Name: Abbreviated Chinese name of a field.
   - Field Type: MaxCompute data type, which can only be String, Bigint, Double, Datetime, or Boolean.
   - Description: Detailed description of a field.
   - Primary Key: Select it to indicate the field is the primary key or a field in the joint primary key.
   - Click Add Field to add a column for a new field.
   - Click Delete Field to delete a created field.

      Note:
     If you delete a field from a created table and submit the table again, you must drop the current table and create one with the same name. This operation is not allowed in the production environment.

   - Click Move Up to adjust the field order of the table to be created. However, to adjust the field order of a created table, you must drop the current table

and create one with the same name. This operation is not allowed in the production environment.

- Click Move Down, the operation is the same as that of Move Up.
- Click Add Partition to create a partition for the current table. To add a partition to a created table, you must drop the current table and create one with the same name. This operation is not allowed in the production environment.
- Click Delete Partition to delete a partition. To delete a partition from a created table, you must drop the current table and create one with the same name. This operation is not allowed in the production environment.
- Action: You can confirm to submit a new field, delete a field, and edit more attributes.

  More properties mainly contain information related to data quality, which is provided for the system to generate validation logic. They are used in scenarios such as data profiling, SQL scan, and test rule generation.

  - 0 Allowed: If it is selected, the field value can be zero. This option is applicable only to bigint and double fields.
  - Negative Value Allowed: If it is selected, the field value can be a negative number. This option is applicable only to bigint and double fields.
  - Security Level： The security level is 0-4. The higher the number, the higher the security requirement. If your security level does not meet the digital requirements, you cannot access the corresponding fields in the form.
  - Unit: Unit of the amount, which can be dollar or cent. This option is not required for fields unrelated to the amount.
  - Lookup Table Name/Kay Value: It is applicable to enumerated value-type fields, such as the member type and status. You can enter the name of the dictionary table (or dimension table) corresponding to the field . For example, the name of the dictionary table corresponding to the member status is dim_user_status. If you use a globally unique dictionary table, enter the corresponding key_type of the field in the dictionary

table. For example, the corresponding key value of the member status is AOBAO_USER_STATUS.

- Value Range: The maximum and minimum values of the current field. It is applicable only to bigint and double fields..
- Regular Expression Verification: Regular expression used by the current field. For example, if a field is a mobile phone number, its value can be limited to an 11-digit number by regular expression (or more strict limitation).
- Maximum Length: Maximum number of characters of the field value. It is applicable only to string fields.
- Date Precision: Precision of the date, which can be set to Hour, Day, Month, or others. For example, the precision of month_id in the monthly summary table is Month, although the field value is 2014-08-01 (it seems that the precision is Day). It is applicable to date values of the Datetime or String type.
- Date Format: It is applicable only to date values of the string type. The format of the date value actually stored in the field is similar to yyyy-mm-dd hh:mm:ss.
- KV Primary Separator/Secondary Separator: It is applicable to a large field (of the string type) combined by KV pairs. For example, if a product expansion attribute has a value similar to "key1:value1;key2:value2;key3:value3;...", the semicolon (;) is the primary separator of the field that separates the KV pairs, and the colon (:) is the secondary separator that separates the key and value in a KV pair.

· Partition Field Design: This option is displayed only when Partition Type in the Physical Model Design area is set to Partitioned Table.
· Field Type: We recommend that you use the string type for all fields.
· Date Partition Format: If a partition field is a date (although its data type may be string), select or enter a date format, such as yyyymmmdd.
· Date Partition Granularity: For example, Day, Month, or Hour. Configure the partition granularity as per your needs. By default, if multiple partition granularities are required, the greater the granularity is, the higher the partition level is. For example, if three partitions (hour, day, and month) exist, the relationship among the multiple partitions is: level-1 partition (month), level-2 partition (day), and level-3 partition (hour).

Submit a table

After editing the table structure information, submit the new table to the development environment and production environment.

- Click Load from Development Environment. If the table has been submitted to the development environment, this button is highlighted. After you click the button, the information of the created table in the development environment overwrites the information on the current page.
- Click Submit to Development Environment. The system checks whether all required items on the current editing page are completely set. If any omission exists, an alarm is reported, forbidding you to submit the table.
- Click Load from Production Environment. The detailed information of the table submitted to the production environment overwrites the information on the current page.
- Click Create in Production Environment. The table is created in the project of the production environment.

Query tables by type

On the Table Management page, you can select Development Environment or Production Environment to query tables. The query results are sorted by folder of topics.

- If you select Development Environment, you can only query tables in the development environment.
- If you select Production Environment, you can query tables in the production environment. Be cautious when operating the tables in the production environmen t.

1.16 External tables

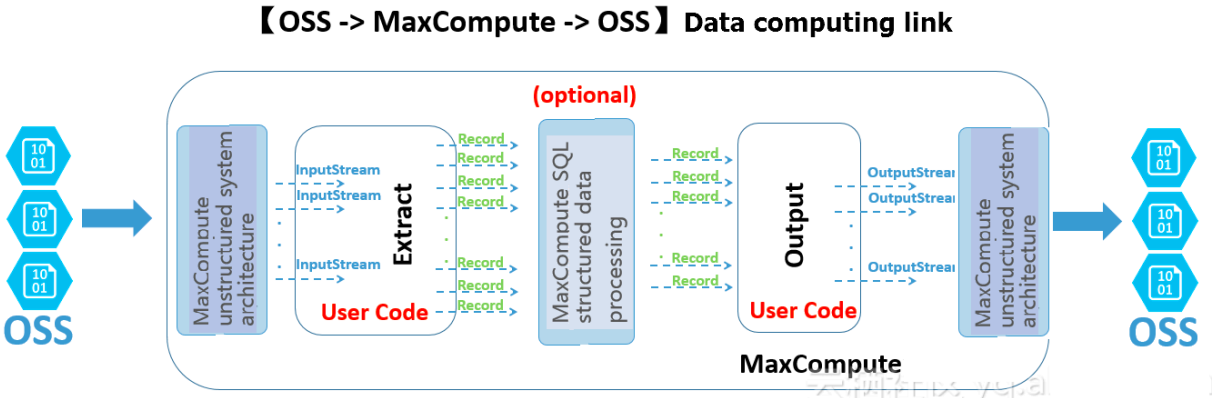
External table overview

Before you use external tables, you need to understand the following concepts.

Name	Description

Object Storage Service（OSS）	OSS supports Standard, Infrequent Access, and Archive storage types. It is applicable to service scenarios that involve different requirements for data storage and access. Additionally, OSS supports seamless integration with Apache Hadoop, E-MapReduce, BatchCompute, MaxCompute, Machine Learning Platform for AI (PAI), Data Lake Analytics, Function Compute, and other Alibaba Cloud services.
MaxCompute	The big data computing service is a fast and fully-managed data warehousing solution. When used in conjunction with OSS, it enables you to effectively analyze and process large-scale data with reduced costs. Forrester names MaxCompute as one of the world's leading cloud-based data warehouses because of its processing performance.
External tables of MaxCompute	This function is based on the new generation of the computing framework of MaxCompute v2.0. It allows you to directly query data that is stored in OSS without loading data into the internal tables of MaxCompute. This not only saves time and effort for data migration but also saves costs for storage of duplicate data. You can use the external tables of MaxCompute to query data that is stored in Table Store in a similar way.

The following figure shows the processing architecture of the external tables.



Currently, MaxCompute supports processing external tables in the storage of unstructured data such as OSS and Table Store. Based on the flow of data and the processing rules, you can understand that the main function of the unstructured data processing framework is to import and export data and connect the input and output of MaxCompute. The following example describes the processing rules applied to external tables in OSS.

1. Data stored in OSS is converted through the unstructured data processing framework and passed to user-defined interfaces using the InputStream Java class

. To implement the extracting rules, you need to read, parse, convert, and calculate the input streams. The data must be returned in the record format, which is the general format in MaxCompute.

2. These records can be used in structured data processing based on the SQL engine built into MaxCompute to generate new records.
3. You can perform further calculations before the data of records are output through the OutputStream Java class and are imported into OSS by MaxCompute.

You can create, search, query, configure, process, and analyze external tables in GUI through DataWorks, which is powered by MaxCompute.

Network and access authorization

Since MaxCompute is separate from OSS, network connectivity between them on different clusters may affect the ability of MaxCompute to access the data stored in OSS. We recommend that you use the private endpoint (it ends with `-internal.aliyuncs.com`) to access the data stored in OSS through MaxCompute.

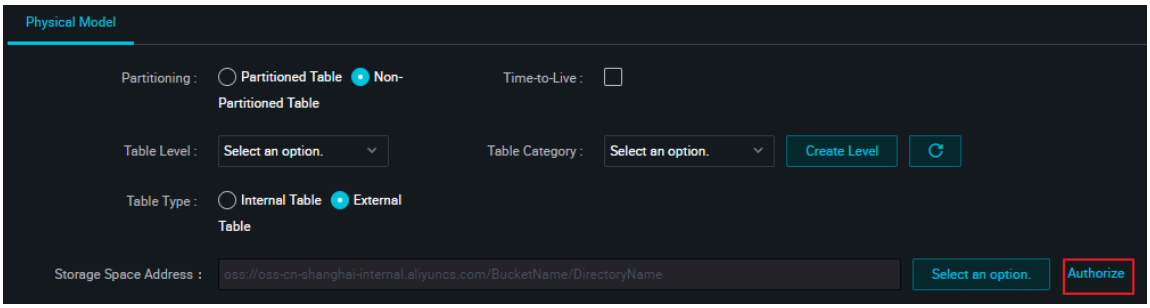
Authorization is required for MaxCompute to access data stored in OSS. MaxCompute guarantees secure access to data using Resource Access Management (RAM) and Security Token Service (STS) provided by Alibaba Cloud. You request the STS token for MaxCompute as the table creator. Therefore, MaxCompute and OSS must be under the same Alibaba Cloud account. A similar authorization process applies when accessing data stored in Table Store.

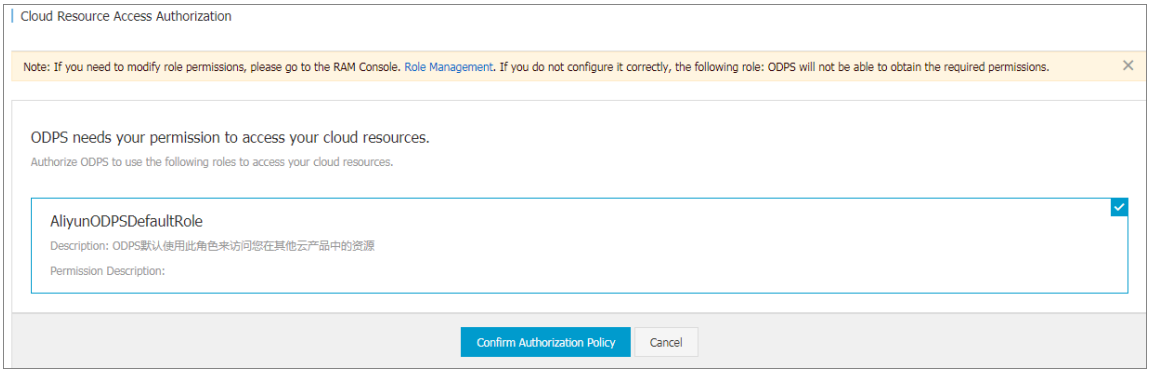
1. STS authorization

   If MaxCompute requires direct access to data stored in OSS, you need to grant the OSS access to RAM users first. Security Token Service (STS) is a security token management service provided by Alibaba Cloud. It is a product based on Resource Access Management (RAM). Authorized RAM users can issue tokens with custom validity and access through STS. Applications can use tokens to directly call Alibaba Cloud APIs to manipulate resources. For more information, see OSS STS

mode authorization. You can choose either of the following methods to grant access.

- If MaxCompute and OSS are under the same Alibaba Cloud account, log on and perform Authorize. You can click Data Development and Create Table to jump to the Authorize page as shown in the following figure.





- Custom authorization. First, you need to grant MaxCompute access to OSS through RAM. Log on to the RAM console (if MaxCompute and OSS are under different Alibaba Cloud accounts, use the account for OSS to log on). Go to the Role Management page and click Create Role. Set the value of Role Name to AliyunODPSDefaultRole or AliyunODPSRoleForOtherUser.

