

# Alibaba Cloud Log Service

## Data shipping

Issue: 20190816

## 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.	 <b>Danger:</b> 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.	 <b>Warning:</b> 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.	 <b>Notice:</b> 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.	 <b>Note:</b> You can use Ctrl + A to select all files.
>	Multi-level menu cascade.	Settings > Network > Set network type
<b>Bold</b>	It is used for buttons, menus, page names, and other UI elements.	Click OK.
Courier font	It is used for commands.	Run the <code>cd / d C :/ windows</code> command to enter the Windows system folder.
<i>Italics</i>	It is used for parameters and variables.	<code>bae log list --instanceid Instance_ID</code>
[ ] or [a b]	It indicates that it is an optional value, and only one item can be selected.	<code>ipconfig [-all -t]</code>

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



# Contents

---

Legal disclaimer.....	I
Generic conventions.....	I
1 Overview.....	1
2 Manage LogShipper tasks.....	2
3 Ship logs to OSS.....	4
3.1 Ship logs to OSS.....	4
3.2 JSON storage.....	13
3.3 CSV storage.....	15
3.4 Parquet storage.....	18
3.5 Snappy compression.....	22
3.6 Advanced RAM authorization.....	24
4 Ship data to MaxCompute.....	28
4.1 Ship data to MaxCompute by using DataWorks.....	28
5 Send logs to an SIEM system.....	42
5.1 Overview.....	42
5.2 Use HTTPS to send logs to an SIEM system.....	44
5.3 Use Syslog to send logs to an SIEM system.....	52



# 1 Overview

---

After you access a log source to Log Service, Log Service starts to collect logs in real time and allows you to consume and ship logs in the console or by using SDKs/APIs. Log Service can ship logs collected to LogHub to Alibaba Cloud storage products such as Object Storage Service (OSS) and Table Store in real time. You can configure to ship logs in the console and LogShipper provides a complete status API and automatic retry function.

## Application scenarios

Interconnection with the data warehouse

## Log source

The LogShipper function of Log Service ships logs that are collected to LogHub. After logs are generated, Log Service collects these logs in real time and ships them to other cloud products for storage and analysis.

## Targets

- OSS (large-scale object storage)
  - [#unique\\_4](#)
  - Formats in OSS can be processed by using Hive. E-MapReduce is recommended.
- Table Store (NoSQL data storage service)
- Maxcompute (large data computing services):
  - Delivery via dataworks Data Integration-operation [-#unique\\_5](#)

## 2 Manage LogShipper tasks

LogShipper is a function in Log Service that allows you to maximize your data value . You can ship the collected logs to Object Storage Service (OSS) in the console to store data for a long term or consume data together with other systems such as E-MapReduce. After the LogShipper function is enabled, Log Service backend regularly ships the logs written to the Logstore to the corresponding cloud products. The Log Service console provides the OSS Shipper page for you to query the data shipping status within a specified time range, which allows you to know the shipping progress and handle online issues in time.

On the Logstore List page, click OSS in the left-side navigation pane. The OSS Shipper page appears. You can manage your LogShipper tasks in the following ways.

### Enable/disable LogShipper tasks

1. Select the target Logstore on the OSS Shipper page.
2. Click Enable or Disable to enable or disable the tasks.

You must reconfigure the shipping rule after you enable the tasks again.

You must reconfigure the shipping rule after you enable the tasks again.

### Configure a shipping rule

After enabling the LogShipper tasks, click Setting to modify the shipping rule.

### View details of a LogShipper task

You can filter the LogShipper tasks to be viewed based on the Logstore, time range, and task shipping status. Then, you can view the details of a specific LogShipper task on this page, such as the status, start time, end time, time when logs are received, and type.

A LogShipper task has three kinds of status.

Status	Description	Operation
Success	Logs are successfully shipped.	No need to pay attention.
Running	Logs are being shipped.	Check whether or not logs are successfully shipped later.

Status	Description	Operation
Failed	Logs failed to be shipped. The LogShipper task has encountered an error because of external reasons and cannot be retried.	For more information, see Manage LogShipper tasks in Ship logs to OSS.

### Delete shipping configuration

#### Procedure

1. On the Logstore list page, click Delete rule.
2. Click Confirm in the dialog box.

Once deleted, you will no longer be able to create an offline archive configuration with the same name. Please choose carefully.

