Alibaba Cloud
Table Store

Data channels

Issue: 20190422

Legal disclaimer

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 MaxCompute

1.1 Overview

Table Store is a distributed NoSQL data storage service that is built on Alibaba Cloud Apsara distributed system. It uses data partitioning and load balancing techniques to seamlessly scale up data size and access concurrency, providing storage of, and real-time access to, massive structured data.

MaxCompute is a big data computing service that provides a fast and fully hosted PB-level data warehouse solution, allowing you to analyze and process massive data economically and efficiently.

Scenarios

Table Store: Provides professional data-persistent storage service and user-oriented real-time read/write operations with high concurrency and low latency.

MaxCompute: Provides computing services, which are generally used for cleaning, correcting, and calculating data.

Activation

Activate Table Store

1. Go to the *Table Store details page*.
2. Click Buy Now.
3. In the *Table Store console*, create *instances* and *tables*.

Note:

- To use the incremental tunnel, you must activate the *Stream function* for tables. You can select 24 hours for the validity period.
- Table Store supports the reserved CUs and additional CUs. If the reserved read and write CUs are both set to zero during table creation, then the additional read and write CU is used. You can adjust the reserved read/write CUs of each table at any time.
- Table Store offers each registered account 25 GB of free storage per month.

Activate MaxCompute

1. Go to the *MaxCompute details page*.
2. Click Buy now.

> Note:
> Two billing methods are available, which are prepayment by CU cost and Pay-As-You-Go.

Data tunnel

- Real-time

  Direct read and write
- Offline
  - Incremental synchronization to MaxCompute

    Wizard mode
  - Full export to MaxCompute

    Script Mode
  - Full import to Table Store

    Script Mode

1.2 Incremental synchronization (wizard mode)

Data Integration supports data synchronization in wizard mode and script mode. The wizard mode is simpler while the script mode is more flexible.

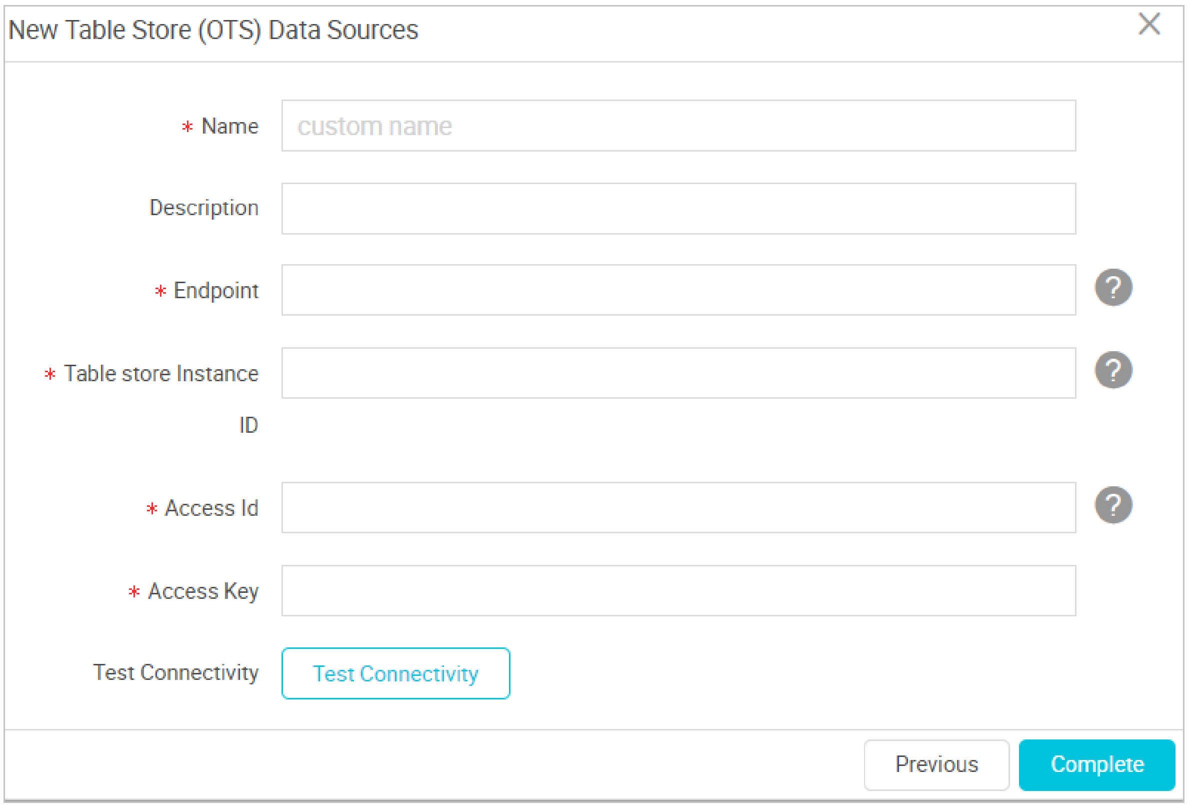
This chapter describes how to synchronize incremental data (generated by the Put, Update, and Delete actions) from Table Store to MaxCompute through the Table Store feature in a near-real-time manner.

> Note:
> Because the offline synchronization mode is used, a latency of about 10 minutes exists.

Step 1. Create Table Store data source

1. Log on to the *Data IDE*.
2. If you are using Data Integration for the first time, you must first *create a Data Integration project*.

3. On the Data Sources page, click New Source.
4. Select Table Store as the data source.
5. Set parameters and click test connectivity.



New Table Store (OTS) Data Sources

* Name	custom name
Description	
* Endpoint	
* Table store Instance ID	
* Access Id	
* Access Key	
Test Connectivity	Test Connectivity

The parameters are described as follows.

Parameter	Description
Name	Name of the Table Store data source. This example uses gps_data.
Description	Description of the data source.

Parameter	Description
Endpoint	Enter the instance address on the Table Store instance page. · If the Table Store instance is in the same region as the MaxCompute instance, enter the private network address. · If the Table Store instance is not in the same region as the MaxCompute instance, enter the public network address. Note: You cannot enter the VPC address.
Table Store ID	Name of the Table Store instance.
Access ID	AccessKeyID of the logon account.
Access Key	AccessKeySecret corresponding to the AccessKeyID of the logon account.

Note:

If the connectivity test fails, check whether the endpoint and instance name are correct. If the problem persists, *open a ticket*.

6. Click complete. Information about the Table Store data source is displayed on the Data Sources page.

Step 2. Create MaxCompute data source

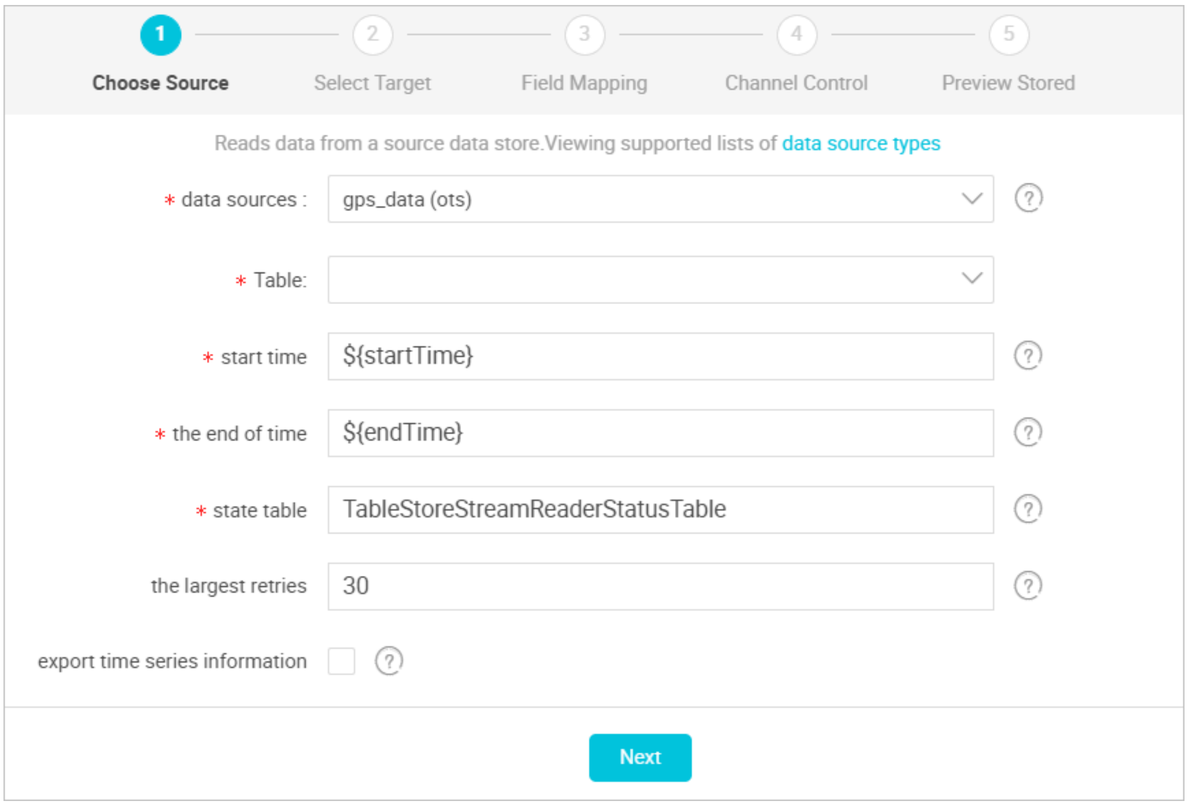
This operation is similar to Step 1. You only need to select MaxCompute as the data source.