  Configure Role Details.

```
-- When   MaxCompute   and   OSS   are   under   the   same
 Alibaba   Cloud   account .
{
" Statement ": [
{
" Action ": " sts : AssumeRole ",
" Effect ": " Allow ",
" Principal ": {
" Service ": [
" odps . aliyuncs . com "
      ]
    }
  }
],
" Version ": " 1 "
}
```

```
-- When   MaxCompute   and   OSS   are   under   different
 Alibaba   Cloud   accounts .
{
" Statement ": [
{
" Action ": " sts : AssumeRole ",
" Effect ": " Allow ",
" Principal ": {
" Service ": [
" Alibaba   Cloud   account   for   MaxCompute @ odps . aliyuncs .
 com "
      ]
    }
  }
],
" Version ": " 1 "
}
```

Configure Role Authorization Policies. Search for the AliyunODPSRolePolicy policy that is required for granting OSS access. Attach the AliyunODPSRolePolicy policy to the role. If you can not find this policy through Search and Attach, authorize the role through Input and Attach. The policy content of the AliyunODPSRolePolicy policy is shown as follows.

```
{
  " Version ": " 1 ",
  " Statement ": [
    {
      " Action ": [
        " oss : ListBucket  s ",
        " oss : GetObject ",
        " oss : ListObject  s ",
        " oss : PutObject ",
        " oss : DeleteObje  ct ",
        " oss : AbortMulti  partUpload ",
        " oss : ListParts "
        ],
        " Resource ": "*",
        " Effect ": " Allow "
  },
  {
      " Action ": [
        " ots : ListTable ",
        " ots : DescribeTa  ble ",
        " ots : GetRow ",
        " ots : PutRow ",
        " ots : UpdateRow ",
        " ots : DeleteRow ",
        " ots : GetRange ",
        " ots : BatchGetRo  w ",
        " ots : BatchWrite  Row ",
        " ots : ComputeSpl  itPointsBy  Size "
      ],
      " Resource ": "*",
      " Effect ": " Allow "
    }
  ]
```

```
}
```

2. **Using OSS data sources in Data Integration**

   You can directly use the OSS data sources that have already been created in Data Integration.

Create external tables

1. **Use DDL statements to create tables**

   Go to the Data Development page. See Table Management and use DDL statements to create tables. You need to follow the ODPS syntax (See Table Operations). If you have STS authorization, then you do not need to include the `odps . properties . rolearn` attribute. The following example shows how to use DDL statements to create a table. The `EXTERNAL` keyword in the statement indicates that this table is an external table.

```
CREATE EXTERNAL TABLE IF NOT EXISTS ambulance_
data_csv_e xternal (
vehicleId int ,
recordId int ,
patientId int ,
calls int ,
locationLa titute double ,
locationLo ngtitue double ,
recordTime string ,
direction string
)
STORED BY ' com . aliyun . odps . udf . example . text .
TextStorag eHandler ' -- The STORED BY clause specifies
the StorageHan dler for the correspond ing file
format . This clause is required .
with SERDEPROPE RTIES (
' delimiter '='\\|', -- The SERDEPROPE RITES clause specifies
the parameters used when serializin g or deserializ
ing data . These parameters are passed into the code
of Extractor through DataAttrib utes . This clause is
optional .
' odps . properties . rolearn '=' acs : ram :: xxxxxxxxxx xxx :
role / aliyunodps defaultrol e '
)
LOCATION ' oss :// oss - cn - shanghai - internal . aliyuncs . com
/ oss - odps - test / Demo / SampleData / CustomTxt / AmbulanceD
ata /' -- The LOCATION clause specifies the
location of the external tables . This clause is
optional .
USING ' odps - udf - example . jar '; -- The USING clause
specifies the Jar files that store the user - defined
```

```
   classes . This clause is optional , depending on
 whether you use user - defined classes .
```

The parameters following `STORED BY` that are corresponding to the built-in storage handlers for csv or tsv files are shown as follows:

- The `com . aliyun . odps . CsvStorage Handler` parameter is for CSV format. It defines how to read and write data in CSV format. The format has columns separated by the comma (,) and rows terminated by the newline character (\n). For example, `STORED BY ' com . aliyun . odps . CsvStorage Handler '` is a sample parameter.
- The `com . aliyun . odps . TsvStorage Handler` parameter is for TSV format. It defines how to read and write data in TSV format. The format has columns separated by the tab character (\t) and rows terminated by the newline character (\n).

The parameters following `STORED BY` also support specifying the storage handlers for the open-source file formats such as TextFile, SequenceFile, RCFile, AVRO, ORC, and Parquet. For TextFile formats, you can specify the SerDe class. For example, `org . apache . hive . hcatalog . data . JsonSerDe` .

- org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe -> stored as textfile
- org.apache.hadoop.hive.ql.io.orc.OrcSerde -> stored as orc
- org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe -> stored as parquet
- org.apache.hadoop.hive.serde2.avro.AvroSerDe -> stored as avro
- org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe -> stored as sequencefile

For external tables that are in the open-source formats, the statements to create tables are as follows.

```
 CREATE EXTERNAL TABLE [ IF NOT EXISTS ] (< column
 schemas >)
[ PARTITIONE D BY ( partition column schemas )]
[ ROW FORMAT SERDE '']
 STORED AS
[ WITH SERDEPROPE RTIES ( ' odps . properties . rolearn '='${
 roleran }'
[,' name2 '=' value2 ',...]
) ]
```

```
LOCATION  ' oss ://${ endpoint }/${ bucket }/${ userfilePa  th }/';
```

Attributes of the SERDEPROPERTIES clause are shown in the following table. Currently, for gzip-compressed data from CSV and TST files in OSS, MaxCompute only supports reading through the built-in extractor. You can choose whether the file is gzip-compressed. Attribute settings are different based on file formats.

Attribute	Value	Default value	Description
odps.text.option.gzip.input.enabled	true/false	false	Enables or disables the reading of compressed data.
odps.text.option.gzip.output.enabled	true/false	false	Enables or disables the writing of compressed data.
odps.text.option.header.lines.count	N (a non-negative integer)	0	Skip the first N lines of the file.
odps.text.option.null.indicator	String	""	Replaces NULL with the value of the string.
odps.text.option.ignore.empty.lines	true/false	true	Specifies whether to ignore blank lines.
odps.text.option.encoding	UTF-8/UTF-16/US-ASCII	UTF-8	Specifies the encoding set of the file.

The `LOCATION` clause specifies the storage address of the external table in the format of oss://oss-cn-shanghai-internal.aliyuncs.com/BucketName/DirectoryName. You can select the directory in OSS through the dialog boxes. Do not select the files.

You can find tables that are created using DDL statements in the node directories in the Tables tab. You can modify Level 1 Topic or Level 2 Topic to change the directories for the tables.

2. External tables in Table Store

   The statements to create external tables in Table Store are as follows.

```
CREATE   EXTERNAL   TABLE   IF   NOT   EXISTS   ots_table_
external (
odps_order  key   bigint ,
odps_order  date   string ,
```

```
 odps_custk  ey   bigint ,
 odps_order  status   string ,
 odps_total  price   double
)
 STORED   BY  ' com . aliyun . odps . TableStore  StorageHan  dler
 '
 WITH   SERDEPROPE  RTIES  (
' tablestore . columns . mapping '=': o_orderkey ,: o_orderdat  e ,
 o_custkey ,  o_ordersta  tus , o_totalpri  ce ', -- ( 3 )
' tablestore . table . name '=' ots_tpch_o  rders '
' odps . properties . rolearn '=' acs : ram :: xxxxx : role /
 aliyunodps  defaultrol  e '
)
 LOCATION  ' tablestore :// odps - ots - dev . cn - shanghai . ots -
 internal . aliyuncs . com ';
```

Description:

- `com . aliyun . odps . TableStore  StorageHan  dler` **is the MaxCompute built-in storage handler to process data in Table Store.**
- `SERDEPROPE  RITES` **provides options for parameters. You must specify tablestore.columns.mapping and tablestore.table.name when using TableStoreStorageHandler.**
  - `tablestore . columns . mapping` **: This parameter is required. It describes the columns of the table in Table Store that MaxCompute accesses, including the primary key columns and property columns. A primary key column is indicated with the colon sign (:) at the beginning of the column name. In this example, primary key columns are p:o_orderkey and :o_orderdate. The others are property columns. Table Store supports up to four primary key columns. The data types include String, Integer and Binary. The first column of the primary key is the partition key. You must specify all primary key columns of the table in Table Store when specifying the mapping. You only need to specify the property columns that MaxCompute accesses instead of specifying all property columns..**
  - `tablestore . table . name` **: The name of the table to access in Table Store. If the table name does not exist in Table Store, an error is reported. MaxCompute does not create a table in Table Store.**
- `LOCATION` **: Specifies the name and the endpoint of the Table Store instance.**

3. Create a table in GUI

   Go to the Data Development page, see Table Management to create a table in GUI. An external table has the following attributes.

   · Basic attributes

   - Table name (Create a table and enter the name)
   - Table alias
   - Level 1 Topic and Level 2 Topic
   - Description

   · Physical model

   - Table type: Select External table.
   - Partition: External tables in Table Store do not support partitioning.
   - Select the memory address: Specify the `LOCATION` clause. You can specify the `LOCATION` clause in the Physical model section. Select an option the storage location of the external table in the dialog box. Then you can perform Authorize.
   - Select storage format: Select the file format as required. CSV, TSV, TextFile, SequenceFile, RCFile, AVRO, ORC, and Parquet, and custom file formats are supported. If you select a custom file format, you need to select the

corresponding resource. The classes are parsed from the resources automatically when you submit the resources. You can select the class name.

- `rolearn` : If you have STS authorization, you do not need to specify the rolearn attribute.

· Table structure design

- Data type: MaxCompute 2.0 supports INYINT, SMALLINT, INT, BIGINT, VARCHAR and STRING types for fields.
- Actions: You can create, modify, and delete the fields.
- Length/Set: You can set the maximum length of the VARCHAR type columns. For composite data types, you can fill in the definitions for them.

Supported data type

Basic data types that are supported by external tables are shown in the following table.

Data type	New	Examples	Description
TINYINT	Yes	1Y, -127Y	A signed eight-bit integer in the range -128 to 127.
SMALLINT	Yes	32767S, -100S	A signed 16-bit integer in the range -32,768 to 32,767.
INT	Yes	1000, -15645787	A signed 32-bit integer in the range -231 to 231-1.
BIGINT	No	100000000000L, -1L	A signed 64-bit integer in the range -263 + 1 to 263 - 1.
FLOAT	Yes	None	A 32-bit binary floating point number.

DOUBLE	No	3.1415926 1E+7	An eight-byte double precision floating-point number (a 64-bit binary floating point number).
DECIMAL	No	3.5BD, 9999999999 9.9999999BD	A decimal exact numeric. Precision can range from -1036 + 1 to 1036 -1, scale from 10 to 18.
VARCHAR(n)	Yes	None	A variable-length character string. The length is n that is in the range 1 to 65535.
STRING	No	“abc”, ’ bcd ’, ” alibaba”	A string. Currently, the maximum length is 8M.
BINARY	Yes	None	A binary number. Currently, the maximum length is 8M.
DATETIME	No	DATETIME ‘2017-11-11 00:00:00’	The data type for dates and times. UTC–8 is used as the standard time of the system. The range is from 0000-01- 01 to 9999-12-31, accurate to a millisecond.
TIMESTAMP	Yes	TIMESTAMP ‘2017-11-11 00:00:00.123456789’	TIMESTAMP data type, which is independent of time zones. The range is from 0000-01-01 to 9999-12-31, accurate to a nanosecond.
BOOLEAN	No	TRUE, FALSE	Logical Boolean (TRUE/FALSE)

Composite data types supported by external tables are shown in the following tables.

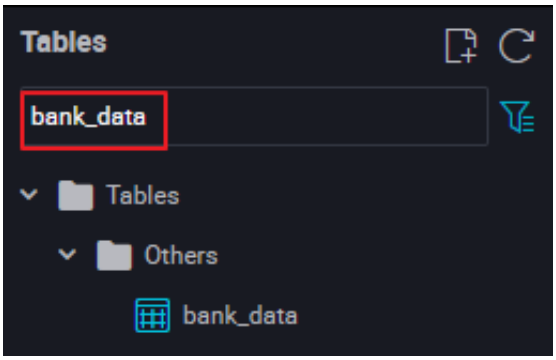
Type	Definition	Constructor
ARRAY	array< int >; array< struct< a:int, b:string >>	array(1, 2, 3); array(array(1, 2); array(3, 4))
MAP	map< string, string >; map < smallint, array< string>>	map(“k1”, “v1”, “k2”, “v2”); map(1S, array(‘a’, ‘b’), 2S, array(‘x’, ‘y))

STRUCT	struct< x:int, y:int>; struct < field1:bigint, field2:array < int>, field3:map< int, int >>	named_struct(‘x’ , 1, ‘ y’ , 2); named_struct(‘ field1’ , 100L, ‘field2’ , array(1, 2), ‘field3’ , map (1, 100, 2, 200)

If you need to use data types newly supported by MaxCompute 2.0 (TINYINT, SMALLINT, INT, FLOAT, VARCHAR, TIMESTAMP, BINARY or composite data types), you need to include `set odps . sql . type . system . odps2 = true ;` before the statements to create a table. Submit and execute the statements to create a table with the set statement. If compatibility with HIVE is required, we recommend that you include the `odps . sql . hive . compatible = true ;` statement.