## 3 Ship logs to OSS

---

### 3.1 Ship logs to OSS

Log Service can automatically archive Logstore data to Object Storage Service (OSS) to achieve more functions of logs.

- OSS data supports lifecycle configuration for long-term log storage.
- You can consume OSS data by using self-built programs and other systems (for example, E-MapReduce).

#### Function advantages

Using Log Service to ship logs to OSS has the following advantages:

- Ease of use. You can configure to synchronize Logstore data of Log Service to OSS in the console.
- Improved efficiency. The log collection of Log Service centralizes logs of different machines, without repeatedly collecting logs from different machines to import to OSS.
- Ease of management Shipping logs to OSS can fully reuse the log grouping in Log Service. Logs in different projects and Logstores can be automatically shipped to different OSS bucket directories, which facilitates the OSS data management.

#### Prerequisites

1. Activate Log Service, create a project and Logstore, and collect log data.
2. Activate OSS, create a bucket in the region where the Log Service project resides.
3. Activate RAM access control.
4. The Log Service project and OSS bucket must be located in the same region. Cross-region data shipping is not supported.

#### Procedure

##### Step 1. Resource Access Management (RAM) authorization

Before you perform a shipping task, Log Service must be granted a permission to write to OSS .

Go to [RAM quick authorization](#) page, on the displayed page, click Agree to Authorize. After authorization is complete, Log Service has a corresponding write permission to OSS.

**Note:**

- For more information about how to modify the authorization policy and configure cross-account shipping task, see [OSS Shipper - Advanced RAM authorization](#).
- For more information about how to authorize sub-account to perform a shipping task, see [#unique\\_9](#) to access Log Service.

**Step 2. Configure an OSS shipping rule in Log Service**

1. Log on to the Log Service console.
2. On the Project List page, click the project name.
3. Select a Logstore, and click OSS in the left-side navigation pane.
4. Click Enable, set the OSS LogShipper configurations, and click Confirm.

See the following table to complete the OSS shipping configurations.

Configuration item	Description	Value range
OSS Shipping Name	The name of the OSS shipping.	The name can be 3–63 characters long, contain lowercase letters , numbers, hyphens (-), and underscores (_), and must begin and end with a lowercase letter or number .
OSS Bucket	The name of the OSS bucket.	Must be an existing bucket name, and make sure the OSS bucket is in the same region as the Log Service project.
OSS Prefix	The prefix of OSS. Data synchronized from Log Service to OSS is stored in this bucket directory.	Must be an existing OSS prefix.

Configuration item	Description	Value range
Partition Format	Use %Y, %m, %d, %H, and %M to format the creation time of the LogShipper task to generate the partition string. This defines the directory hierarchy of the object files written to OSS, where a forward slash (/) indicates a level of OSS directory. The following table describes how to define the OSS target file path by using OSS prefix and partition format.	For more information about formatting, see <a href="#">Strptime API</a> .
RAM Role	The Arn and name of the RAM role. The RAM role is used to control the access permissions and is the identity for the OSS bucket owner to create a role. The ARN of the RAM role can be viewed in the basic information of this role.	For example, <code>acs : ram :: 45643 : role / aliyunlogd : defaultrole</code> .
Shipping Size	Automatically control the interval of creating LogShipper tasks and configure the maximum size of an OSS object (not compressed).	The value range is 5–256. The unit is MB.
Storage Format	The storage format after log data is shipped to OSS.	Three formats are supported ( <a href="#">#unique_10</a> , <a href="#">#unique_11</a> , and <a href="#">#unique_12</a> ).

Configuration item	Description	Value range
Compression	The compression method of OSS data storage.	<ul style="list-style-type: none"><li>· Do Not Compress: The raw data is not compressed.</li><li>· Compress (snappy): Use <a href="#">snappy</a> algorithm to compress data, reducing the usage of OSS bucket storage space.</li></ul>

Configuration item	Description	Value range
Shipping Time	The time interval between LogShipper tasks.	The default value is 300. The value range is 300–900. . The unit is second.

Figure 3-1: Delivery log

\* OSS Shipping   
Name:

\* OSS Bucket:   
OSS Bucket name. The OSS Bucket and Log Service project should be in the same region.

OSS Prefix:   
Data synchronized from Log Service to OSS will be stored in this directory under the Bucket.