In this example, the MaxCompute data source is named OTS2ODPS.

Step 3: Create an incremental real-time data tunnel

1. On the *Data IDE* page, click Sync Tasks.
2. At the right side of the page, click Create a synchronization task.
3. Select Wizard mode.

4. Select the Table Store data source created in Step 1.



Reads data from a source data store.Viewing supported lists of data source types

* data sources : gps_data (ots)

* Table:

* start time ${startTime}

* the end of time ${endTime}

* state table TableStoreStreamReaderStatusTable

the largest retries 30

export time series information ☐

The parameters are described as follows.

Parameter	Description
Data sources	The Table Store data source you created . In this example, gps_data is selected.
Table	Data Integration automatically obtains the latest data table from Table Store. Stream must be activated for the selected table. If Stream is not activated, click Activate Stream in One Click at the right side to activate Stream. The incremental data is valid for up to 24 hours.
Start time	Start time of incremental export. For a periodic task, the variable value is required. The default value is `${ start_time }`.

Parameter	Description
End time	End time of incremental export. For a periodic task, the variable value is required. The default value `is ${ end_time }`.
Status table	It is used to store status values during incremental export. The default value is recommended.
Maximum number of retries	It indicates the maximum number of retries to perform during when the network is unstable. The default value is 30. You can set the value as needed.
Export time series information	It indicates whether the exported data contains the time information. It is not selected by default.

5. On the Select Target page, select the MaxCompute data source created in Step 2.

   The parameters are described as follows.

Parameter	Description
Data sources	The MaxCompute data source you created. In this example, OTS2ODPS is selected.
Table	Select a table in this data source. If no table is available, at the right side click Create New Target Table to create a table. In the dialog box that appears, replace `your_table _name` with the name of the table to be created, for example, ots_gps_data. (Because timestamp is a reserved field in MaxCompute and cannot be used in this box, ts can be used to represent timestamp if necessary.)
Partition information	The default value is `${ bdp . system . bizdate }`, indicating data in MaxCompute is partitioned by date.
Cleaning rule	Select Clean Existing Data Insert Overwrite Before Writing.

6. On the Field Mapping page, make sure the Table Store table maps the MaxCompute table.
7. On the Channel Control page, set the parameters.

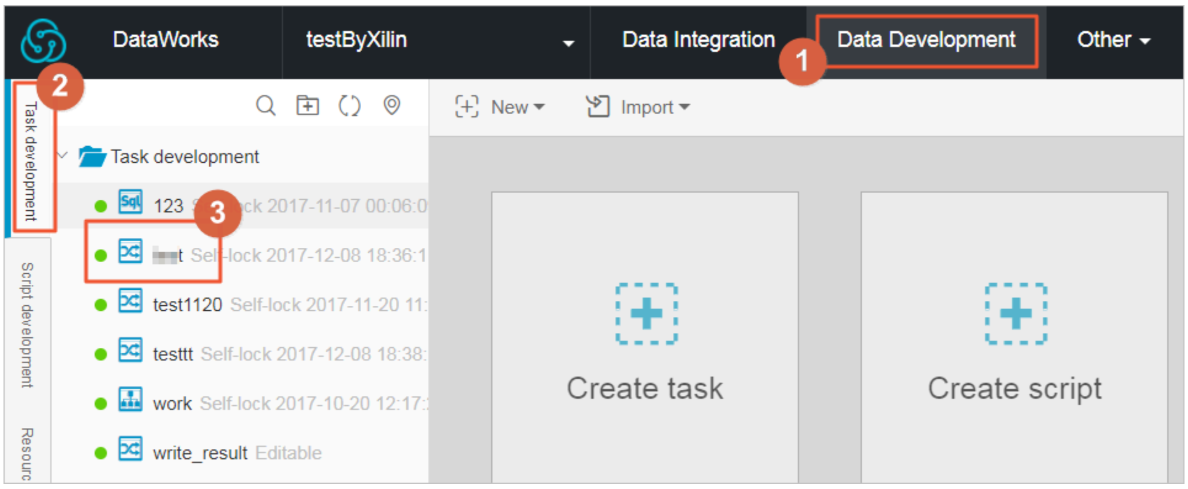
   The parameters are described as follows.

Parameter	Description
Job speed limit	Range: 1 MB/s to 20 MB/S. To request a higher job speed limit, *open a ticket*.
Number of concurrent jobs	The maximum value is 10. Maximum rate of a job = Task speed limit/Number of concurrent jobs
Number of error records	The task fails when the number of error records exceeds the value. The default value is 0.

8. On the preview page, check the configurations.
9. Click Save. In this example, the task name that is saved is OTStoODPS.

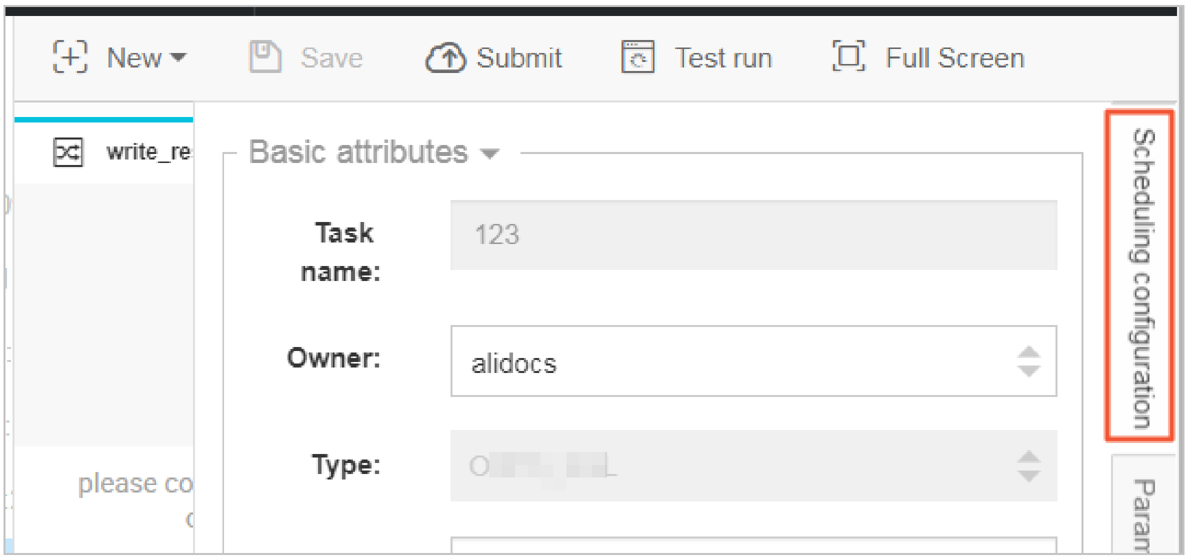
Step 4. Set scheduling parameters

1. At the top of the page, click Data Development.
2. On the Task Development tab, double-click the created task OTStoODPS.



3. Click Scheduling configuration to set the scheduling parameters.

   To set the task to run on the next day, configure the following parameters as shown .



Task name: 123

Owner: alidocs

Type: [redacted]

The parameters are described as follows.

Parameter	Description
Scheduling status	Indicates the running of the task. By default, it is not selected.
Error retry	We recommend that you select this parameter so that the system can retry if an error occurs.
Start date	The default value is recommended.
Scheduling cycle	Minute is used in this example.
Start time	It is set to 00:00 in this example.
Scheduling interval	The scheduling interval is set to 5 minutes in this example.
End time	It is set to 23:59 in this example.
Dependency attributes	Set the Dependency Attribute field based on your business needs, or retain the default value.
Cross-cycle dependency	Set the Cross-Cycle Dependency field based on your business needs, or retain the default value.

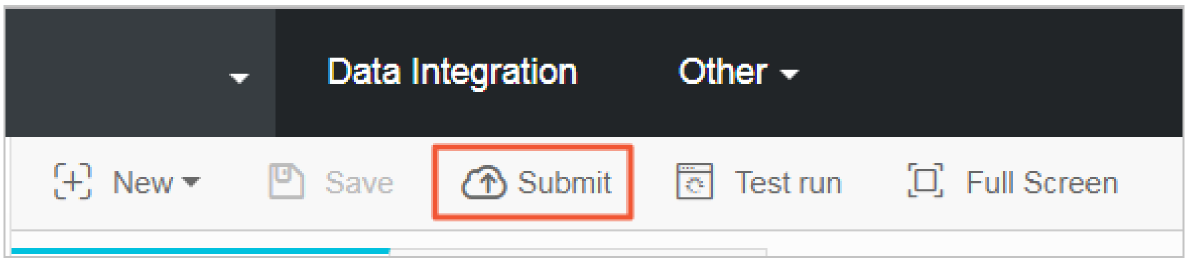
4. Click Parameter Configuration.

   The parameters are described as follows.