View and process external tables

You can find the external tables in the Tables view.

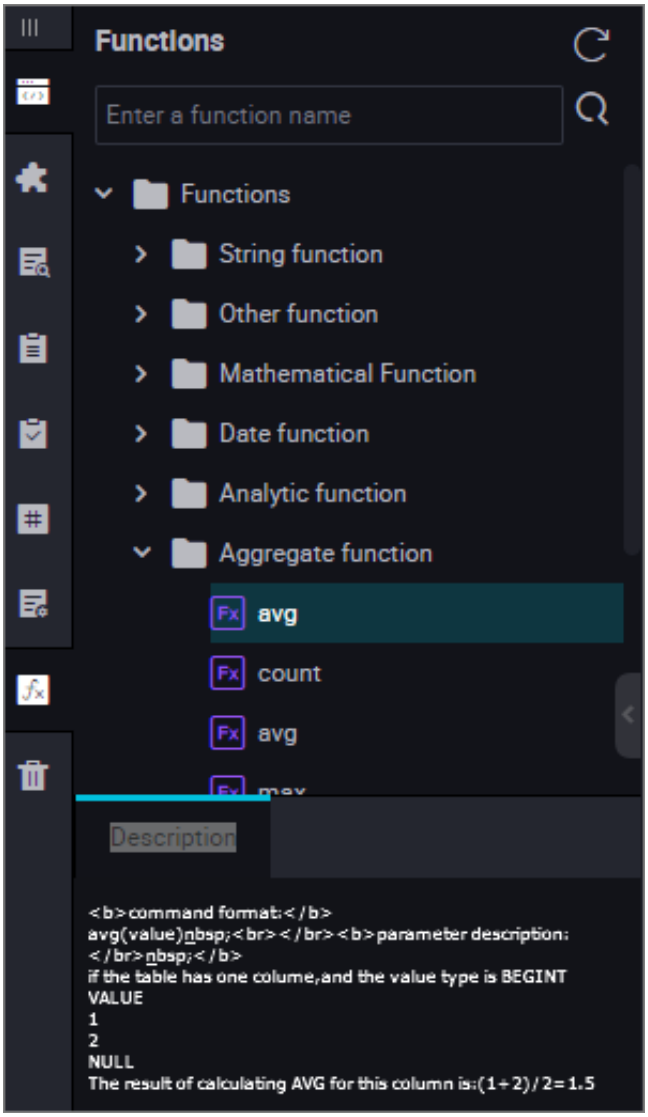


The processing of external tables is similar to that of internal tables. For more information about external tables, see #unique_15 and #unique_122.

1.17 Functions

The function list provides the currently available functions, function classification, function usage description, and instances.

The function list contains six parts, including other functions, string processing functions, mathematical functions, date functions, window functions, and aggregate functions. These functions are provided by the system. You can view the description and example of a function by dragging the function.



1.18 Editor shortcut list

Common shortcuts for code editing.

Windows chrome version

`Ctrl + S` Save

`Ctrl + Z` Undo

`Ctrl + Y` Redo

`Ctrl + D` Select the same word

`Ctrl + X` Cut a row

`Ctrl + Shift + K` Delete a row

`Ctrl + C` Copy the current row

`Ctrl + i` Select a row

`Shift + Alt + Dragging with the mouse` Column mode editing, modifying all the contents in this part

`Alt + mouse` Click multi-column mode edit, multi-line indents

`Ctrl + Shift + L` Add a cursor for all the identical string instances, batch changes

`Ctrl + F` Find

`Ctrl + H` Replace

`Ctrl + G` Locate to a specified row

`Alt + Enter` Select all the matching keywords in search

`Alt ↓ / Alt ↑` Move the current row down/up

`Shift + Alt + ↓ / Shift + Alt + ↑` Copy the current row down/up

`Shift + Ctrl + K` Delete the current row

`Ctrl + Enter / Shift + Ctrl + Enter` Move the cursor down/up

`Shift + Ctrl + \` Jump the cursor to the matching brackets

`Ctrl + ] / Ctrl + [` Increase/decrease indent

`Home / End` Move to the beginning/end of the current row

`Ctrl + Home / Ctrl + End` Move to the beginning/end of the current file

`Ctrl + → / Ctrl + ←` Move the cursor right/left by words

`Shift + Ctrl + [ / Shift + Ctrl + ]` Hide/Show block pointed by cursor

`Ctrl + K + Ctrl + [ / Ctrl + K + Ctrl + ]` Hide/Show subblock pointed by cursor

`Ctrl + K + Ctrl + 0 / Ctrl + K + Ctrl + j` Fold/unfold all areas

`Ctrl + /` Write/Cancel comments for the row or code block where the cursor stays

MAC chrome version

`cmd + S` Save

`cmd + Z` Undo

`cmd + Y` Redo

`cmd + D` Select the same word

`cmd + X` Cut a row

`cmd + shift + K` Delete a row

`cmd + C` Copy the current row

`cmd + i` Select the current row

`cmd + F` Find

`cmd + alt + F` Replace

`alt ↓ / alt ↑`Move the current row down/up

`shift + alt + ↓ / shift + alt + ↑` Copy the current row down/up

`shift + cmd + K` Delete the current row

`cmd + Enter / shift + cmd + Enter` Move the cursor down/up

`shift + cmd + \` Jump the cursor to the matching brackets

` `cmd +] / cmd + [` Increase/decrease indent

`cmd + ← / cmd + →` Move to the beginning/end of the current row

`cmd + ↑ / cmd + ↓` Move to the beginning/end of the current file

`alt + → / alt + ←` Move the cursor right/left by words

`alt + cmd + [ / alt + cmd + ]` Hide/Show block pointed by cursor

`cmd + K + cmd + [ / cmd + K + cmd + ]` Hide/Show subblock pointed by cursor

`cmd + K + cmd + 0 / cmd + K + cmd + j` Fold/unfold all areas

`cmd + /`Write/Cancel comments for the row or code block where the cursor stays

Multiple cursors/select

`alt + Clicking with the mouse` Insert the cursor

`alt + cmd + ↑/↓` Insert the cursor up/down

`cmd + U` Undo the last cursor operation

`shift + alt + I` Insert a cursor to the end of each row of the selected code block

`cmd + G / shift + cmd + G` Find the next/previous item

`cmd + F2` Select all the characters that the mouse has chosen

`shift + cmd + L` Select all the parts that the mouse has chosen

`alt + Enter` Select all the matching keywords in search

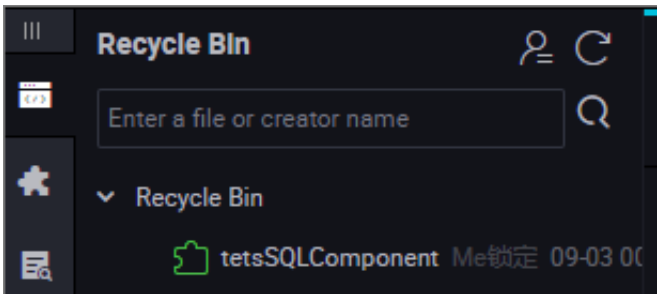
`shift + alt + Dragging with the mouse` Select multi-columns for editing

`shift + alt + cmd + ↑ / ↓` Move the cursor up/down to select multi-columns for editing

`shift + alt + cmd + ← / →` Move the cursor right/left to select multi-columns for editing

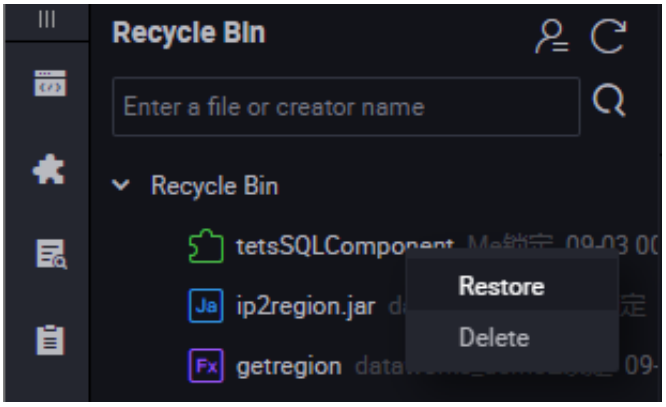
1.19 Recycle Bin

DataWorks has its own recycle bin, click Recycle Bin in the upper left corner of the page.



On the Recycle Bin page, you can check all deleted nodes in the current project. You can also right-click a node to restore or permanently delete it.

Click Show My Files on the right of the Recycle Bin page to view your deleted nodes.



Note:

If a node is permanently deleted from the recycle bin, it cannot be restored.

2 DataService studio

2.1 DataService studio overview

DataService Studio aims to build a data service bus to help enterprises centrally manage private and public APIs. DataService Studio allows you to quickly create APIs based on data tables and register existing APIs with the DataService Studio platform for centralized management and release. In addition, DataService Studio is connected to API Gateway. You can deploy APIs to API Gateway with one-click. DataService Studio works together with API Gateway to provide a secure, stable, low-cost, and easy-to-use data sharing service.

DataService Studio adopts the serverless architecture. All you need to care is the query logic of APIs, instead of the infrastructure such as the running environmen t. DataService Studio prepares the computing resources for you, supports elastic scaling, and requires zero O&M cost.

Creation of data APIs

DataService Studio currently supports the use of the visualized wizard to quickly create data APIs based on tables of the relational database and NoSQL database. You can configure a data API in several minutes without writing codes. To meet the personalized query requirements of advanced users, DataService Studio provides the custom SQL script mode to allow you compile the API query SQL statements by yourself. It also supports multi-table association, complex query conditions, and aggregate functions.

API registration

DataService Studio also supports centralized management of the existing API services that you register with DataService Studio and the APIs created based on data tables . Currently only RESTful APIs can be registered. Supported request methods include GET, POST, PUT, and DELETE. Supported data types include forms, JSON data, and XML data.

API gateway

API Gateway provides API management services, including API publish, management , and maintenance, and API subscription duration management. It provides you with

a simple, fast, low-cost, and low-risk method to implement microservice aggregation, frontend-backend isolation, and system integration, and opens functions and data to partners and developers.

DataService Studio has been connected to API Gateway. You can deploy any APIs created and registered in DataService Studio to API Gateway for management, such as API authorization and authentication, traffic control, and metering.

API Market

The Ali cloud API market is the most comprehensive API trading market in China , covering finance, artificial intelligence, e-commerce, transportation geography , Living Services, corporate management and the eight main categories of public affairs, thousands of API products have been sold online.

After your APIs from DataService Studio have been published to API Gateway, you can then publish them to Alibaba Cloud API Marketplace. This is an easy way to achieve financial gains for your company.

2.2 Glossary

The data services related words are explained below.

- Data sources: database links. Data Service accesses data through data sources. Data sources can only be configured in Data Integration.
- Create APIs: create APIs based on data tables.
- Register APIs: register existing APIs to Data Service for central management.
- Wizard: guides you through the procedure of API creation. This method is suitable for beginners who want to create simple APIs. You do not need to write any code.
- Script: allows you to create APIs by writing SQL scripts. This method supports table join queries, complex queries, and aggregate functions. This method is suitable for experienced developers who want to create complex APIs.
- API groups: an API group is a set of APIs for a certain scenario or for consuming a specific service. API groups are the smallest group units in Data Service, as well as the smallest units managed by API Gateway. API groups are published in Alibaba Cloud API Market as API products.
- API Gateway: a service provided by Alibaba Cloud to manage APIs. API Gateway supports API subscription duration management, permission management, access management, and traffic control.

- API Market: Alibaba Cloud API Market is the most complete and integrated domestic API trading platform established on Alibaba Cloud Market.

2.3 Generate API

2.3.1 Configure the Data Source

Before you can use the data API to generate a service, you must configure the data source in advance. Data Service allows you to obtain schema information of data tables from data sources and handle API requests.

You can configure a data source on the data integration > data source page in the dataworks console, support for different data source types and how to configure them is shown in the following table.

Data source name	Wizard mode to generate data API	Script Mode generation data API	Configuration method
RDS (ApsaraDB for RDS)	Supported	Supported	The RDS includes MySQL, PostgreSQL, and SQL Server.
DRDS	Supported	Supported	#unique_131
MySQL	Supported	Supported	#unique_132
PostgreSQL	Supported	Supported	#unique_133
SQL Server	Supported	Supported	#unique_134
Oracle	Supported	Supported	#unique_135
AnalyticDB(ADS)	Supported	Supported	#unique_136
Table Store(OTS)	Yes	No	#unique_137
MongoDB	Supported	No	#unique_138

2.3.2 Overview of generating API

The Data Service currently supports faster generation of tables from relational and neosql databases through a visually configured wizard mode. data API, you don't need to have the ability to code to configure a data API in a matter of minutes. To meet the personalized query requirements of advanced users, Data Service provides the custom SQL script mode to allow you compile the API query SQL statements by

yourself. It also supports multi-table association, complex query conditions, and aggregate functions.