Partition Format:   
Generated by the log time. The default value is %Y/%m/%d/%H/%M, for example 2017/01/23/12/00. Note that the partition format cannot start or end with forward slash (/). For how to use with E-MapReduce (Hive/Impala), refer to [Help Link](#)

\* RAM Role:   
The RAM role created by the OSS Bucket owner for access control. For example, 'acs:ram:: 13234:role/logrole'.

\* Shipping Size:   
Automatically controls the creation interval of shipping tasks and sets the upper limit of the OSS object size (calculated in MBs according to the non-compressed data).

Figure 3-2: Role arn

AliyunLogDefaultRole

Basic information Edit Basic Information

Role Name AliyunLogDefaultRole Description -

Created At 2018-03-23 13:52:10 Arn

```
{
  "Statement": [
    {
      "Action": "sts:AssumeRole",
      "Effect": "Allow",
      "Principal": {
        "RAM": [
          "acs:ram::5204593714859318:root"
        ]
      }
    }
  ]
}
```



**Note:**

Log Service concurrently implements data shipping at the backend. Large amounts of data may be processed by multiple shipping threads. Each shipping thread jointly determines the frequency of task generation based on the size and time. When any condition is met, the shipping thread creates the task.

**Partition format**

Each LogShipper task is written into an OSS file, with the path format of `oss :// OSS - BUCKET / OSS - PREFIX / PARTITION - FROMAT_RANDOM - ID`. Use the LogShipper task created at 2017-01-20 19:50:43 as an example to describe how to use the partition format.

OSS Bucket	OSS Prefix	Partition format	OSS file path
test-bucket	test-table	%Y/%m/%d/%H/%M	oss :// test - bucket / test - table / 2017 / 01 / 20 / 19 / 50_1484913 0433515253 51_2850008
test-bucket	log_ship_oss_example	year=%Y/mon=%m/day=%d/log_%H%M%s	oss :// test - bucket / log_ship_oss_example / year = 2017 / mon = 01 / day = 20 / log_195043 _148491304 3351525351 _2850008 . parquet

OSS Bucket	OSS Prefix	Partition format	OSS file path
test-bucket	log_ship_oss_example	ds=%Y%m%d/%H	oss :// test - bucket / log_ship_oss_example / ds = 20170120 / 19_1484913 0433515253 51_2850008 . snappy
test-bucket	log_ship_oss_example	%Y%m%d/	oss :// test - bucket / log_ship_oss_example / 20170120 / _148491304 3351525351 _2850008
test-bucket	log_ship_oss_example	%Y%m%d%H	oss :// test - bucket / log_ship_oss_example / 2017012019 _148491304 3351525351 _2850008

Analyze the OSS data by using big data platforms such as Hive and MaxCompute.

To use the partition data, set each level of directory to key=value format (Hive-style partition).

For example, oss://test-bucket/log\_ship\_oss\_example/year=2017/mon=01/day=20/log\_195043\_1484913043351525351\_2850008.

parquet can be set to three levels of partition columns: year, month, and day.

### LogShipper tasks management

After the LogShipper function is enabled, Log Service regularly starts the LogShipper tasks in the backend. You can view the status of the LogShipper tasks in the console. With LogShipper tasks management, you can:

## View all the LogShipper tasks

- in the last two days and check their status. The status of a LogShipper task can be Success, Failed, and Running. The status Failed indicates that the LogShipper task has encountered an error because of external reasons and cannot be retried. In this case, you must manually solve the problem.
- For the failed LogShipper tasks created within two days, you can view the external reasons that cause the failure in the task list. After fixing the external errors, you can retry all the failed tasks separately or in batches.

## Procedure

1. Log on to the Log Service console.
2. On the Project List page, click the project name.
3. Select a Logstore, and click OSS in the left-side navigation pane.

You can view the information such as task start time, task end time, time when logs are received, data lines, and task status.

If the LogShipper task fails, a corresponding error message is displayed in the console. The system retries the task based on the policy by default. You can also manually retry the task.

## Retry a task

Generally, log data is synchronized to OSS within 30 minutes after being written to the Logstore.