Parameter	Description
${bdp.system.bizdate}	It does not need to be configured.
startTime	It is the Start Time variable set in Scheduling Configuration. In this example, it is set to $[yyyymmddhh24miss-10/24/60], indicating a time equal to the scheduling task start time minus 10 minutes.
endTime	It is the End Time variable set in Scheduling Configuration. In this example, it is set to `$[ yyyymmddhh 24miss - 5 / 24 / 60 ]`, indicating a time equal to the scheduling task start time minus 5 minutes.

Step 5. Submit the task

1. At the top of the page, click Submit.

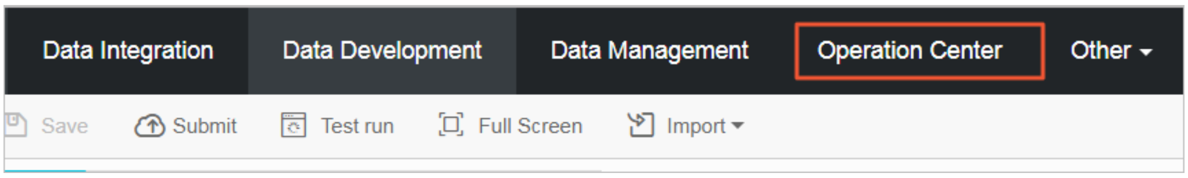


2. In the dialog box, click Confirm Submission.

After the task is submitted, the system prompts The current file is read-only.

Step 6. Check the task

1. At the top of the page, click Operation Center.

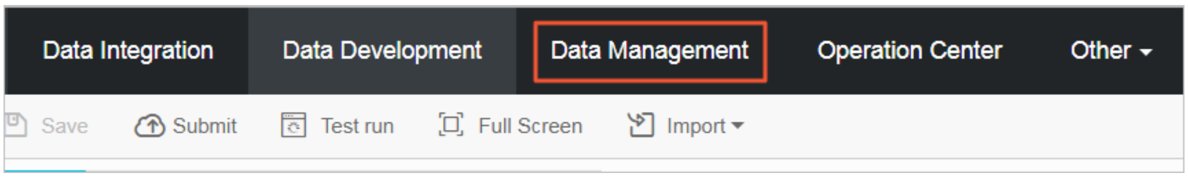


2. In the left-side navigation pane, click Task List > Cycle Task to view the created task OTStoODPS.

3. The task starts running at 00:00 on the next day.
   - In the left-side navigation pane, click Task O&M > Cycle Instance to view scheduling tasks to be executed on the day. Click the instance name to view the details.
   - You can view the log when a task is running or after it is completed.

Step 7. View the data that has been imported to MaxCompute

1. At the top of the page, click Data Management.



2. In the left-side navigation pane, click Query Data. All the tables in MaxCompute are listed.
3. Find the table (ots_gps_data) to which the data is imported, and click the table to go to the table details page.
4. Click Data Preview to view the imported data.

1.3 Full export (script mode)

Data Integration supports data synchronization in wizard mode and script mode. Wizard mode is simpler while script mode is more flexible.

This topic describes how to export full data from Table Store (generated by the Put, Update, and Delete actions) to MaxCompute through Data Integration.

Step 1. Create a Table Store data source.

Note:

- Skip this step if a data source is already created.
- If you do not want to create the data source, you can specify the endpoint, instanceName, AccessKeyID, and AccessKeySecret on the subsequent configurat ion page.

For more information about how to create a data source, see *Create a Table Store data source*.

Step 2. Create a MaxCompute data source

This operation is similar to Step 1. You only need to select MaxCompute as the data source.

In this example, the data source is named "OTS2ODPS".

Step 3. Create a full export tunnel

1. On the *Data IDE* page, click Sync Tasks.
2. Select Script Mode.
3. In the Import Template dialog box that appears, set Source Type to Table Store and Type of Objective to MaxCompute (ODPS).
4. Click OK to go to the configuration page.
5. Set configuration parameters.

```
{
" type ": " job ",
" version ": " 1 . 0 ",
" configurat  ion ": {
" setting ": {
  " errorLimit ": {
    " record ": " 0 "    #  Maximum   number   of   errors
 allowed
  },
  " speed ": {
    " mbps ": " 1 ",    #  Maximum   traffic  ,  in   Mbps  .
    " concurrent ": " 1 "  #  Number   of  concurrent   tasks  .
  }
},
" reader ": {
  " plugin ": " ots ",  #  Name   of   the   plugin   read
  " parameter ": {
    " datasource ": "",  #  Name   of   the   data   source
    " table ": "",  #  Name   of   the   table
    " column ": [  #  Name   of   the   column   in   Table   Store
   that   needs   to   be   exported   to   MaxCompute
      {
        " name ": " column1 "
      },
      {
        " name ": " column2 "
      },
      {
        " name ": " column3 "
      },
      {
        " name ": " column4 "
      },
      {
        " name ": " column5 "
      }
    ],
    " range ": {  #  Range   of   the   data   to   be   exported  .
 In   full   export   mode  ,  the   range   is   from   INF_MIN   to
   INF_MAX  .
```

```
      " begin ": [ #  Start   position   of   the   data   to   be
exported .  The   minimum   position   is   INF_MIN .  The   number
  of   configurat  ion   items   set   in  " begin "  must   be
the   same   as   the   number   of   primary   key   columns   of
  the   table   in   Table   Store .
        {
          " type ": " INF_MIN "
        },
        {
          " type ": " INF_MIN "
        },
        {
          " type ": " STRING ",  #  Indicates   that   the   start
position   in   the   third   column   is   begin1 .
          " value ": " begin1 "
        },
        {
          " type ": " INT ",  #  Indicates   that   the   start
position   in   the   fourth   column   is   0 .
          " value ": " 0 "
        }
      ],
      " end ": [  #  End   position   of   the   data   to   be
exported
        {
          " type ": " INF_MAX "
        },
        {
          " type ": " INF_MAX "
        },
        {
          " type ": " STRING ",
          " value ": " end1 "
        },
        {
          " type ": " INT ",
          " value ": " 100 "
        }
      ],
      " split ": [  #  Indicates   the   partition   scope ,  which
  is   not   configured   in   normal   cases .  If   performanc
e   is   poor ,  you   can   open   a   ticket   to   submit   a
query .
        {
          " type ": " INF_MIN "
        },
        {
          " type ": " STRING ",
          " value ": " splitPoint  1 "
        },
        {
          " type ": " STRING ",
          " value ": " splitPoint  2 "
        },
        {
          " type ": " STRING ",
          " value ": " splitPoint  3 "
        },
        {
          " type ": " INF_MAX "
        }
      ]
    }
  }
```

```
},
" writer ": {
  " plugin ": " odps ",  #  Name   of   the   plugin   written   by
   MaxCompute
  " parameter ": {
    " datasource ": "",  #  Name   of   the   MaxCompute   data
 source
    " column ": [],  #  Name   of   the   column   in   MaxCompute
 .  The   column   name   sequence   correspond  s   to   that   in
   Table   Store .
    " table ": "",  #  Name   of   a   table   in   MaxCompute .  It
   must   be   created   first ;  otherwise ,  the   task   may
 fail .
    " partition ": "",  #  It   is   required   if   the   table
   is   partitione  d .  For   non - partition   tables ,  do
 not   set   this   parameter .  The   partition   informatio  n
 of   the   data   table   must   be   written .  Specify   the
 parameter   until   the   last - level   partition .
    " truncate ": false   #  Indicates   whether   to   clear   the
   previous   data
  }
}
}
}
```

Note:

For detailed configurations, see *Configure Table Store Reader* and *Configure MaxCompute Writer*.

6. Click Save.

Step 4. Run the task (test)

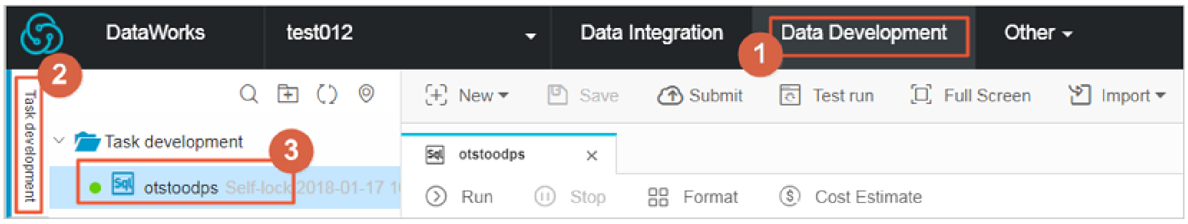
1. At the top of the page, click operation.

   If no variable is included in the configurations, the task is executed immediately. If a variable exists, you must enter the actual value of the variable, and then click OK. Then, the task starts running.

2. After running the task, you can check whether the task is successful, and view the number of exported data rows in the log.