The functions of the wizard mode and the script mode are listed as follows:

Features	Features	Wizard mode	Script Mode
Query object	Query a single data table from one data source	Supported	Supported
	Query multiple joined tables from one data source	No	Supported
Filter bar	Query for an exact number	Supported	Supported
	Query for a range of numbers	No	Supported
	Match an exact string	Supported	Supported
	Fuzzy search for strings	Supported	Supported
	Set required and optional parameters	Supported	Supported
Query results	Return the field value	Supported	Supported
	Return a mathematical calculation of field values	No	Supported
	Return an aggregate calculation of field values	No	Supported
	Display results with pagination	Supported	Supported

2.3.3 Generate API in Wizard Mode

This article will introduce you to the steps and considerations of the wizard mode generation API.

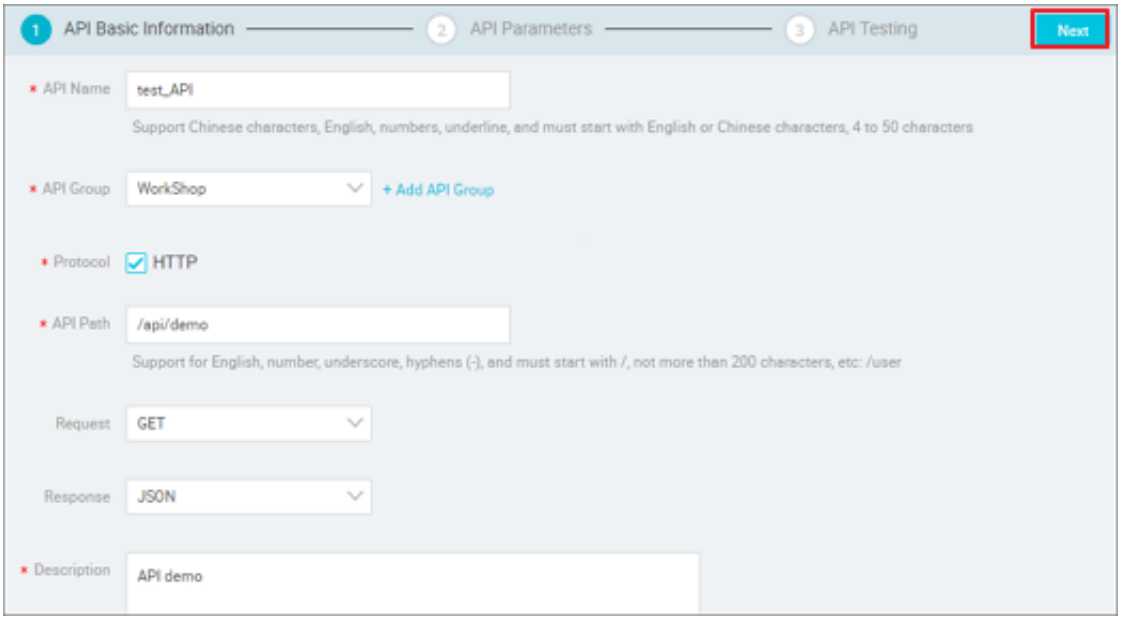
Using the wizard mode to generate data, the API is simple and easy to get started without writing any code, the API can be quickly generated by checking the configuration from the product interface. We recommend that users who do not have high requirements for the functions of the API or have little code development experience use the wizard.

 Note:

Before you configure the API, configure the data source in the Data integration > Data Source page of the dataworks console.

Configure the API basic information

1. Navigate to the API Service list > Generate API.
2. Click Wizard Mode to fill in the API basics.



Note the settings for the API grouping during configuration. An API group includes a collection of APIs that are used for a specific scenario. It is the minimum management unit in API Gateway. In the Alibaba Cloud API Market, each API group corresponds to a specific API product.

> **Note:**
> The set up example for API grouping is as follows:
>
> For example, you would like to configure an API product for weather inquiry, weather search API by city name weather search API, scenic spot name search weather API and zip search weather API three kinds of APIS, then you can create an API group called a weather query, and put the above three APIs in this group. The API is shown as a weather query product when published to the market.
>
> Of course, if your generated API is used in your own app, you can use grouping as a classification.

Currently, the build API only supports HTTP protocol, GET request mode, and JSON return type.

3. After providing the API basic information, click Next to go to the API parameter configuration page.

Configure API parameters

1. Navigate to the Data source type > Data source name > Table and select the tables that you want to configure.

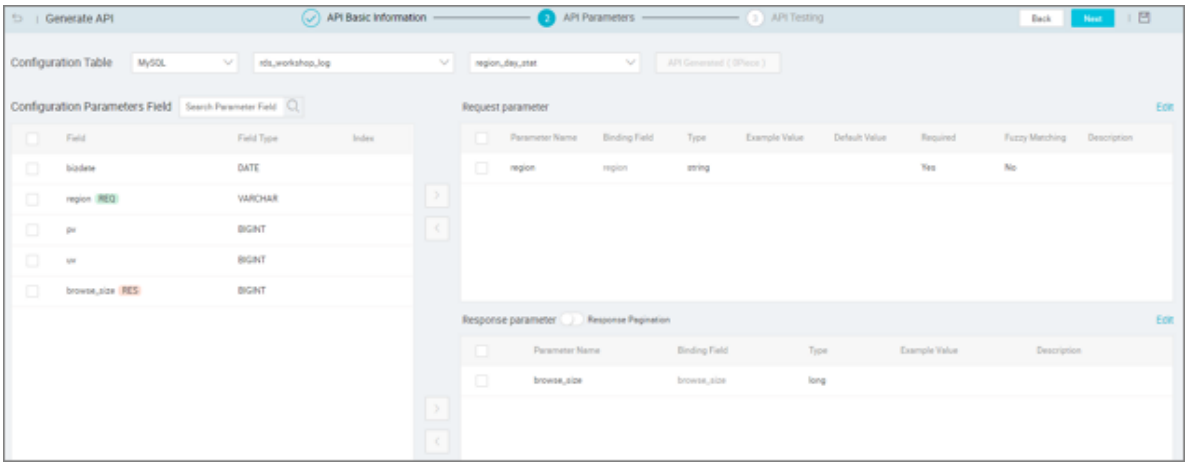
   > **Note:**
   > You need to configure the data source in advance in the data set, and the data table drop-down box supports the table name search.

2. Second, specify request and response parameters.

   When a data table has been selected, all fields of the table are displayed on the left . Select the fields to be used as request parameters and response parameters, then add them to the corresponding parameter list.

3. Finally, edit and complete parameter information.

   Click Edit in the upper-right corner of the request and return parameter lists to enter the parameter information Edit page, sets the name of the parameter, sample value, default, mandatory, fuzzy match (only string type is supported) settings) and the description. The optional and description fields are required.



You need to pay attention to the settings that return result paging during the configuration process.

- If you do not enable the response pagination, the API outputs up to 500 records by default.
- If the return result may exceed 500, turn on the response pagination function.

The following public parameters are available only when the response pagination feature is enabled:

- Common request parameters
  - pageNum: the current page number.
  - Pagesize: The page size, that is, the number of records per page.
- Common response parameters
  - pageNum: the current page number.
  - Pagesize: The page size, that is, the number of records per page.
  - totalNum: the total number of records.

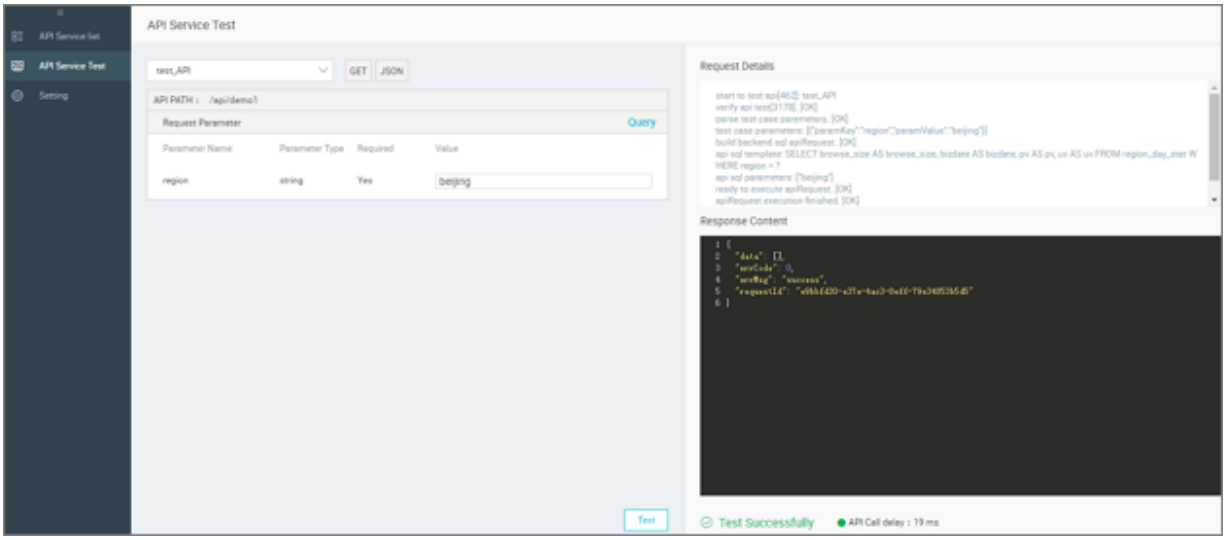
 Note:

- The request parameter only supports the equivalent query, and the return parameter only supports the output of the field value as is.
- As far as possible, set an indexed field to a request parameter.
- You are allowed to specify no request parameters for an API. In that case, the pagination feature must be enabled.
- To make it easy for API callers to understand the details of an API, we recommend that you specify the sample value, default value, and description parameters of the API.
- Click on the configured API to view a list of the APIs that have been generated in the current table, avoid generating the same API.

When the configuration of the API parameters is complete, click Next to enter the API testing section.

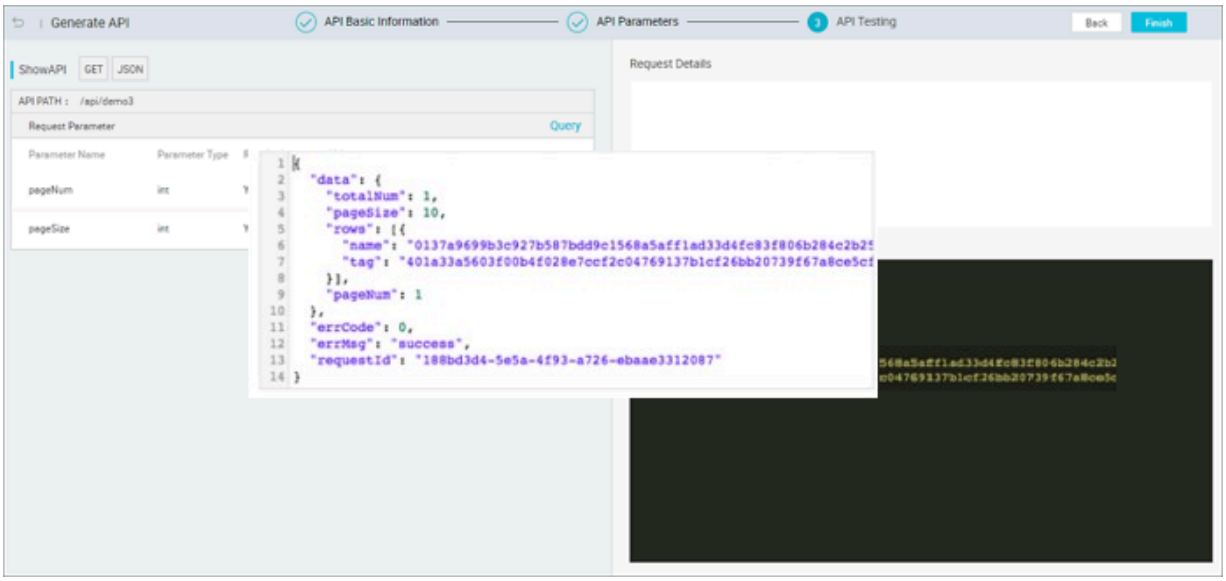
API Testing

After completing configuration of API parameters, you can start the API test.



Set parameters and click Start Test to send the API request online. The API request details and response are displayed on the right. If the test fails, read the error message carefully and make the appropriate adjustments to test your API again.

You need to note the settings for the normal return example during the configuration process. When testing an API, the system automatically generates exception examples and error codes. However, normal response examples are not automatically generated. After the test succeeds, you need to click Save as Normal Response Sample to save the current test result as the normal response sample. If sensitive data is included in the response, you can manually edit it.