By default, Log Service retries the tasks in the last two days based on the annealing policy. The minimum interval for retry is 15 minutes. A task that has failed once can be retried in 15 minutes, a task that has failed twice can be retried in 30 minutes (2 x 15 minutes), and a task that has failed three times can be retried in 60 minutes (2 x 30 minutes).

To immediately retry a failed task, click **Retry All Failed Tasks** in the console or specify a task and retry it by using APIs/SDKs.

## Failed tasks errors

See the following common errors that cause the task failure.

Error Message	Error cause	Handling method
Unauthorized	No permission.	Make sure that: - The OSS user has created a role. - The account ID in the role description is correct. - The role has been granted the permissions of writing OSS buckets. - The role-arn is correctly configured.
ConfigNotExist	The configuration does not exist.	This error is generally caused by the deletion of a shipping rule. Retry the task after reconfiguring the shipping rule.
InvalidOssBucket	The OSS bucket does not exist.	Make sure that: The OSS bucket is in the same region as the Log Service project. The bucket name is correctly configured
InternalServerError	The internal error of Log Service.	Retry the task.

## OSS data storage

You can access the OSS data in the console or by using APIs/SDKs.

To access OSS data in the console, log on to the OSS console, click a bucket name in the left-side navigation pane. For more information about OSS, see OSS documentation.

For more information about OSS, see OSS documentation.

## Object Address

```
oss :// OSS - BUCKET / OSS - PREFIX / PARTITION - FROMAT_RAN DOM - ID
```

- Descriptions of path fields
  - OSS-BUCKET and OSS-PREFIX indicate the OSS bucket name and directory prefix respectively, and are configured by the user. INCREMENTID is a random number added by the system.
  - PARTITION-FORMAT is defined as %Y/%m/%d/%H/%M, where %Y, %m, %d, %H, and %M indicate year, month, day, hour, and minute respectively. They are

obtained by using `strptime` API to calculate the created time of the LogShipper task in Log Service.

- RANDOM-ID is the unique identifier of a LogShipper task.
- Directory time

The OSS data directory is configured according to the created time of LogShipper tasks. Assume that the data is shipped to OSS every five minutes. The LogShipper task created at 2016-06-23 00:00:00 ships the data that is written to Log Service after 2016-06-22 23:55. To analyze the complete logs of the full day of 2016-06-22, in addition to all objects in the `2016 / 06 / 23 / 00 /` directory, you must check whether the objects in the first 10 minutes in the `2016/06/23/00/directory` contain the log of 2016-06-22.

#### Object storage format

- JSON

For more information, see [#unique\\_10](#).

- Parquet

For more information, see [#unique\\_11](#).

- CSV

For more information, see [#unique\\_12](#).

## 3.2 JSON storage

This document introduces the configurations about JSON storage for Log Service logs that are shipped to Object Storage Service (OSS). For more information about shipping logs to OSS, see [#unique\\_4](#).

The compression types and file addresses of OSS files are as follows.

Compression type	File suffix	Example of OSS file address
Do Not Compress	None	oss://oss-shipper-shenzhen/ecs_test/2016/01/26/20/54_1453812893059571256_937
snappy	.snappy	oss://oss-shipper-shenzhen/ecs_test/2016/01/26/20/54_1453812893059571256_937.snappy

#### Do Not Compress

An object is combined by multiple logs. Each line of the file is a log in the JSON format. See the following example:

```
{ " __time__ ": 1453809242 , " __topic__ ":" , " __source__ ":" 10 . 170
. ***.***" , " ip ":" 10 . 200 . **.*" , " time ":" 26 / Jan / 2016 :
19 : 54 : 02 + 0800 " , " url ":" POST
/ PutData ? Category = YunOsAccou ntOpLog &
AccessKeyId = < yourAccess KeyId > & Date = Fri % 2C % 2028 % 20Jun
% 202013 % 2006 % 3A53 % 3A30 % 20GMT & Topic = raw & Signature = <
yourSignature >
HTTP / 1 . 1 " , " status ":" 200 " , " user - agent ":"
aliyun - sdk - java " }
```

### Compress (snappy)

Use [Snappy C++](#) (Snappy.Compress method) to compress the data in none format at the file level. You can obtain the file in none format after decompress the .snappy file. You can obtain the file in none format after decompress the . snappy file.

### Decompressing through C++ Lib

Download Lib from [Snappy official website](#) and use the Snappy.Uncompress method to decompress the .snappy file.