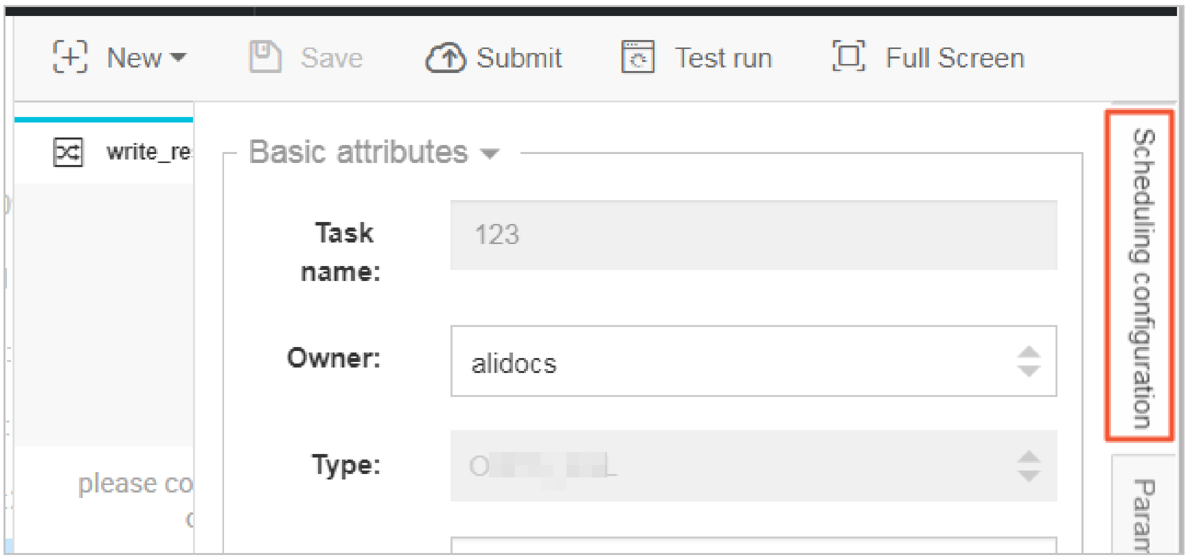
Step 5. Set scheduling parameters

1. At the top of the page, click Data Development.
2. On the Task Development tab, double-click the created task OTStoODPS.



3. Click Scheduling Configuration to set the scheduling parameters.

   To set the task to start running on the next day, configure the following parameters as shown.



Task name: 123

Owner: alidocs

Type: O

The configurations are described as follows:

Parameter	Description
Scheduling status	It is not selected by default, indicating running the task.
Auto retry	We recommend that you select this parameter so that the system can retry after an error occurs.
Activation date	The default value is recommended.
Scheduling period	Minute is used in this example.
Start time	It is set to 00:00 in this example.
Interval	The scheduling interval is set to 5 minutes in this example.
End time	It is set to 23:59 in this example.
Dependency attribute	Set the `Dependency Attribute` based on your business needs, or retain the default value.
Cross-cycle dependency	Select Self-dependent; operation can continue after the conclusion of the previous scheduling period.

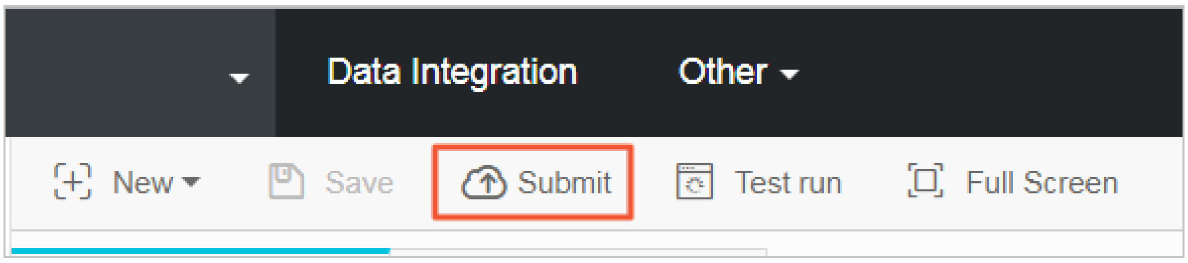
4. Click Parameter Configuration to set the parameters.

   The parameters are described as follows.

Parameter	Description
${bdp.system.bizdate}	It does not need to be configured.
startTime	It is the Start Time variable set in Scheduling Configuration. In this example, it is set to `$[ yyyymmddhh 24miss - 10 / 24 / 60 ]`, indicating a time equal to the scheduling task start time minus 10 minutes.
endTime	It is the End Time variable set in Scheduling Configuration. In this example, it is set to `$[ yyyymmddhh 24miss - 5 / 24 / 60 ]`, indicating a time equal to the scheduling task start time minus 5 minutes.

Step 6. Submit the task

1. At the top of the page, click Submit.

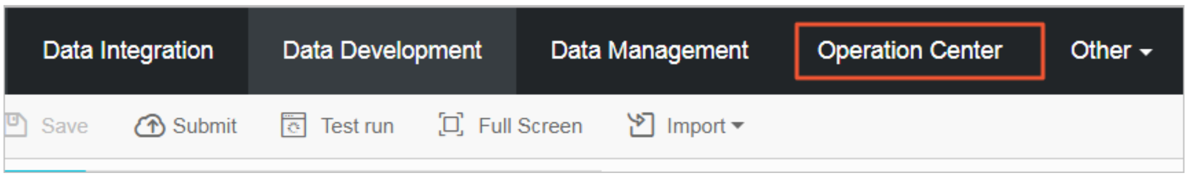


2. In the displayed box, click Confirm Submission.

   After the task is submitted, the current file is read-only.

Step 7. Check the task

1. At the top of the page, click Operation Center.

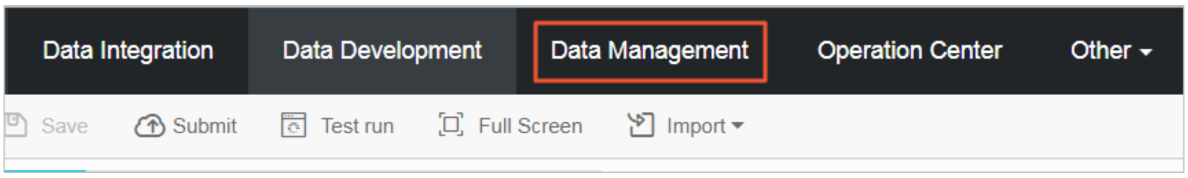


2. In the left-side navigation pane, click Task List > Cycle Task to view the newly created task OTStoODPS.

3. The task starts running at 00:00 on the next day.
    - In the left-side navigation pane, click Task O&M > Cycle Instance to view scheduling tasks to be executed on the day. Click the instance name to view the details.
    - You can view the log when a task is running or after it is completed.

Step 8. View the data that has been imported to MaxCompute

1. At the top of the page, click Data Management.



2. In the left-side navigation pane, click All Data.
3. Find the table (ots_gps_data) to which the data is imported, and click the table to go to its corresponding details page.
4. At the right-side, click the preview data tab to view the imported data.

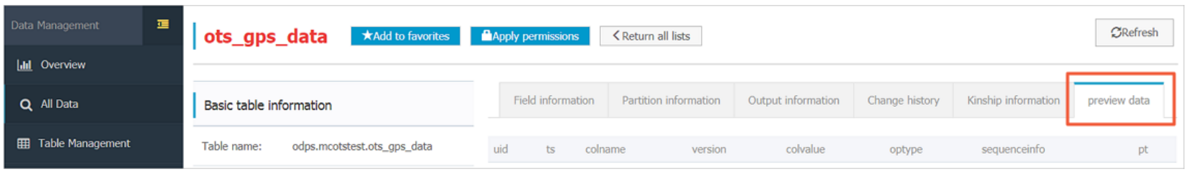


Table name: odps.mcotstest.ots_gps_data

uid	ts	colname	version	colvalue	optype	sequenceinfo	pt

2 OSS

2.1 Overview

Table Store is a distributed NoSQL data storage service that is built on Alibaba Cloud Apsara distributed system. It uses data partitioning and load balancing techniques to seamlessly scale up data size and access concurrency, providing storage of, and real-time access to, massive structured data.

Object Storage Service (OSS) is a massive-volume, secure, low-cost, and highly-reliable cloud storage service. It provides 99.999999999% data reliability. You can use RESTful API for storage and access in any place on the Internet. Its capacity and processing capability can be elastically scaled, and multiple storage modes are provided, comprehensively optimizing the storage cost.

Scenarios

Table Store: Provides professional data-persistent storage service and user-oriented real-time read/write operations with high concurrency and low latency.

OSS: Supports backup at an extremely low cost.

Usage

- Write

  Data can be directly written to Table Store.
- Read

  Data can be directly read from Table Store.
- Back up

  Automatic backup is supported.
- Restoration

  Data can be re-written to Table Store through Data Integration (OSSReader and OTSWriter).

Constraints

- Write by whole row

  Table Store Stream requires that a whole row of data be written to Table Store each time. Currently, the whole-row data write mode is applied to the writing of time sequence data such as IoT data. Therefore, data cannot be modified subsequently.
- Synchronization latency

  Currently, periodic scheduling is used and the scheduling interval is 5 minutes. The plugin has a latency of 5 minutes and the total latency of a synchronization task is 5 to 10 minutes.

Activation

- Activate Table Store

  1. Go to the *Table Store product details page*.
  2. Click Buy Now.
  3. In the *Table Store console*, create *instances* and *data tables*.

  Note:

  - To use the incremental tunnel, you must activate the *Stream function* for data tables. You can select 24 hours for the validity period.
  - Table Store supports the reserved CUs and additional CUs. If the reserved read and write CUs are both set to zero during table creation, then the additional read and write CU is used. You can adjust the reserved read/write CU of each table at any time.
  - Table Store offers each registered account 25 GB of free storage per month.
- Activate OSS

  1. Go to the *OSS product details page*.
  2. Click Buy Now.

Data tunnel

Offline

- Export the full data to OSS.