Note:

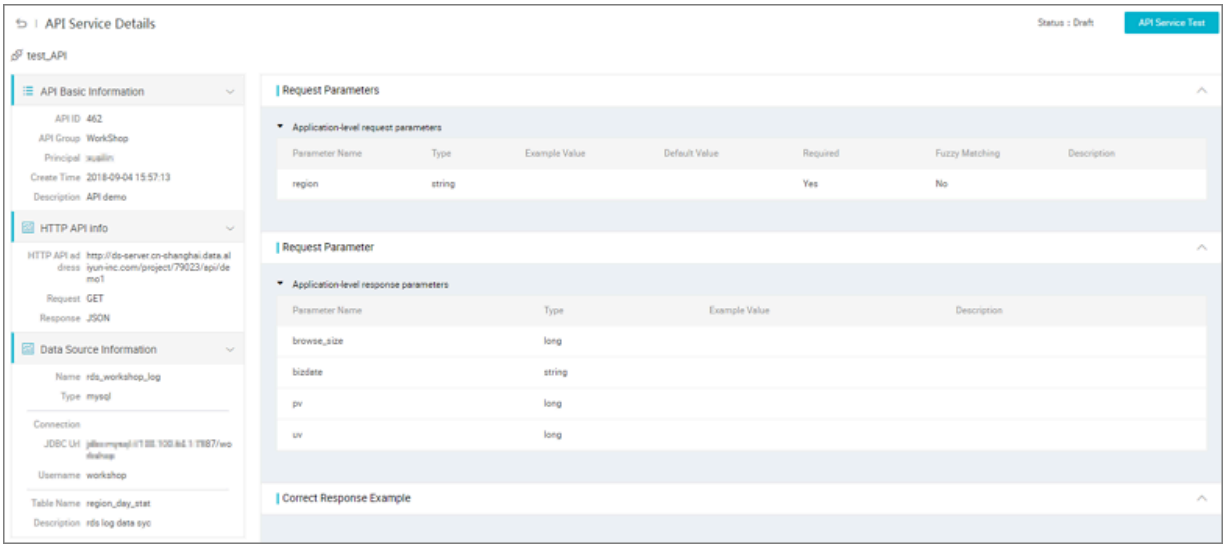
- Normal response examples provide an important reference value for the API callers. Specify an example if possible.

- The API calling delay is the delay of the current API request, which is used to evaluate the API performance. If the latency is too high, you may consider optimizing your database.

After completing the API test, click Finish. The data API is successfully created.

API details viewing

Back on the API service list page, click details in the Action column to view the details of the API. This page displays detailed information about an API from the view of a caller.



2.3.4 Generate API in Script Mode

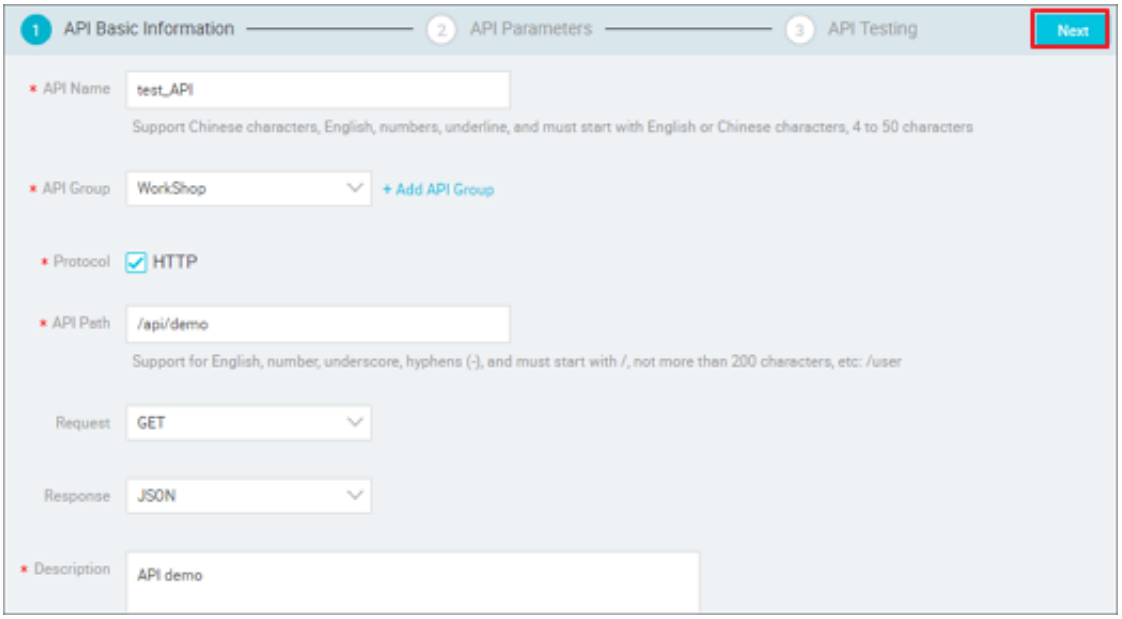
This article introduces you to the steps that script mode can take to generate the API.

To meet the needs of high-end users for personalized queries, the Data Service also provides a script pattern for customizing SQL, allows you to write your own SQL queries for the API, multi-Table Association, complex query conditions and Aggregate functions are supported.

Configure the API basic information

1. Navigate to the API Service list > Generate API.

2. Click Script Mode to fill in the API basics.



Note the settings for the API grouping during configuration. An API group includes a collection of APIs that are used for a specific scenario. It is the minimum management unit in API Gateway. In the Alibaba Cloud API Marketplace, each API group corresponds to a specific API product.

> Note:
> The set up example for API grouping is as follows:
>
> For example, you would like to configure an API product for weather inquiry, weather search API by city name weather search API, scenic spot name search weather API and zip search weather API three kinds of APIS, then you can create an API group called a weather query, and put the above three APIs in this group. The API is shown as a weather query product when published to the marketplace.
>
> Of course, if your generated API is used in your own app, you can use grouping as a classification.

Currently, the build API only supports HTTP protocol, GET request mode, and JSON return type.

3. After providing the API basic information, click Next to go to the API parameter configuration page.

Configure the API Parameters

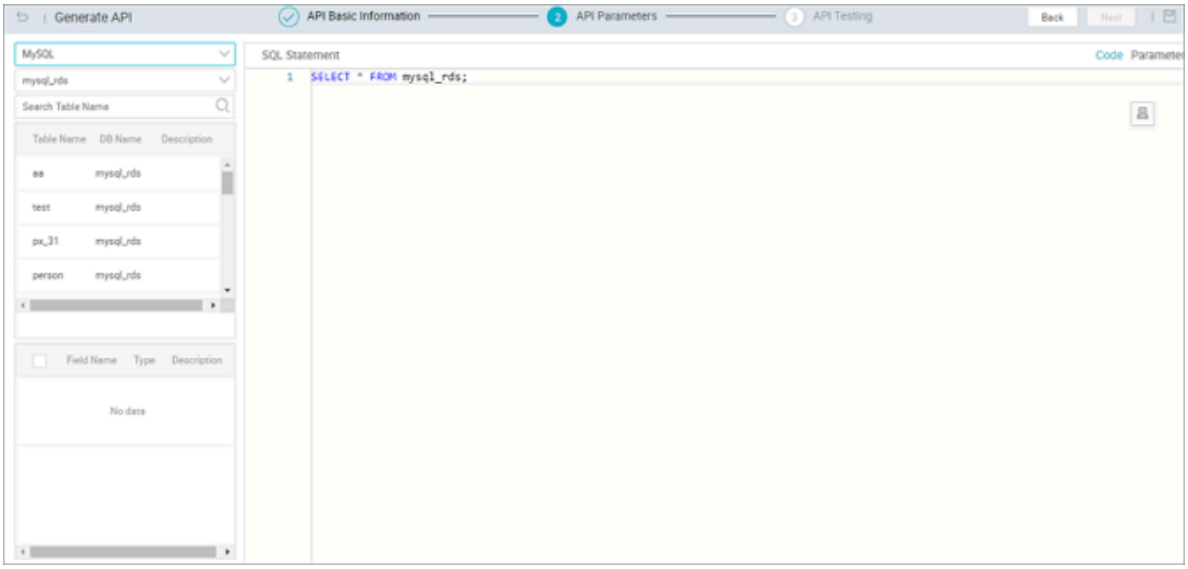
1. Select the data source and table.

   Navigate to the data source type > data source name > data table, click the appropriate table name in the data table list, you can view the field information for this table.

   > Note:
   > - You need to configure the data source in advance in the data set formation.
   > - You must select a data source. Table join queries across data sources are not supported.

2. Write SQL queries for the API.

   You can enter the SQL code in the code box on the right side. The system supports one-click SQL function, checking fields in the list of fields, and clicking Generate SQL, the SQL statement for `SELECT xxx FROM xxx` is automatically generated and inserted at the right cursor.

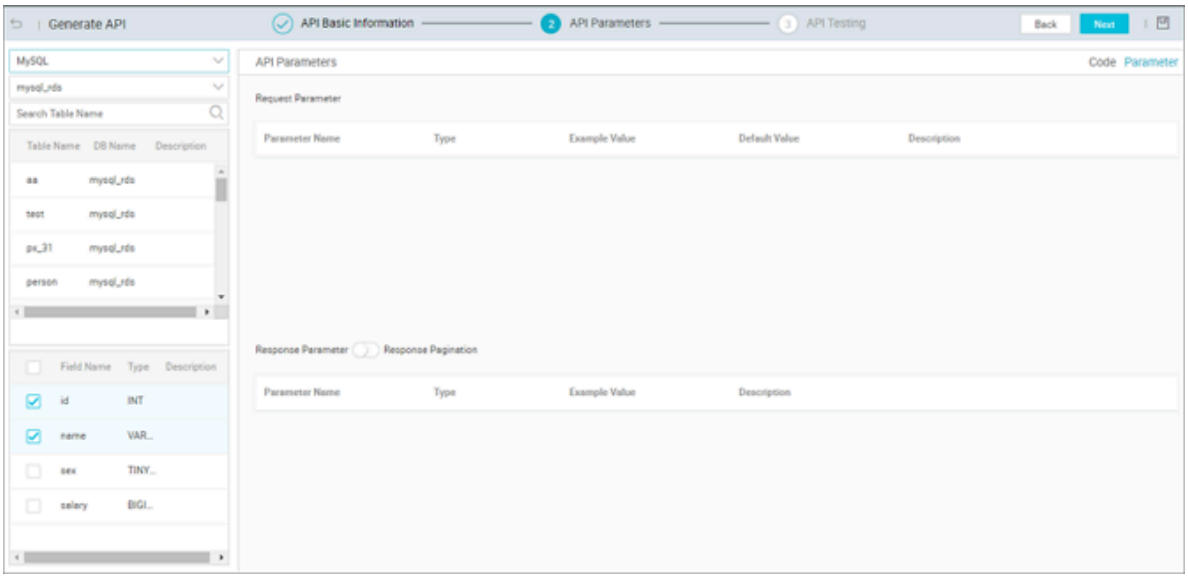


   > Note:
   > - One-click SQL addition is especially useful when the number of fields is relatively large, which can greatly improve the efficiency of SQL writing.
   > - The field of the SELECT query is the return parameter of the API, the parameter at the where condition is the request parameter for the API, And the request parameter is identified with $.

3. Finally, edit and complete parameter information.

   After writing the API query SQL, click the parameters in the upper-right corner to switch to the parameter information Edit page, you can edit the type, sample values, default values, and descriptions of the parameters here, where Type and description are required.

   Note:
   To help the caller of the API get a more comprehensive understanding of the API, please complete the API parameter information as much as possible.



You need to pay attention to the settings that return result paging during the configuration process.

- If you do not enable the response pagination, the API outputs up to 500 records by default.
- If the return result may exceed 500, turn on the response pagination function.

The following public parameters are available only when the response pagination feature is enabled:

- Common request parameters
  - pageNum: the current page number.
  - Pagesize: The page size, that is, the number of records per page.

- Common response parameters
  - pageNum: the current page number.
  - pageSize: The page size, that is, the number of records per page.
  - totalNum: the total number of records.