### Java Lib

[xerial snappy-java], use Snappy.Uncompress or Snappy.SnappyInputStream (SnappyFramedInputStream not supported).

```
< dependency >
< groupId > org . xerial . snappy </ groupId >
< artifactId > snappy - java </ artifactId >
< version > 1 . 0 . 4 . 1 </ version >
< type > jar </ type >
< scope > compile </ scope >
</ dependency >
```



#### Note:

Version 1.1.2.1 may not decompress parts of the compressed file because of a [bug](#) , which is

### Snappy.Uncompress

```
String fileName = " C :\\ My download \\ 36_1474212 9631886006
84_4451886 . snappy ";
RandomAccessFile randomFile = new RandomAccessFile (
fileName , " r ");
int fileLength = ( int ) randomFile . length ();
randomFile . seek ( 0 );
byte [] bytes = new byte [ fileLength ];
int byteread = randomFile . read ( bytes );
System . out . println ( fileLength );
```

```
System . out . println ( bytread );
byte [] uncompress ed = Snappy . uncompress ( bytes );
String result = new String ( uncompress ed , " UTF - 8 " );
System . out . println ( result );
```

### Snappy.SnappyInputStream

```
String fileName = " C :\\ My download \\ 36_1474212 9631886006
84_4451886 . snappy ";
SnappyInput tStream sis = new SnappyInput tStream ( new
FileInputS tream ( fileName ));
byte [] buffer = new byte [ 4096 ];
int len = 0 ;
while (( len = sis . read ( buffer )) != - 1 ) {
    System . out . println ( new String ( buffer , 0 , len ));
}
```

### Unzipping tool under Linux environment

For Linux environment, a tool used to decompress .snappy file is provided. Click to download the [snappy\\_tool](#).

```
./ snappy_too l 03_1453457 0065480787 22_44148 . snappy
03_1453457 0065480787 22_44148
compressed . size : 2217186
snappy :: Uncompress return : 1
uncompress ed . size : 25223660
```

## 3.3 CSV storage

This document introduces the configurations about CSV storage for Log Service logs that are shipped to Object Storage Service (OSS). For more information about shipping logs to OSS, see [Ship logs to OSS](#).

### Configure CSV storage fields

#### Configuration page

You can view multiple key-value pairs of one log on the Log Service data preview page or index query page. Enter the field names (keys) you want to ship to OSS in sequence .

If the key name you entered cannot be found in the log, the corresponding column is set to null.

Figure 3-3: Configuration item

The screenshot shows a configuration window for shipping logs to OSS. It includes the following fields and options:

- Storage Format:** A dropdown menu set to "csv".
- CSV Keys:** A table with a "Name+" column and a "Delete" column. It contains five entries: "\_\_source\_\_", "\_\_time\_\_", "log\_key\_1", "log\_key\_2", and "log\_key\_3". Each entry has a corresponding "x" icon in the "Delete" column.
- How to use oss shipper to generate csv file?:** A link for more information.
- Delimiter:** A dropdown menu set to "Dot".
- Quote:** A dropdown menu set to ".".
- Invalid Key Value:** A text field containing "Used as value when specified key not exist, i".
- Display Key:** A toggle switch that is currently turned off. Below it, a note says: "Indicate whether generate key name in csv file, default is closed".
- Shipping Time:** A text field containing "300". Below it, a note says: "The time interval between shipping tasks. The unit is in seconds."

At the bottom right, there are two buttons: "Confirm" (in blue) and "Cancel" (in grey).

Configuration item

Configuration item	Value	Note
Delimiter	character	A one-character string used to separate different fields.



Configuration item	Value	Note
Quote	character	A one-character string. If a field contains a delimiter or a line break, use quote to enclose this field to avoid incorrect field separation in data reading.
Escape	character	A one-character string . The default settings are the same as those of quote. Modification is not supported currently. If a field contains a quote ( used as a regular character instead of an escape character), an escape character must be added before this quote.
Invalid Key Value	string	If the specified key value does not exist, this string is entered in the field to indicate the field is null.
Display Key header	boolean	Indicates whether or not to add the field name to the first line of the CSV file.

For more information, see [CSV standard](#) and [postgresql CSV description](#).

#### Configurable reserved fields

Besides the key-value pairs of the log, Log Service also provides the following optional reserved fields when shipping logs to OSS.