  - *Script mode*

- Synchronize data to OSS in incremental mode.
  - *Script mode*
- Fully import data into Table Store.
  - Script mode

2.2 Full export (script mode)

Data Integration supports data synchronization in wizard mode and script mode. Wizard mode is simpler, while script mode is more flexible.

This section describes how to export full data in Table Store to OSS using the script mode of Data Integration, so that you can download the data as needed or save it as backup data of Table Store to OSS.

Channels

Script mode of Data Integration:

- Reader: OTSReader
- Writer: OSSWriter

Step 1. Create a Table Store data source

Note:
Skip this step if you have created a Table Store data source.

For more information about how to create a data source, see *Create a Table Store data source*.

Step 2. Create an OSS data source

This operation is similar to Step 1. You only need to select OSS as the data source.

Note:
During parameter configuration of the OSS data source, Endpoint does not contain bucketName.

Step 3. Create an export task

1. Log on to the Data Integration console.
2. On the Sync Tasks page, select Script Mode.

3. In the Import Template dialog box, set Source Type to Table Store (OTS) and Type of Objective to OSS.
4. Click OK to go to the configuration page.

Step 4. Set configuration items

1. On the configuration page, templates for OTSReader and OSSWriter are provided. Complete the configurations by referring to the following annotations.

```
{
" type ": " job ",    #  It   cannot   be   modified .
" version ": " 1 . 0 ",  #  It   cannot   be   modified .
" configurat  ion ": {
 " setting ": {
   " errorLimit ": {
     " record ": " 0 "  #  The   import   task   fails   when   the
   number   of   error   records   exceeds   the   value .
   },
   " speed ": {
     " mbps ": " 1 ",  #  Import   speed ,  in   Mbps .
     " concurrent ": " 1 "  #  Concurrenc  y .
   }
 },
 " reader ": {
   " plugin ": " ots ",  #  It   cannot   be   modified .
   " parameter ": {
     " datasource ": "",   #  Name   of   the   data   source   in
 Data   Integratio  n ,  which   must   be   set   in   advance .
 You   can   configure   data   source   or   write   authentica
 tion   informatio  n   such   as   the   AccessKeyI  D   in
 plaintext .  We   recommend   that   you   configure   data
 source .
     " table ": "",    #  Table   name   in   Table   Store .
     " column ": [   #  Name   of   the   column   that   needs   to
   be   exported   to   OSS .  If   all   the   columns   need   to
   be   exported   to   OSS ,  set   this   parameter   to   an
 empty   array .
       {
         " name ": " column1 "    #  Name   of   the   column   in
 Table   Store ,  which   needs   to   be   imported   to   OSS
       },
       {
         " name ": " column2 "   #  Name   of   the   column   in
 Table   Store ,  which   needs   to   be   imported   to   OSS
       }
     ],
     " range ": {
       " begin ": [
         {
           " type ": " INF_MIN "   #  Start   position   of   the
 first   primary   key   column   in   Table   Store .  If   you
 want   to   export   full   data ,  set   this   parameter   to
 INF_MIN .  If   you   want   to   export   a   portion   of   the
   data ,  set   this   parameter   as   needed .  The   number   of
   configurat  ion   items   in  " begin "  must   be   the   same
 as   the   number   of   primary   key   columns .
         }
       ],
       " end ": [
         {
          " type ": " INF_MAX "   #  End   position   of   the
 first   primary   key   column   in   Table   Store .  If   you
 want   to   export   full   data ,  set   this   parameter   to
 INF_MAX .  If   you   want   to   export   a   portion   of   the
 data ,  set   this   parameter   as   needed .
        }
      ],
      " split ": [  #  Used   to   configure   partition
 informatio  n   about   the   Table   Store   table ,  which   can
   accelerate  s   the   export .  In   the   next   version ,  this
   configurat  ion   is   automatica  lly   processed .
      ]
    }
  }
},
" writer ": {
  " plugin ": " oss ",
  " parameter ": {
    " datasource ": "",  #  Name   of   the   OSS   data   source
    " object ": "",  #  Prefix   of   the   object   excluding
 the   bucket   name ,  for   example ,  tablestore / 20171111 /.
 If   the   export   is   scheduled ,  a   variable ,  for   example
 ,  tablestore /${ date },  must   be   used ,  and  ${ date }  must
   be   configured   when   the   scheduling   parameters   are
 set .
    " writeMode ": " truncate ", #  truncate ,  append ,  and
 nonConflic  t   are   supported .  truncate   is   used   to
 clear   existing   files   with   the   same   name ,  append
 is   used   to   add   the   data   to   existing   files   with
 the   same   name ,  and   nonConflic  t   is   used   to   return
   an   error   when   files   with   the   same   name   exist .
 truncate   is   used   during   full   export .
    " fileFormat ": " csv ", #  CSV   and   TXT   are   supported .
    " encoding ": " UTF - 8 ",  #  Encoding   mode
    " nullFormat ": " null ", #  Defines   a   string   identifier
   that   represents   the   null   value .  It   can   be   an
 empty   string .
    " dateFormat ": " yyyy - MM - dd   HH : mm : ss ",  #  Time
 format
    " fieldDelim  iter ": "," #  Delimiter   of   each   column
  }
 }
}
}
```

2. Click Save to save the task.

Step 5. Run the task

1. Click operation to run the task.

   If the configurations contain variables, for example, *${date}*, the variable setting page is displayed. You can set only specific values.

2. View logs in the lower part of the page.

   If no error is logged, the task is successfully executed, and you can check the data in the target OSS instance.

   > Note:
   > Full export is generally a one-time task, and thus you do not need to set automatic scheduling parameters. For more information about how to set the scheduling parameters, see *Incremental synchronization*.

Step 6. Check the data exported to OSS

1. Log on to the *OSS console*.
2. Select the bucket and file name, and verify its contents.

2.3 Incremental synchronization (script mode)

Data Integration supports data synchronization in wizard mode and script mode. Wizard mode is simpler while script mode is more flexible.

This section describes how to synchronize incremental data in Table Store to OpenSearch using the script mode of Data Integration.

Channels

Script mode of Data Integration

- Reader: OTSStream Reader
- Writer: OSSWriter

Configure Table Store

No prior configurations required.

Configure OSS

No prior configurations required.

Configure Data Integration

1. Create a Table Store data source.

   > Note:
   > - If you have already created a Table Store data source, skip this step.

- If you do not want to create a data source, you can specify the endpoint , instanceName, AccessKeyID, and AccessKeySecret on the subsequent configuration page.

For more information about how to create a data source, see *Create a Table Store data source*.

2. Create an OSS data source.

   This step is similar to Step 1. You only need to select OSS as the data source.

   Note:
   During parameter configuration of the OSS data source, Endpoint does not contain bucketName.

3. Create a synchronization task.
   a. Log on to the *Data Integration console*.
   b. On the Sync Tasks page, select Script Mode.
   c. In the Import Template dialog box that appears, set Source Type to Table Store Stream (OTS Stream) and Type of Objective to OSS.
   d. Click OK to go to the configuration page.
4. Set configuration items.
   a. On the configuration page, templates of OTSStreamReader and OSSWriter are provided. Complete the configurations by referring to the following annotations.

```
{
" type ": " job ",
" version ": " 1 . 0 ",
" configurat  ion ": {
" setting ": {
" errorLimit ": {
 " record ": " 0 "  #  Allowed   number   of   errors .  If
 the   number   of   errors   exceeds   the   value ,  the
 synchroniz  ation   task   fails .
},
" speed ": {
 " mbps ": " 1 ",  #  Maximum   traffic   of   each   synchroniz
 ation   task .
 " concurrent ": " 1 "   #  Number   of   concurrent   synchroniz
 ation   tasks   each   time .
}
},
" reader ": {
" plugin ": " otsstream ",  #  Name   of   the   Reader   plugin .
" parameter ": {
 " datasource ": "",  #  Name   of   the   Table   Store   data
   source .  If   this   parameter   is   set ,  you   do   not
```

```
    need   to   set   endpoint ,  accessID ,  accessKey ,  and
  instanceNa  me .
  " dataTable ": "", #  Name   of   the   table   in   Table
  Store .
  " statusTabl  e ": " TableStore  StreamRead  erStatusTa  ble ",
  #  Table   that   stores   the   Table   Store   Stream   status
  ;  using   the   default   value   is   recommende  d
  " startTimes  tampMillis ": "",  #  Start   time   of   the
  export .  In   incrementa  l   export   mode ,  the   task
  needs   to   be   executed   cyclically ,  and   the   start
  time   is   different   at   each   execution .  Therefore ,  you
    must   set   a   variable ,  for   example , ${ start_time }.
  " endTimesta  mpMillis ": "",  #  End   time   of   the   export
  .  You   must   set   a   variable ,  for   example , ${ end_time
  }.
  " date ": " yyyyMMdd ",  #  Date   from   which   data   is
  exported .  This   parameter   is   the   same   as   startTimes
  tampMillis   and   endTimesta  mpMillis ,  and   therefore   must
    be   deleted .
  " mode ": " single_ver  sion_and_u  pdate_only ", #  Format
  of   the   data   exported   from   Table   Store   Stream .
  Currently ,  the   parameter   must   be   set   to   single_ver
  sion_and_u  pdate_only .  Add   this   parameter   if   it   is
  not   in   the   configurat  ion   template .
  " column ":[  #  Names   of   the   columns   to   be   exported
    from   Table   Store   to   OSS .  Add   this   parameter   if
    it   is   not   in   the   configurat  ion   template .  Set
  this   parameter   as   needed .
          {
             " name ": " uid "  #  Name   of   the   column .  It
    is   the   primary   key   column   in   Table   Store .
          },
          {
             " name ": " name "  #  Name   of   the   column .  It
    is   an   attribute   column   in   Table   Store .
           },
  ],
  " isExportSe  quenceInfo ":  false , #  This   parameter   can
    only   be   set   to   false   in   single_ver  sion_and_u
  pdate_only   mode .
  " maxRetries ":  30  #  Maximum   number   of   retry   times .
 }
 },
 " writer ": {
 " plugin ": " oss ", #  Name   of   the   Writer   plugin
 " parameter ": {
  " datasource ": "", #  Name   of   the   OSS   data   source
  " object ": "",  #  Prefix   of   the   name   of   the   last
  file   to   be   backed   up   to   OSS .  The   recommende  d
    value   is   the   Table   Store   instance   name ,  table
  name ,  or   date ,  for   example , " instance / table /{ date
  }".
  " writeMode ": " truncate ", #  truncate ,  append ,  and
  nonConflic  t   are   supported .  truncate   is   used   to
  clear   existing   files   with   the   same   name ,  append
  is   used   to   add   the   data   to   existing   files   with
    the   same   name ,  and   nonConflic  t   is   used   to
  return   an   error   when   files   with   the   same   name
  exist .
  " fileFormat ": " csv ", #  File   format
  " encoding ": " UTF - 8 ", #  Encoding   mode
  " nullFormat ": " null ", #  Mode   of   representa  tion   in
  a   TXT   file   under   control
  " dateFormat ": " yyyy - MM - dd   HH : mm : ss ", # #  Time
  format
  " fieldDelim  iter ": "," #  Delimiter   of   each   column
}
}
}
}
```