 Note:

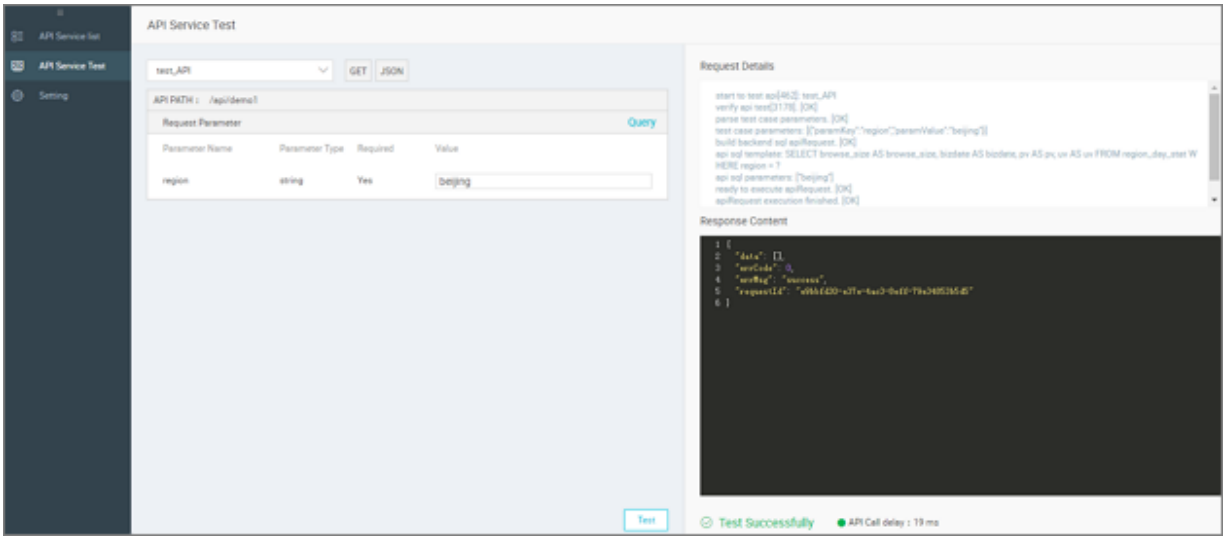
SQL rule prompt.

- Only one SQL statement is supported, and multiple SQL statements are not supported.
- Only the `SELECT` clause is supported. Other clauses such as `INSERT`, `UPDATE `, and `DELETE` are not supported.
- The query field for select is the return parameter for the API, the variable Param in the $ {Param} in the where condition is a request parameter for the API.
- `SELECT \*` is not supported, columns of the query must be specified explicitly.
- Single table queries, table join queries, and nested queries within one data source are supported.
- If the column name of the SELECT query column has a table name prefix (such as T. name), the alias must be taken as the return parameter name (such as T. name as name).
- If you use the aggregate function (min/max/sum/count, etc), the alias must be taken as the return parameter name (such as sum (Num) as total \ _ num).
- In SQL, $ {Param} is uniform when the request parameter is replaced, contains $ {Param} in the string }. When $ {Param} has an escape character \, it does not do request parameter processing, processed as an ordinary string.
- Putting $ {Param} in quotation marks is not supported, such as '$ {ID}', 'ABC $ {xyz} 123 ', concat ('abc ', $ {xyz}, '123') can be passed if necessary ') implementation.

When the configuration of the API parameters is complete, click Next to enter the API testing section.

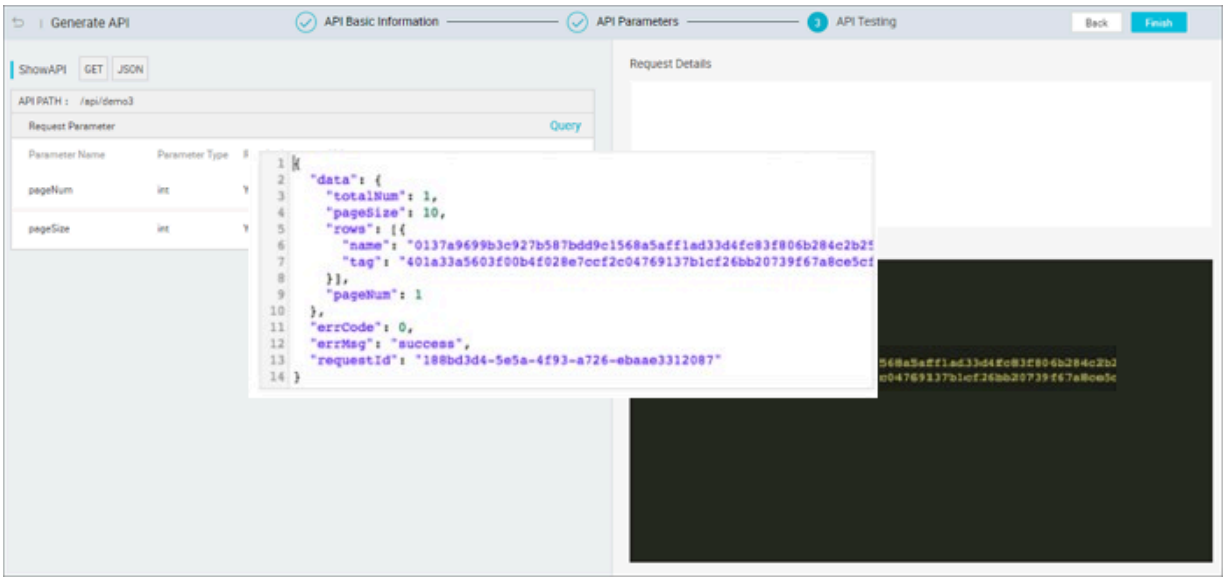
API Testing

After completing configuration of API parameters, you can start the API test.



Set parameters and click Start Test to send the API request online. The API request details and response are displayed on the right. If the test fails, read the error message carefully and make the appropriate adjustments to test your API again.

You need to note the settings for the normal return example during the configuration process. When testing an API, the system automatically generates exception examples and error codes. However, normal response examples are not automatically generated. After the test succeeds, you need to click Save as Normal Response Sample to save the current test result as the normal response sample. If sensitive data is included in the response, you can manually edit it.



Note:

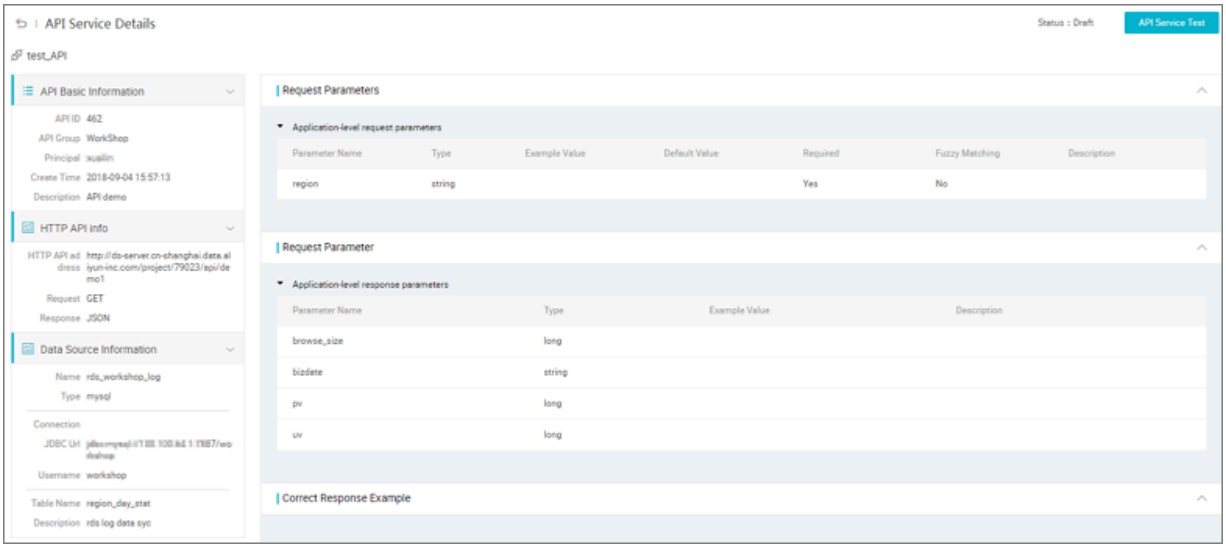
- Normal response examples provide an important reference value for the API callers. Specify an example if possible.

- The API calling delay is the delay of the current API request, which is used to evaluate the API performance. If the latency is too high, you may consider optimizing your database.

After completing the API test, click Finish. The data API is successfully created.

API details viewing

Back on the API service list page, click details in the Action column to view the details of the API. This page displays detailed information about an API from the view of a caller.



2.4 Register API

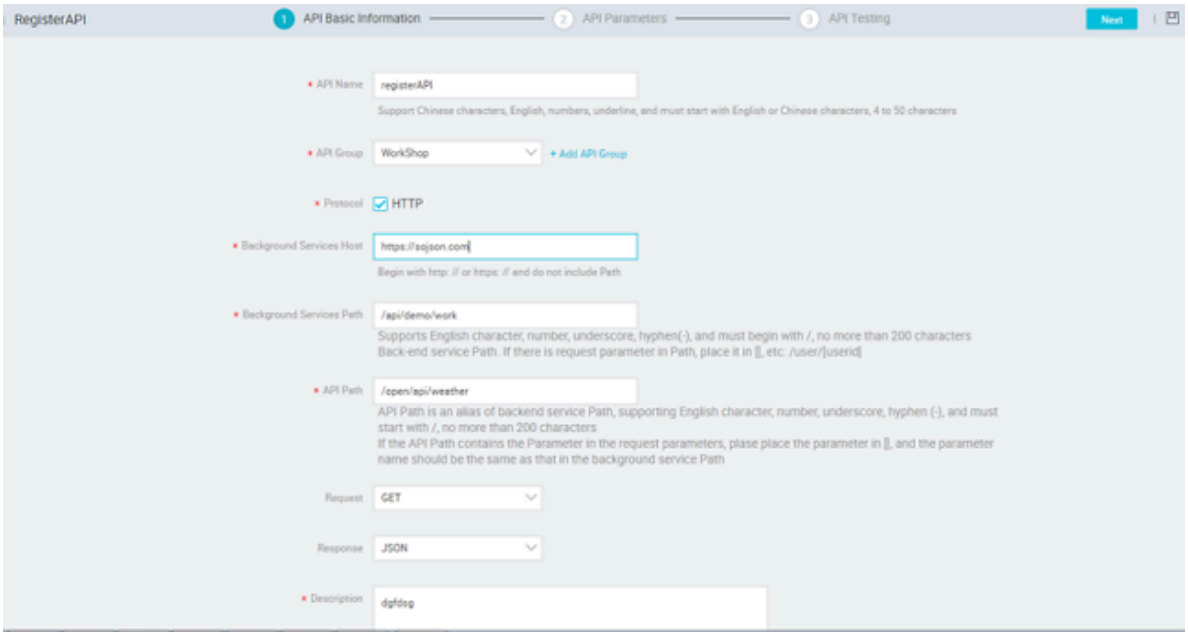
This section describes how to register an API.

You can register currently available APIs in Data Service. These APIs can be managed and published to API Gateway together with APIs created based on data tables. Currently, you can only register RESTful APIs supporting GET, POST, PUT, and DELETE requests and content types form,JSON,and XML.

Configure the API basic information

1. You can go to the registration API page by selecting the Register API in the API Service list.

2. Configure the API basic information.



Parameters:

- Protocol: Only HTTP is supported.
- Backbround Service Host: Enter the host of the API. The host must start with http:// or https://, and cannot contain the path.
- Backbround Service Path: Enter the path of the API. Put parameter names in brackets, for example, /user/[userid].

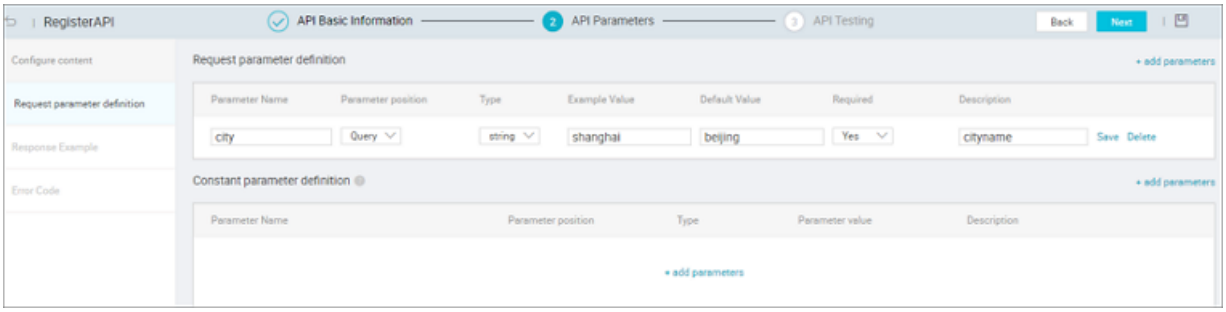
  If a parameter is defined in the path, the system automatically adds the parameter in the path to the request parameter list in the second step of the API registration wizard.
- API path: The alias of the background service path. It allows an API for the background service host and path to register as multiple APIs.

  Parameters defined in Backbround Service Path must also be defined in brackets in API Path.
- Request method: The options include GET, POST, PUT, and DELETE. Different request methods correspond to different subsequent configuration items.
- Return Type: Select JSON or XML.

3. After providing the API basic information, click Next to go to the API parameter configuration page.

Configure API parameters

After configuring the basic API information, you can configure the API parameters. including the request parameters, response example, and error code of the API.