Reserved field	Description
<code>__time__</code>	The UNIX timestamp of a log (the number of seconds since 1970-01-01), which is calculated according to the time field of your log.
<code>__topic__</code>	The log topic.
<code>__source__</code>	The IP address of the client from which a log comes.

The preceding fields are included by default in JSON storage.

You can select which fields you want to include in the CSV storage as per your needs.

For example, you can enter the field name `__topic__` if you need the log topic.

#### OSS storage address

Compression type	File suffix	Example of OSS file address
Do Not Compress	.csv	oss://oss-shipper-shenzhen/ecs_test/2016/01/26/20/54_1453812893059571256_937.csv
snappy	.snappy.csv	oss://oss-shipper-shenzhen/ecs_test/2016/01/26/20/54_1453812893059571256_937.snappy.csv

#### Consume data

##### HybridDB

We recommend that you configure as follows:

- Delimiter: comma (,)
- Quote: double quotation marks ( “ )
- Invalid Key Value: empty
- Display Key: not selected (no field name in the first line of the CSV file for HybridDB by default)

For more information, see HybridDB document.

CSV is a readable format, which means that a file in CSV format can be directly downloaded from OSS and viewed in text form.

If Compress (snappy) is used as the compression type, see the decompression descriptions of snappy in [#unique\\_10](#).

## 3.4 Parquet storage

This document introduces the configurations about Parquet storage for Log Service logs that are shipped to Object Storage Service (OSS). For more information about shipping logs to OSS, see [Ship logs to OSS](#).

#### Configure Parquet storage fields

##### Data types

The Parquet supports the storage in six formats, including string, boolean, int32, int64, float, and double.

Log Service data will be converted from strings into the target Parquet type during log shipping. If any data fails to be converted into a non-string type, the corresponding column is filled with null.

#### Configure columns

Configure the Log Service data field names and the target data types required by Parquet. Parquet data is organized according to this field order when being shipped . The Log Service field names are used as the Parquet data column names. The data column is set to null if:

- This field name does not exist in Log Service data.

- This field fails to be converted from a string to a non-string (such as double and int64).

Figure 3-4: Field Configuration

\* Shipping Size:

Automatically controls the creation interval of shipping tasks and sets the upper limit of the OSS object size (calculated in MBs according to the non-compressed data).

\* Compression:

Compression method of OSS data storage. It can be none or snappy. None indicates that the original data is not compressed. Snappy indicates that the data is compressed using the snappy algorithm to reduce the OSS bucket storage being used.

\* Storage Format:

\* Parquet Key:

Name+	Type	Delete
<input type="text" value="key1"/>	<input type="text" value="string"/>	<input type="text" value="x"/>
<input type="text" value="key2"/>	<input type="text" value="float"/>	<input type="text" value="x"/>
<input type="text" value="key3"/>	<input type="text" value="int32"/>	<input type="text" value="x"/>

[How to use oss shipper to generate parquet file?](#)

\* Shipping Time:

The time interval between shipping tasks. The unit is in seconds.

### Configurable reserved fields

Besides the key-values of the log, the Log Service also provides the following optional reserved fields for the shipping to OSS:

Reserved field	Description
<code>__time__</code>	The UNIX timestamp of a log (the number of seconds since 1970-01-01), which is calculated according to the time field of your log.

Reserved field	Description
<code>__topic__</code>	The log topic.
<code>__source__</code>	The IP address of the client from which a log comes.

The preceding fields are carried by default in JSON storage.

You can select which fields you want to include in the Parquet or CSV storage as per your needs. For example, you can enter the field name `__topic__` and select string as the type if you need the log topic.

#### OSS storage address

Compression type	File suffix	Example of OSS file address
Do Not Compress	.parquet	oss://oss-shipper-shenzhen/ecs_test/2016/01/26/20/54_1453812893059571256_937.parquet
snappy	.snappy.parquet	oss://oss-shipper-shenzhen/ecs_test/2016/01/26/20/54_1453812893059571256_937.snappy.parquet

#### Consume data

E-MapReduce/Spark/Hive

See [community document](#).

Stand-alone verification tool

The [parquet-tools](#) provided by the open-source community is used to verify the Parquet format, view schema, and read data at the file level.