> Note:
> For detailed configuration description, see *Configure OTSStreamReader* and *Configure OSSWriter*.

b. Click Save.

5. Run the task.

    a. Click operation.
    b. In the dialog box that appears, set the variable parameters.
    c. Click OK.
    d. After the task is completed, log on to the *OSS console* to verify whether files are backed up.

6. Configure scheduling.

    a. Click Submit.
    b. In the dialog box that appears, set the scheduling parameters.

    The parameters are described as follows.

Parameter	Description
Scheduling type	Select cycle control.
Automatically re-run	This parameter indicates that the task reruns for three times at an interval of 2 minutes if the task fails.
Start date	The default value is recommended, which is from January 1, 1970 to 100 years later.
Scheduling cycle	Select Minute.
Start Time	Select “00:00 to 23:59”, which indicates that scheduling is required for a full day.
Interval	Select 5 Minutes.

Parameter	Description
start_time	Enter `$[yyyymmddhh24miss-10/24/60]`, which indicates the time of the scheduling task minus 10 minutes.
end_time	Enter `$[yyyymmddhh24miss-5/24/60]`, which indicates the time of the scheduling task minus 5 minutes.
date	Enter `${bdp.system.bizdate}`, which indicates the scheduling date.
Dependency attributes	Set this parameter if a dependency exists. If no dependency exists, do not set this parameter.
Cross-cycle dependency	Self-dependent: The operation can continue only after the previous scheduling cycle is completed.

c. Click OK.

The periodic synchronization task is configured, the configuration file status is Read-only.

7. Check the task.

   a. At the top of the page, click Operation Center.
   b. On the left-side navigation pane, click Task List > Cycle Task to view the created synchronization task.
   c. The new task begins running at 00:00 on the next day.
      - In the left-side navigation pane, choose Task O&M > Cycle Instance to view each pre-created synchronization task of the day. The scheduling interval is 5 minutes and each task processes data from the past 5 to 10 minutes.
      - Click the instance name to view its details.
   d. You can view the log when a task is running or after it is completed.

8. Check the data exported to OSS.

   Log on to the *OSS console* to check whether a new file is generated and whether the file content is correct.

Once the preceding settings are completed, data in Table Store can be automatically synchronized to OSS at a latency of 5 to 10 minutes.

3 LogHub Shipper for Table Store

3.1 Overview

LogHub Shipper for Table Store (LogHub Shipper) writes data from Log Service to a specified table in Table Store after data scrubbing and conversion. This service publishes data to the Alibaba Cloud Container Hub by using the Docker image method and runs on your ECS instances based on Container Service.

Description

Log Service stores data in the JSON format, and writes to and reads from a *log group* as the basic unit. Therefore, you cannot quickly search and analyze logs based on specific conditions in Log Service, for example, log data of an app for the last 12 hours.

LogHub Shipper converts log data in Log Service into structured data, and then writes the data to data tables in Table Store in real time. This provides an accurate and high-performance online service in real time.

Example

For example, Log Service contains log data in the format as follows:

```
{" __time__ ": 1453809242 ," __topic__ ":""," __source__ ":" 10 . 170
 . 148 . 237 "," ip ":" 10 . 200 . 98 . 220 "," time ":" 26 / Jan /
 2016 : 19 : 54 : 02  + 0800 "," url ":" POST  / PutData ?  Category
 = YunOsAccou  ntOpLog & AccessKeyI  d = U0U *** 45A & Date = Fri % 2C
 % 2028 % 20Jun % 202013 % 2006 % 3A53 % 3A30 % 20GMT & Topic = raw &
 Signature = pD12XYLmGx  KQ % 2Bmkd6x7hA  gQ7b1c % 3D   HTTP / 1 . 1
 "," status ":" 200 "," user - agent ":" aliyun - sdk - java "}
```

When LogHub Shipper writes the data to a data table that contains the ip and time primary keys in Table Store, the data format is as follows.

ip	time	source	status	user-agent	url
10.200.98.220	26/Jan/2016:19:54:02 +0800	10.170.148.237	200	aliyun-sdk-java	POST /PutData···

In this way, you can easily and accurately retrieve history data of a specified IP address based on a specified time period by using Table Store.

LogHub Shipper provides flexible data mapping rules. You can configure the mappings between the fields of log data and the attribute columns of data tables and easily convert the data.

Concepts

Related products

Before using LogHub Shipper, you should understand the following concepts:

- Log Service
  - *Endpoint, Project, Logstore, and Partition*
  - *Consumer group*

    We recommend that you use the same unique consumer group in LogHub Shipper for the same project, Logstore, and target table.
- Table Store
  - *Endpoint, Instance*
  - *Table, Primary key column, and Attribute column*
  - *Throughput*
- ECS instance
  - *Pay-As-You-Go, Subscription*
- Container Service
  - *Cluster, Node, Application, Service, and Container*

    We recommend that the number of containers in a single LogHub Shipper process be the same as or less than the number of partitions in the correspond ing Logstore.
- Access control
  - *RAM user*

    We recommend that you authorize the RAM user of LogHub Shipper to only read from Logstores and write to Table Store.

Data table

This is a target table that stores your log data after data scrubbing and conversion.

When using a data table, follow these rules:

- You have to manually create a target table, because LogHub Shipper does not automatically create tables.
- If Log Service and Table Store are both available, the latency between the time when a log entry enters Log Service and the time when the log entry goes to Table Store is measured in a few hundred milliseconds (ms).
- When Table Store is unavailable, LogHub Shipper will wait for a period of 500 ms or less and try again.
- LogHub Shipper regularly records persistent breakpoints.
- If LogHub Shipper is unavailable, for example, during an upgrade, the service continues to consume logs from the last breakpoint upon recovery.
- We recommend that different log entries in the same Logstore correspond to different rows in the target table. Therefore, any retries cannot affect the eventual consistency of the target table.
- LogHub Shipper writes data by using the UpdateRow operation in Table Store. Therefore, multiple LogHub Shipper processes can share the same target table. In this situation, we recommend that LogHub Shipper write data to different attribute columns during these processes.

Status table

LogHub Shipper uses a status table that you create in Table Store to indicate some status.

When using a status table, follow these rules:

- Multiple LogHub Shipper processes can share the same status table.
- When no errors occur, each LogHub Shipper container adds a record to the status table at five-minute intervals.
- When an error occurs but Table Store is still available, each LogHub Shipper container immediately adds a record to the status table.
- We recommend that you set Time To Live (TTL) by days to only keep recent data.

The status table has four primary key columns:

- project_logstore: String type. This column indicates the project and Logstore of Log Service, separated with vertical bars (|).
- shard: Integer type. This column indicates the shard number in Log Service.
- target_table: String type. This column indicates the name of the target table where you store data in Table Store.

- timestamp: Integer type. This column indicates the time when a LogHub Shipper container adds a record to the status table. This is UNIX time, measured in milliseconds.

In addition, the following attribute columns record data import status. In any row in a status table, all attribute columns are optional and may not exist.