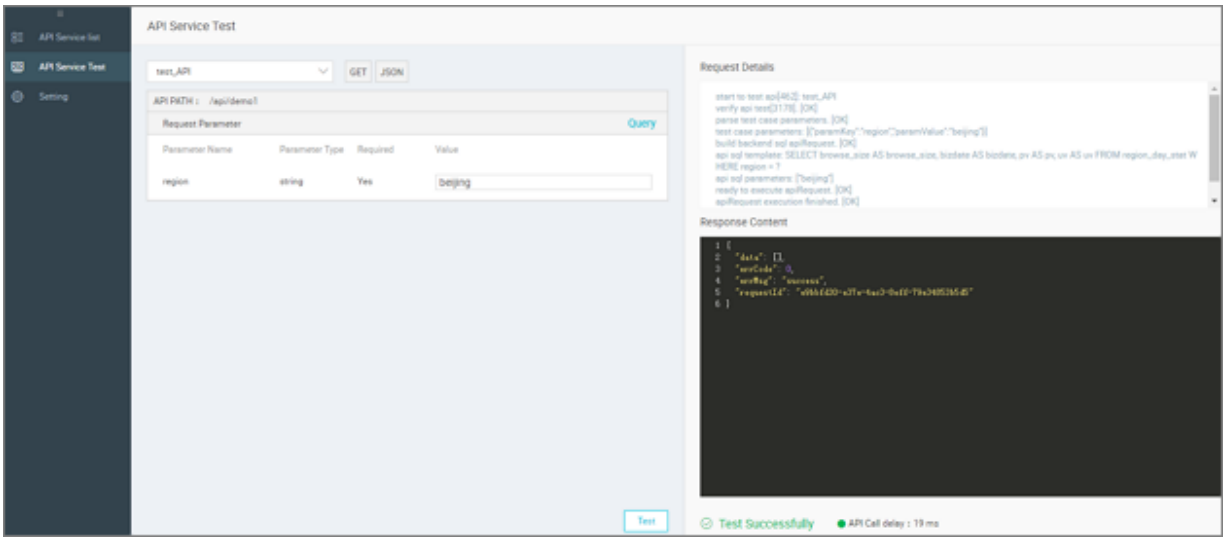
- Request Parameters:
  - Parameter location: The options include Path, Header, Query, and Body. Different request methods support different optional parameter locations. You can select the options as required.
  - Constant parameters: The parameters that have the fixed values and are invisible to callers. The constant parameters do not need to be input during API calling. However, the background service always receives the defined constant parameters and their values. Constant parameters are applicable if you want to fix the value of a parameter or hide the parameters to the callers.
- Request Body is required only when the request mode is POST or PUT. You can enter the desc The content types of the request body include JSON and XML.

  Note:
  If the request body is defined in the request body definition and the body location parameter is defined in the request parameter definition, the body location parameter is invalid. The request body is applied.
- You can enter a normal example or an exception example for API callers to refer to when writing the return parse code.
- Enter the common errors and solutions in API calling. This enables API callers to troubleshoot and solve these errors.

  Note:
  To ensure that the API is easily used by the callers, provide complete API parameter information if possible, especially the parameter sample values, default values, and response examples.

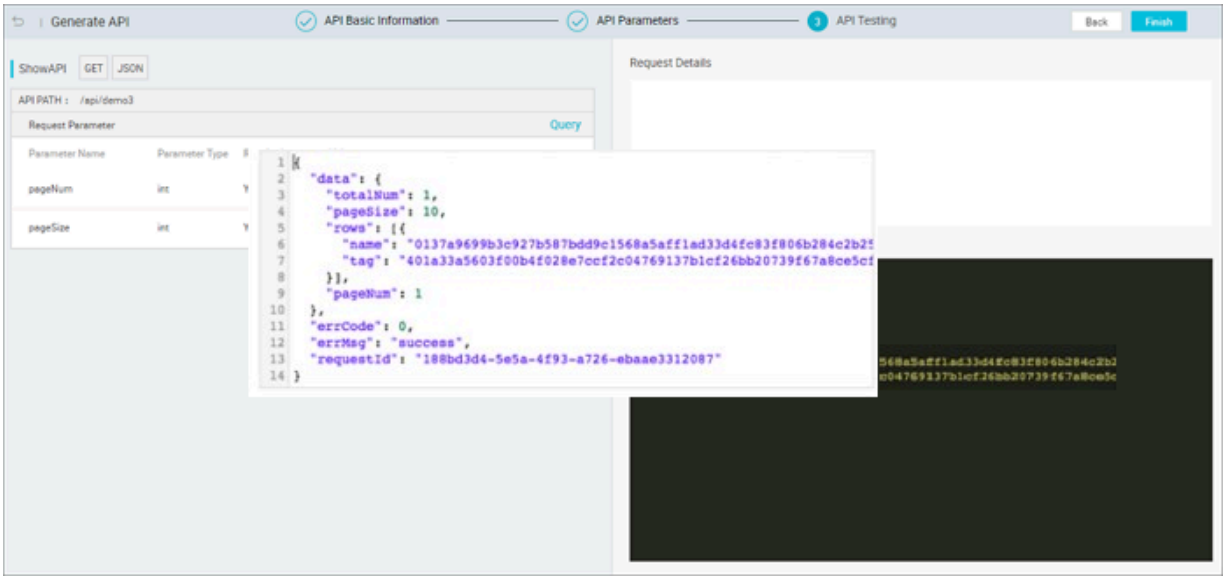
API Testing

After completing configuration of API parameters, you can start the API test.



Set parameters and click Start Test to send the API request online. The API request details and response are displayed on the right. If the test fails, read the error message carefully and make the appropriate adjustments to test your API again.

You need to note the settings for the normal return example during the configuration process. When testing an API, the system automatically generates exception examples and error codes. However, normal response examples are not automatically generated. After the test succeeds, you need to click Save as Normal Response Sample to save the current test result as the normal response sample. If sensitive data is included in the response, you can manually edit it.



Note:

- Normal response examples provide an important reference value for the API callers. Specify an example if possible.
- The API calling delay is the delay of the current API request, which is used to evaluate the API performance. If the latency is too high, you may consider optimizing your database.

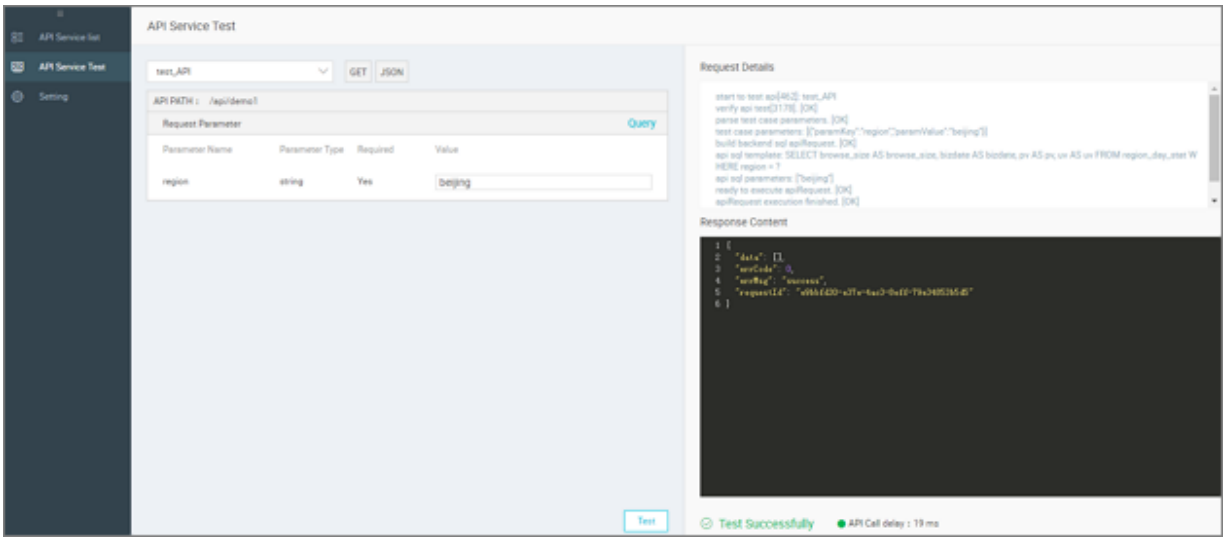
After completing the API test, click Finish. The data API is successfully created.

2.5 API service test

This article will show you how to test your API.

When creating and registering an API, you can test the API. For more information, see #unique_144.

The system also provides an independent API service test function for you to perform routine API tests online. You can choose More > Test in the Actions column of the API list to go to the API test page. Alternatively, you can click API Service Test in the left-side navigation pane, enter the API test page, and select the corresponding API.



Note:

The API service test page provides only the API online test function and does not allow update and storage of the API normal response examples. To update an API normal response example, click Edit in the API list, enter the API editing mode, and update the content of the normal response example in the API test process.

2.6 Publish an API

API Gateway is an API hosting service that provides full life cycle management covering API release, management, O&M, and sales. It provides you with a simple, fast, low-cost, and low-risk method to implement microservice aggregation, frontend-backend isolation, and system integration, and opens functions and data to partners and developers.

API Gateway provides permission management, traffic control, access control, and metering services. The service makes it easy for you to create, monitor, and secure APIs. Therefore, we recommend that you publish the APIs that have been created and registered in Data Service to API Gateway. Data Service and API Gateway are connected, which allows you to publish APIs to API Gateway easily.

Publish APIs to API Gateway

Note:

To release an API, you must first activate the API Gateway service.

After activating API Gateway, you can click Publish in the Actions column of the API service list to release the API to API Gateway. The system automatically registers the API to API Gateway during the publish process. The system creates a group in API Gateway with the same name as the API group and releases the API to the group.

After the release, you can go to the API Gateway console to view the API information. You can also set the throttling and access control functions in API Gateway.

If your API needs to be called by your application, you must create an application in API Gateway, authorize the API to the application, and encrypt the signature call using the AppKey and AppSecret. For more information, see API Gateway help documentation. At the same time, the API gateway also provides the SDK in the mainstream programming language, you can quickly integrate your API into your own applications, for more information, please refer to the SDK download and user's guide.

Publish APIs to Alibaba Cloud API Marketplace

After your APIs from Data Service have been published to API Gateway, you can then publish them to Alibaba Cloud API Marketplace. This is an easy way to achieve financial gains for your company.

Before selling the API to the Ali cloud API market, first of all, it is necessary to enter the Ali cloud market as a service provider.

Note:
Select to enter API Marketplace as shown in the following figure. Note: only enterprise users are allowed to enter Alibaba Cloud API Marketplace.

Procedure

1. Enter the Ali cloud service provider platform.
2. Click commodity management > publish the merchandise and select the access type as the API service.
3. Select the API grouping that you want to list (one grouping corresponds to one API commodity).
4. Configure commodity information and submit audit.

Once your product has been successfully published to Alibaba Cloud API Marketplace , users can purchase it worldwide.

2.7 Delete API

Choose More > Delete in the Actions column of the API service list to delete an API.

Note:

- An API can be deleted only when it is in offline status. If it is online, deprecate the API and then delete it.
- The delete operation is irreversible. Delete an API with caution.

2.8 Call an API

This section describes how to call an API after this API is released on API Gateway.

API Gateway provides API authorization and the SDK for calling APIs. You can authorize yourself, your associates, or third parties to use APIs. If you want to call an API, perform the following operations.

Three elements for calling an API

To call an API, you need the following three elements:

- API: the API that you are about to call, which is clearly defined by the API parameters.
- app: Identity that you use to call the API. The AppKey and AppSecret are provided to authenticate your identity.
- Permission relationship between the API and app: When an app needs to call an API, the app must have the permission of this API. This permission is granted through authorization.

Procedure

1. Get the API documentation

   The acquisition method varies according to the channel that you use to obtain the API. It is generally divided into API services purchased from the data market and not required to purchase, two ways are actively authorized by the provider. For more information, see get API documentation.

2. Create a project

   The app is the identity that you use to call an API. Each app has a set of AppKey and AppSecret, which are equivalent to an account and a password. For more information, see creating an application.

3. Get the permission

   Authorization means granting an app the permission to call an API. Your app must be authorized first to call an API.

   The authorization method varies according to the channel that you use to obtain the API. For more information, see obtaining authorization.

4. Call API

   You can directly use the multi-language call sample provided by API Gateway Console, or use a self-compiled HTTP or HTTPS request to call the API. For more information, see calling the API.

2.9 FAQ

- Q: Do I have to activate the API gateway?

  A: API Gateway provides the API hosting service. If you plan to open your APIs to other users, the API Gateway service must be activated first.

- Q: Where can I configure the data sources?

  A: To create a data source, select DataWorks > Data Integration > Data Sources . After the configuration, Data Service automatically reads the data source information.

- Q: What is the difference between a wizard-created API and a script-created API?

  A: The script mode provides more powerful functions. For more information, see #unique_149.

- Q: What is an API group in Data Service? Is it the same as an API group in API Gateway?