- shipper_id: String type. This column indicates the ID of a LogHub Shipper container. Currently, this is the name of the container host.
- error_code: String type. This column indicates an *error code* defined in Table Store. This attribute column does not exist if no error occurs.
- error_message: String type. This column indicates the specific error message that Table Store returns. This attribute column does not exist if no error occurs.
- failed_sample: String type. This column indicates an error log entry as a JSON string.
- `__time__` : Integer type. This column indicates the maximum value that the specified LogHub Shipper container writes to the *_time__ field* of log data in Table Store after this container last updates the status table.
- row_count: Integer type. This column indicates the number of log entries that the specified LogHub Shipper container writes to Table Store after this container last updates the status table.
- cu_count: Integer type. This column indicates the number of *Capacity Units (CUs)* that the specified LogHub Shipper container consumes after this container last updates the status table.
- skip_count: Integer type. This column indicates the number of log entries that the specified LogHub Shipper container cleans after this container last updates the status table.
- * skip_sample: Str type. This column indicates one of the log entries, a JSON string , that the LogHub Shipper container discards after this container last updates the status table. The log of the container records each discarded log entry and the reason for discarding the log entry.

Configuration

When you create LogHub Shipper, specify the following environment variables for the container:

- access_key_id and access_key_secret: these are the AccessKeyId and AccessKeyS ecret of the Alibaba Cloud account that you use in LogHub Shipper.
- loghub: this is the configuration of the Log Service instance that LogHub Shipper requires. This JSON object includes:
  - endpoint
  - logstore
  - consumer_group
- tablestore: this is the configuration of the Table Store instance that LogHub Shipper requires. This JSON object includes:
  - endpoint
  - instance: the name of the instance.
  - target_table: the name of the data table. This table must exist under this instance.
  - status_table: the name of the status table. This table must exist under this instance.
- exclusive_columns: the blacklist of attribute columns. This is a JSON array that consists of JSON strings.

  If you have set this variable to a field, LogHub Shipper does not write the specified field to the target table. For example, the target table contains primary key A, the exclusive_columns environment variable has ["B", "C"] configured, and a log entry contains three fields: A, B, and D. Then, one row appears in the target table to indicate the log entry. This row contains primary key A and attribute column D . Column C does not exist in the log entry, so LogHub Shipper does not write this column to the target table. Column B exists in the log entry, but this column is specified as an exclusive column, so LogHub Shipper does not write this column to the target table either.
- transform: indicates a simple conversion. This is a JSON object. The key in this variable is the name of the column that can be a primary key column written in the

target table. The value is the simple conversion expression that LogHub Shipper defines as follows:

- A log field is an expression.
- An unsigned integer is an expression.
- A string in double quotes is an expression. The string can contain the escape characters `\"` and `\\\\`.
- `( func arg ... )` is also an expression. Zero or multiple spaces or tabs can exist preceding and following the parentheses. At least one space exists between func and the parameter that follows func, and between different parameters. Each parameter must be an expression. The system supports the following functions:
  - `-> int` : converts a string to an integer. This function requires two parameters. The first is the base, which can be 2 to 36. The second is the string that LogHub Shipper converts. The letter in the second parameter is case insensitive and indicates a number from 10 to 35.
  - `-> bool` : converts a string to a Boolean value. This function requires one parameter that is the string LogHub Shipper converts. "true" corresponds to a true value and "false" corresponds to a true value. Other strings are regarded as errors.
  - `crc32` : calculates CRC32 for a string and outputs the result as an Integer value. This function requires one parameter that is the string LogHub Shipper converts.

If a log entry is missing or an error occurs during conversion, the column correspond ing to the key is regarded as a missing column. If an error occurs, the log of the container records error details.

Data scrubbing follows only one rule: if a primary key column is missing, LogHub Shipper cleans the corresponding log entry.

3.2 Prepare the environment

This topic describes how to build LogHub Shipper for TableStore (LogHub Shipper), and describes how LogHub Shipper converts log data in Log Service into structured data and stores the data to Table Store.

Prerequisites

Log Service

You have activated Log Service and requested a project and a Logstore. LogHub Shipper does not modify log data from Log Service, and can use the project and Logstore that you have requested. In the following example, the project is lhshipper-test, the Logstore is test-store, and the region is China (Hangzhou).

Table Store

You have activated Table Store and prepared two tables.

- The first table is a data table that stores log data synchronized from Log Service. This table has three primary key columns as follows:
  - rename: the type is STRING.
  - trans-pkey: the type is INTEGER.
  - keep: the type is STRING.
- The second table is a status table for LogHub Shipper. This table stores informatio n about the progress of synchronizing log data from each project and each shard of Log Service. LogHub Shipper for multiple projects and Logstores can share the same status table. We recommend that you set the Time To Live (TTL) of this table to one or two days to reduce the usage costs. The status table has four primary key columns as follows:
  - project_logstore: the type is STRING.
  - shard: the type is INTEGER.
  - target_table: the type is STRING.
  - timestamp: the type is INTEGER.

Access control

You have requested the AccessKeyId and AccessKeySecret as a Resource Access Management (RAM) user.

To secure data, we recommend that you use a RAM user to build LogHub Shipper. You can authorize the RAM user to read log data from Log Service (AliyunLogReadOnly) and to write log data to Table Store (AliyunTableStoreWriteOnlyAccess).

ECS and Container Service

You have activated an Elastic Compute Service (ECS) instance and Container Service. You need to create a Pay-As-You-Go ECS instance in follow-up steps.

Build LogHub Shipper

1. Log on to the *Container Service console*.
2. In the left-side navigation pane, click Clusters to go to the Cluster List page.
3. Click Create Cluster in the upper-right corner to go to the Create Cluster page.
4. To set a cluster, follow these rules:
    - Try to select a region where Log Service and Table Store are located. Then, you can use a private IP address to avoid the latency and downstream traffic fees in a public network.
    - Do not select the Swarm Mode Cluster checkbox in the example.

      However, we recommend that you select Swarm Mode Cluster in the running environment that your service requires to achieve better performance.
    - Click Create in the example.

      However, we recommend that you click Add in the running environment that your service requires to add an existing ECS instance.
    - LogHub Shipper does not require high-configuration instances. Select 1 Core 1 G from the Instance Type drop-down list.
    - LogHub Shipper supports dynamic and horizontal scaling. You can select multiple ECS instances.
    - LogHub Shipper does not use HTTP to transmit data, and does not expose any port. Do not select the Automatically Create Server Load Balancer checkbox.
5. Click Create in the upper-right corner to create a cluster.

   The cluster has a delay in starting initialization. You can check the cluster status on the Cluster List page.

Note:

If you have an ECS instance, you can add this instance to the specified cluster. For more information, see *Add an existing ECS instance*.

Create an application

1. Log on to the *Container Service console*.
2. In the left-side navigation pane, click Applications to go to the Application List page.
3. Click Create Application in the upper-right corner.
4. To set basic information of the application named loghub-shipper in this example, follow these rules:
   - Select the cluster where you want to create the application from the Cluster drop-down list.
   - We recommend that you select the Pull Docker Image checkbox to facilitate follow-up version upgrades.
5. Click Create with Image.

   Note:
   You click Create with Image in the example to describe the core procedure. However, in the running environment that your service requires, an application may contain multiple services. In this case, you can click Create with Orchestrat ion Template to easily manage the application.
6. Configure the application. Follow these rules:
   - To locate the required image, type loghub-shipper in the search text box and click Global search.
   - In the Environment field, specify the following variables:
     - access_key_id
     - access_key_secret
     - loghub: a JSON string that indicates the information of Log Service and that includes endpoint, logstore, and consumer_group. In this string, consumer_group can be any character string. Multiple container instances can share consumer_group for the same LogHub Shipper process. But

consumer_group cannot be repeated for multiple LogHub Shipper processes. This variable is shown as follows:

```
{" endpoint ": " https :// lhshipper - test . cn - hangzhou .
 log . aliyuncs . com ",
" logstore ": " test - store ",
" consumer_g  roup ": " defaultcg "}
```

- tablestore: a JSON string that indicates the information of Table Store. The variable includes endpoint (the domain name of the Table Store instance), instance (the name of the Table Store instance), target_table (the name of a data table), and status_table (the name of a status table). This variable is shown as follows:

```
{" endpoint ": " https :// lhshipper - test . cn - hangzhou .
 ots . aliyuncs . com ",
" instance ": " lhshipper - test ",
" target_tab  le ": " loghub_tar  get ",
" status_tab  le ": " loghub_sta  tus "}
```

- exclusive_columns: a JSON string to indicate the field of log data that is not imported to Table Store. This variable is shown as follows:

```
[" __source__ "," time "]
```

- transform: a JSON string that indicates format conversion for log data. All log data is string type. For example, you can convert data between the rename attribute column and another attribute. This variable is shown as follows:

```
{" rename ": " original ",
" trans - pkey ": "(-> int   10   original )"}
```

In this example, LogHub Shipper changes the log data in the original field into the rename attribute column in the data table. LogHub Shipper also converts the log data in the original field into decimal integers and saves the data to the trans-pkey attribute column of the data table. For more information about type conversions, see Concept and configuration information.

Note:

You do not need to specify primary keys for the data table during the configuration. LogHub Shipper automatically reads schema information of the data table. However, log data or the transform variable must include all primary key fields. Otherwise, the system discards the corresponding log data.

7. Click Create. Service deployment takes a short period of time. You can check the service status on the Service List page.

Then, LogHub Shipper is ready to run.

3.3 User guide

This topic describes how to use LogHub Shipper for Table Store (LogHub Shipper).

Check service details

1. Log on to the *Container Service console*.
2. In the left-side navigation pane, choose Container Service - Swarm > Services to go to the Service List page.
3. Click a service name in the Name column such as loghub-shipper to go to the Details page. On the Containers tab page, you can perform the following operations:

   - View service logs.

     Click Logs in the `Action` column to view real-time logs of LogHub Shipper.
   - Check the running status and alarm rules of the container.

     a. Click Monitor in the `Action` column to check the running status of the container.
     b. Click View History Monitoring Data/Set Alarm Rules to view monitoring charts and alarm rules. On this page, you can create alarm rules.

Example

Configure the service as follows:

```
loghub :{" endpoint ":" https :// lhshipper - test . cn - hangzhou .
log . aliyuncs . com ", " logstore ":" test - store "," consumer_g
roup ":" defaultcg "}
tablestore :{" endpoint ":" https :// lhshipper - test . cn -
hangzhou . ots . aliyuncs . com ", " instance ":" lhlshipper - test
", " target_tab  le ":" loghub_tar  get ", " status_tab  le ":"
loghub_sta  tus "}
exclusive_  columns : [" __source__ ", " __time__ "]
```

```
transform : {" rename ":" original ", " trans - pkey ":"(-> int   10
  original )"}
```

Write data to Log Service

You can call an API operation of Log Service to write data to Log Service, as shown in the following example:

```
LogItem   logItem  =  new   LogItem ();
log . PushBack (" original ", " 12345 ");
log . PushBack (" keep ", " hoho ");
ArrayList   logs  =  new   ArrayList < LogItem >();
logs . add ( log );
loghub . PutLogs (" lhshipper - test ", " test - store ", " smile ",
logs , "");
```

This log entry contains two fields related to users. One is the original field whose value is 12345, and the other is keep whose value is hoho. Log Service also adds three fields to the log entry. The topic field is set to smile, and the __source__ and __time__ fields change with the environment.

View data

By using a tool, you can see that one data item appears in the data table of Table Store . The following example uses the JSON format.

```
[{" rename ": " 12345 ", " trans - pkey ":  12345 , " keep ": " hoho
 "},
{" __topic__ ": " smile ", " original ": " 12345 "}]
```

In this example, rename, trans-pkey, and keep are primary key columns, and __topic__ and original are attribute columns.

Based on environment variables, follow these configuration rules:

- Define `" rename ": " original "` in the transform field. The original field value is 12345 in the log data, so the rename field value is also 12345 in Table Store.
- Define `" trans - pkey ": "(-> int   10   original )"` in the transform field. Thus, LogHub Shipper converts the original field value to a decimal integer, and writes the value to the trans-pkey column of Table Store.
- The keep field value does not require conversion. This value is the same in both Table Store and log data.
- LogHub Shipper does not write data in the __source__ and __time__ fields to the data table, because the exclusive_columns environment variable has [“__source__ “, “__time__ “] configured.

· LogHub Shipper writes the __topic__ and original fields as attribute columns to the data table.

Query synchronization status

The status table updates the status of each process of LogHub Shipper at five-minute intervals. The following example shows the status data in the JSON format.

```
[{" project_lo  gstore ": " lhshipper - test | test - store ", " shard
 ":  0 , " target_tab  le ": " loghub_tar  get ", " timestamp ":
 1469100705  202 },
{" skip_count ":  0 , " shipper_id ": " fb0d62cacc  94 - loghub -
 shipper - loghub - shipper - 1 ", " cu_count ":  1 , " row_count ":
 1 , " __time__ ":  1469100670 }]
```

A worker (” shipper_id” : “fb0d62cacc94-loghub-shipper-loghub-shipper-1”) of LogHub Shipper adds a status item (” timestamp” : 1469100705202) at 2016-07-21 T11 :31:45.202000Z.

During the five minutes before the timestamp, this worker consumed one log entry from shard 0 (“shard” : 0) of the test-store Logstore (“project_logstore” : “ lhshipper-test|test-store”) in the lhshipper-test log project. The worker skips no log (“skip_count” : 0), writes one log (“row_count” : 1), and consumes one Capacity Unit or CU (“cu_count” : 1).

Indicate a wrong log format

The status table shows the number of log entries skipped in the past five minutes. System exceptions or system upgrades cause some log entries that do not follow the format requirement. LogHub Shipper cannot convert these log entries, and has to skip them.

For example, LogHub Shipper processes the following log data:

```
LogItem   log  =  new   LogItem ()
log . PushBack (" original ", " abcd ")
log . PushBack (" keep ", " hoho ")
ArrayList   logs  =  new   ArrayList < LogItem >()
logs . add ( log )
loghub . PutLogs (" lhshipper - test ", " test - store ", " smile ",
logs , "")
```

Based on environment settings, LogHub Shipper converts the original field of log data to an integer, and writes the value to the trans-pkey column in the data table. But the value of original is not a number in the preceding log data, so the status table shows the following data:

```
[{" project_lo  gstore ": " lhshipper - test | test - store ", " shard
 ":  0 , " target_tab  le ": " loghub_tar  get ", " timestamp ":
 1469102805  207 },
{" skip_sampl  e ": "{\" __time__ \": 1469102565 ,\" __topic__ \":\"
 smile \",\" __source__ \":\" 10 . 0 . 2 . 15 \",\" original \":\"
 abcd \",\" keep \":\" hoho \"}", " skip_count ":  1 , " shipper_id
 ": " fb0d62cacc  94 - loghub - shipper - loghub - shipper - 1 ", "
 cu_count ":  0 , " row_count ":  0 , " __time__ ":  0 }]
```

In the status table, skip_count is 1. The skip_sample attribute column indicates that the original log data has been skipped.

Also, the log of Container Service shows the following data:

```
 loghub - shipper_lo  ghub - shipper_1  |  2016 - 07 - 21T12 : 02 : 56
 . 113581003Z   12 : 02 : 56 . 111  [ pool - 4 - thread - 3 ]  ERROR
 shipper . error  -  abcd   is   not   10 - based   int
 loghub - shipper_lo  ghub - shipper_1  |  2016 - 07 - 21T12 : 02 : 56
 . 114039933Z   12 : 02 : 56 . 111  [ pool - 4 - thread - 3 ]  INFO
 shipper . core  -  skip   1   rows
 loghub - shipper_lo  ghub - shipper_1  |  2016 - 07 - 21T12 : 02 : 56
 . 139854766Z   12 : 02 : 56 . 139  [ pool - 4 - thread - 3 ]  INFO
 shipper . core  -  skip : {" __time__  "  1469102565 , " __topic__ " "
 smile ", " __source__ " " 10 . 0 . 2 . 15 ", " original " " abcd ", "
 keep " " hoho "}
```

The log indicates the cause of the skipped data: abcd is not 10-based int.

Modify the configuration

1. Log on to the *Container Service console*.
2. In the left-side navigation pane, click Services to go to the Service List page.
3. Click Update in the Action column next to the LogHub Shipper service to go to the edit page.
4. Click OK.

Upgrade the image

1. Log on to the *Container Service console*.
2. In the left-side navigation pane, click Applications to go to the Application List page.
3. Click Redeploy in the Action column next to the LogHub Shipper service to go to the edit page.
4. Click OK.

Scale the service

LogHub Shipper is highly scalable and automatically allocates shards based on the number of container instances. When the data transmission capability does not meet the service requirements, you can easily scale the service.

Scale out the service

You can scale out the service in two ways. If you frequently scale out and in the service, you can use *Auto Scaling* and *Resource Orchestration Service*.

- Add a node.
  1. Log on to the *Container Service console*.
  2. In the left-side navigation pane, click Clusters to go to the Cluster List page.
  3. Click Manage in the Action column next to the cluster where LogHub Shipper is deployed.
  4. Click Expand on the right of the cluster page to add a Pay-As-You-Go instance, or click Add Existing Instances to add existing instances to the cluster.

     Note:
     For more information about adding existing instances, see *Add an existing ECS instance*.
- Add containers to LogHub Shipper: the nodes in a cluster do not increase with the containers of LogHub Shipper. To modify the configuration of LogHub Shipper and then increase the number of LogHub Shipper containers, follow these steps:
  1. Log on to the *Container Service console*.
  2. In the left-side navigation pane, click Services to go to the Service List page.
  3. Click Update in the Action column next to the LogHub Shipper service to go to the edit page.
  4. Change the number of containers, and click OK.

Scale in the service

To scale in the service, follow these steps:

1. Log on to the *Container Service console*.
2. In the left-side navigation pane, click Services to go the Service List page.

3. Click Update in the Action column next to the LogHub Shipper service to go to the edit page.
4. Reduce the value of Container Quantity.
5. Click OK to return to the Service List page.
6. Click Reschedule in the Action column next to the LogHub Shipper service.
7. In the left-side navigation pane, click Nodes to go to the Node List page.
8. Choose More > Remove in the Action column next to the instance that you want to remove.
9. Different from scaling out, Container Service does not release the removed ECS instance. You have to manually release the instance in the ECS console.