  A: An API group contains several APIs in a certain scenario. It is the minimum unit. In a word, the two are equivalent. When you publish an API group from Data Service to API Gateway, the gateway automatically creates an API group with the same name.

- Q: How can I configure an API group appropriately?

  A: Typically, an API group includes APIs that provide similar functions or solve a specific issue. For example, the API for querying weather by city name and the API for querying weather by latitude and longitude can be put into an API group named "weather query".

- Q: How many API groups can be created?

  A: An Alibaba Cloud acocunt can create up to 100 API groups.

- Q: In what situations do I have to enable API response output pagination?

  A: By default, an API outputs up to 500 records. To output more records, enable API response output pagination. When no API request parameters have been set, the API may output a large number of records, and the API response output pagination is automatically enabled.

- Q: Do APIs created by Data Source support POST requests?

  A: Currently, a created API supports only the GET request.

- **Q: Does Data Service support HTTP?**

  A: Currently, Data Service does not support HTTP. HTTP may be supported in later versions.

3 Function Studio

3.1 Overview

Function Studio is a web project coding and development tool independently developed by the Alibaba Group for function development scenarios. It is an important component of DataWorks.

Based on an innovative underlying architecture, Function Studio occupies few resources, supports high concurrency, and is convenient, flexible, and efficient.

Function Studio provides functions such as syntax highlighting, automatic code completion, intelligent error correction, and syntax error hinting. It also supports online development and debugging, collaborative coding, and publishing of UDF resources and functions to DataWorks with one click.

Features

- Function Studio allows you to edit MaxCompute Java user-defined functions (UDFs) and to compile and publish them to DataWorks with one click.
- Function Studio allows you to manage the files and folders of projects.
- Function Studio provides a context-based intelligent editor that allows you to intelligently edit multiple Java files concurrently. Function Studio also supports searching for definitions and references, code hinting and completion, highlighting of keywords in the syntax, and real-time syntax error hinting.
- Function Studio supports UDF, user defined aggregate function (UDAF), and user-defined table-generating function (UDTF) templates, and automatically publishes resources and functions to the workflows in DataWorks, greatly improving the UDF development efficiency.
- Function Studio allows you to perform common Git operations such as commit and push in the DataWorks development environment, supporting version control of code files.
- Function Studio allows you to redirect DataStudio to Function Studio with one click to view the source codes of UDFs, facilitating the online maintenance and management of UDFs.

Future versions

Function Studio will support more languages, such as Python, and more platform-based function development scenarios, such as real-time computing.

3.2 Releases

This topic describes the releases of Function Studio to inform you about the new features and syntax characteristics of Function Studio, improving the efficiency of project development.

Function Studio 1.0

Released on: December 11, 2018

- An IDE that supports the online development of Java UDFs.
- UDF development with one click, including compilation and publishing of UDF resources and functions.
- Maintenance or secondary development of published functions or resources in Function Studio.
- Advanced editing functions of Java, such as code hinting, redirection, and refactoring.
- Git features.
- Online debugging and hot code replacement in run or debug mode.

3.3 Get started

3.3.1 Create projects

This topic describes how to create a project in Function Studio.

You can click Create Project or Import Project from Git to create a project.

1. In the top navigation bar, choose Project > Create Project, or choose Project > Import Project from Git.
   - If you select Create Project, in the Create Project dialog box, set the project name, project description, and project template.

      Note:

Currently, Function Studio supports Java and Python.

- If you select Import Project from Git, in the Import Project from Git dialog box, set the Git repo URL, project name, project description, and project template.

  Note:
  Function Studio allows you to directly import a project from Git. Only HTTP projects are supported. You must convert SSH projects to HTTP projects.

2. Click OK.

3.3.2 Develop UDFs

After a project is created, the framework code is automatically generated, based on which you can create Java UDFs, UDAFs, and UDTFs. This topic takes creating a UDF as an example to describe how to develop UDFs, UDAFs, and UDTFs.

1. Choose Add > UDF.
2. In the dialog box that appears, enter the class name and click OK. The framework code is automatically generated.
3. Modify the variables in the `evaluate` method as needed.

3.3.3 Debug UDFs

You can debug UDFs (Java only), UDAFs, and UDTFs.

Debug UDFs (Java only)

1. Create a main method.

   Currently, Function Studio only allows you to call and debug UDFs online by using the main method.

2. Set the debugging configuration.

   Click Run/Debug Config in the upper-right corner.

   Select the main method you just created. Other information is automatically generated.

Parameter	Description
Main class	Required. The main method you want to debug.
VM options	Optional. The JVM startup parameter.
Program Variables	Optional. The startup variables.
JRE	Currently, only JDK1.8 is supported.

Parameter	Description
PORT	The HTTP port you must open. This parameter is optional for UDF, UDAF, and UDTF projects.
ECS instance	The instance type.
Enable Hot Code Replacement	Indicates whether to enable hot code replacement.

3. Enable debugging.

   In the Lower.java file, set a breakpoint for the evaluate method and click Debug.

   After debugging is enabled, you can debug the method. You can click `Step In` to step into the UDF and view the variables.

Debug UDAFs

To debug UDAFs, you must manually construct the relevant data and use a warehouse to simulate the MaxCompute table. The schema and data of the table are saved in the warehouse, which can be used to compile the main method for testing.

After the warehouse is initialized, call relevant UDAFs for testing.

Debug UDTFs

You can debug UDTFs in the same way as you debug UDAFs. The initialized project already provides the UDTF test class. You can directly run the class to simulate the data and debug UDTFs.

Click Run. If the application throws no error, the UDTF test is passed.

3.3.4 Publish UDFs

This topic describes how to submit resources or functions to the DataWorks development environment.

Submit resources to the DataWorks development environment

1. In the MaxCompute project section, click the Publish icon in the upper-right corner.
2. Click Submit Resource to Development Environment.
3. In the dialog box that appears, set the parameters, and then click OK. After the resource is published, the link of the resource in the DataWorks appears.
4. Open the link to locate the published resource.

Submit functions to the DataWorks development environment

1. In the MaxCompute project section, click the Publish icon in the upper-right corner.
2. Click Submit Function to Development Environment.
3. In the dialog box that appears, set parameters, and then click OK. After the function is published, the link of the function in the DataWorks appears.
4. Open the link to go to the function details page. You can edit the function in Function Studio.

> Note:
> The resources and functions you published are in the development environment. To use them online, you must publish them to an online environment through the Publish Task page.

3.3.5 Develop MapReduce projects

After a project is created, the framework code is automatically generated for the project, based on which you can create MapReduce tasks. This topic takes the WordCount sample code as an example to describe how to perform the test and publish the project from the very beginning.

Create projects

1. In the top navigation bar, choose Project > Create Project.
2. In the Create Project dialog box, set parameters.

   In the specified MaxCompute workspace cdo_datax, create a project named wordcountDemo, and select UDFJava Project as the project template.
3. Click OK.

Develop projects

The mapred package comes with the MapReduce sample code of WordCount. The sample code is used to count words in an input table and write the statistical result to an output table. The input table and output table are different tables. For more information, see MapReduce.

Debug projects

Currently, you cannot debug MapReduce projects in Function Studio. To debug a MapReduce project, you must publish the code to the DataWorks development environment, and then verify the logic in DataWorks.

> Note:
> Currently, Function Studio only allows you to write, compile, and package code.

Publish projects

1. Function Studio allows you to compile and package the code and publish it to the DataWorks development environment.
   a. Click the Publish icon and then select Submit Resource to Development Environment.
   b. In the Submit Resource to Development Environment dialog box, set parameters.

Parameter	Description
Target Workspace	The target workspace in which you publish the JAR package. The target workspace must be the same as the workspace where you create the DataWorks compute node of the ODPS MR type in step 2. In this example, the target workspace is cdo_datax.
Target Workflow	The target workflow.
Resource	You can specify the resource name, which is referenced in the subsequent compute node scripts.
Force Overwrite	The project name can be the same as the name used in the previous publish. If you select Force Overwrite, the new name is used.

   c. Click OK. The code is published to the DataWorks development environment.

      A message appears, indicating whether the code is published successfully.

2. Create a compute node of the ODPS MR type in DataWorks for testing.
   a. Open the DataWorks workspace named cdo_datax, and create a compute node of the ODPS MR type.
   b. The information in the red box of the following figure must be added to the script of the compute node. Currently, you must manually replace some variables in the script with those in the JAR package.

      Note:
      Replace the following variables in the script with those in the JAR package that you published in Function Studio and generate the final code:

      - `jar -resources`: Replace it with the name of the JAR package that you published in Function Studio.
      - `- classpath` : Replace it with the path of the JAR package in DataWorks.
      - Separate the parameters of the main method of a class by space.

   c. Click the target workflow, and select Resource to view the information of the JAR package that you published in Function Studio and replace relevant information in the script with that in the JAR package.

      - The name of the JAR package is `WordCountD emo_1 . 0 . 0 . jar ,` corresponding to `- resource` in the script.
      - Right-click the JAR package and choose View Change History. The path of the JAR package is `http :// schedule @{ env } inside . cheetah . alibaba - inc . com / scheduler / res ? id = 106342493 ,` corresponding to `- classpath` in the script.

      The final script:

```
# Manually replace relevant informatio n in the
 script with that in the JAR package . The final
 script is generated .
 jar - resources WordCountD emo_1 . 0 . 0 . jar
- classpath http :// schedule @{ env } inside . cheetah .
 alibaba - inc . com / scheduler / res ? id = 106342493
```

```
com . alibaba . dataworks . mapred . WordCount   wordcount_
demo_input   wordcount_  demo_outpu  t
```

d. Create a test table and prepare test data.

After the data is prepared, run the script in the development environment.

Now, the test of WordCount in the development environment has been completed. The compute node, JAR package, and input and output tables of WordCount are all in the development environment. Therefore, you need to publish them to the production environment.

3. Publish the resource package, data tables, and nodes to the production environment of DataWorks.

a. Commit the code of the compute node.

b. Configure the publish items.

c. On Publish Task page, select the JAR package and node that you want to publish and click Publish in the Actions column.

d. Publish the tables.

e. On the Operation Center tab page, perform testing for the MapReduce project online.

The log indicates that the project has run successfully.

Function Studio allows you to write, compile, and publish the code to a DataWorks compute node. You must manually generate a compute node in DataWorks, and run the compute node in the development environment and production environment separately.

3.3.6 Perform Git operations

You can associate a new project to a Git repo and then perform common Git operations on the project.

1. In the top navigation bar, choose Version > Version Control.

- Locate the row that contains the target file, and then click + next to the file to add the file to the Git repo.
- Click √ to commit and push the file.
- Click ... to pull or push the file.

2. Click master at the bottom to associate a remote branch with a local branch. You can also create a branch.

3.3.7 Collaboratively edit the same code file

Function Studio allows multiple users to edit the same file of the same project collaboratively.

You can click Share in the upper-right corner and invite other users to edit the file collaboratively.

You can click Shared from Others to view the list of shared projects.

The following figure shows a scenario where multiple users edit the same file of the same project.

3.3.8 Perform unit testing

Currently, Function Studio supports the unit testing (UT) feature, including detecting the UT runner, running the UT code, and displaying the running results.

Detect the UT runner

> Note:
>
> - The UT class files must be stored in the `src / test / java` directory. A Java UT class file that is not stored in this directory cannot be identified as the Java UT class.
> - After the Java UT class file is created, add the `@ Test` annotation of `org . junit . Test` to the test case.

Run the UT code

Click Run test to run the UT code.

3.3.9 Search a project's code by keyword

This topic describes how to search a project's code by keyword in Function Studio.

Function Studio allows you to search the code of a project by keyword.

3.3.10 Automatically generate code

Currently, Function Studio supports most of the code generation features available in Java, including generating a constructor, getter, or setter method, overriding methods that a subclass inherits from a superclass, and implementing methods of an interface.

Portal

Currently, you can generate the Java code in either of the following two ways:

- Right-click the blank area and choose Generate.
- Press cmd+m.

Constructor

1. On the Generate Code panel, click Constructor.
2. Select the fields for the constructor. The constructor that contains the statement to initialize such fields is generated.

Getter and setter

You can generate the code of the getter and setter methods in the same way as you generate the constructor.

> Note:
> If a Java class does not have any field or the Java class is overridden by the @data annotation of Lombok, the getter or setter function is not required for the Java class. In this case, the Getter, Setter, and Getter And Setter options do not appear on the Generate panel.

Override methods

On the Generate panel, click Override Methods. All methods that can be overridden are listed on the Generate Code panel.

Select a method. The code for overriding this method is generated.

Implement methods

You can generate code for implementing methods in a similar way as that described in "Override methods."

> Note:
> When creating a class to implement an interface, each of the methods defined on the interface must have code implementation. Otherwise, the statement has a syntax error and is marked by a red wave line.

In addition to the Implement Methods option on the Generate panel, you can also use code hinting to achieve the same purpose.