You can compile this tool by yourself or click [Download](#) to download the version provided by Log Service.

- View the schema of the Parquet file

```
$ java -jar parquet-tools-1.6.0rc3-SNAPSHOT.jar
schema -d 00_1490803 5321364704 39_124353 .snappy.parquet
| head -n 30
message schema {
  optional int32 __time__ ;
  optional binary ip ;
  optional binary __source__ ;
  optional binary method ;
  optional binary __topic__ ;
  optional double seq ;
```

```

    optional    int64    status ;
    optional    binary    time ;
    optional    binary    url ;
    optional    boolean   ua ;
}
creator : parquet - cpp    version    1 . 0 . 0
file    schema :    schema
-----
__time__ : OPTIONAL    INT32    R : 0    D : 1
ip : OPTIONAL    BINARY    R : 0    D : 1
.....

```

- View all contents of the Parquet file

```

$ java -jar parquet - tools - 1 . 6 . 0rc3 - SNAPSHOT . jar
  head - n    2    00_1490803  5321364704  39_124353 . snappy .
parquet
__time__ = 1490803230
ip = 10 . 200 . 98 . 220
__source__ = *. *. *. *
method = POST
__topic__ =
seq = 1667821 . 0
status = 200
time = 30 / Mar / 2017 : 00 : 00 : 30 + 0800
url = / PutData ? Category = YunOsAccou ntOpLog & AccessKeyI
d =*****& Date = Fri % 2C % 2028 % 20Jun % 202013
% 2006 % 3A53 % 3A30 % 20GMT & Topic = raw & Signature
=***** HTTP / 1 . 1
__time__ = 1490803230
ip = 10 . 200 . 98 . 220
__source__ = *. *. *. *
method = POST
__topic__ =
seq = 1667822 . 0
status = 200
time = 30 / Mar / 2017 : 00 : 00 : 30 + 0800
url = / PutData ? Category = YunOsAccou ntOpLog & AccessKeyI
d =*****& Date = Fri % 2C % 2028 % 20Jun % 202013
% 2006 % 3A53 % 3A30 % 20GMT & Topic = raw & Signature
=***** HTTP / 1 . 1

```

For more operation instructions, run the `java -jar parquet - tools - 1 . 6 . 0rc3 - SNAPSHOT . jar - h` command.

## 3.5 Snappy compression

You can set the compression method to [snappy](#) for logs that Log Service ships to Object Storage Service (OSS). Then, you can use the C++ library, Java library, Python library, and decompression tool for Linux to decompress these logs.

If you use the [snappy](#) algorithm to compress the logs to be shipped to OSS, these logs can occupy less bucket storage space in OSS. This topic describes the following methods for decompressing a .snappy file.

- [C++ library](#)
- [Java library](#)
- [Python library](#)
- [Decompression tool for Linux](#)

### C++ library

Download the C++ library from the [Snappy official website](#) and use the `Snappy.Uncompress` method to decompress a .snappy file.

### Java library

Download [xerial snappy-java](#) and use the `Snappy.Uncompress` or `Snappy.SnappyInputStream` method to decompress a .snappy file. The `SnappyFramedInputStream` method is not supported.



#### Note:

Due to an existing [bug](#), snappy-java 1.1.2.1 may fail to decompress some .snappy files. This bug has been fixed in snappy-java 1.1.2.6 and later versions. We recommend that you use the latest Java library.

```
< dependency >
< groupId > org . xerial . snappy </ groupId >
< artifactId > snappy - java </ artifactId >
< version > 1 . 0 . 4 . 1 </ version >
< type > jar </ type >
< scope > compile </ scope >
</ dependency >
```

- `Snappy.Uncompress`

```
String fileName = " C :\\ Downloads \\ 36_1474212 9631886006
84_4451886 . snappy ";
RandomAccessFile randomFile = new RandomAccessFile (
fileName , " r ");
int fileLength = ( int ) randomFile . length ();
randomFile . seek ( 0 );
byte [] bytes = new byte [ fileLength ];
int byteread = randomFile . read ( bytes );
System . out . println ( fileLength );
System . out . println ( byteread );
byte [] uncompressed = Snappy . uncompress ( bytes );
String result = new String ( uncompressed , " UTF - 8 " );
System . out . println ( result );
```

- `Snappy.SnappyInputStream`

```
String fileName = " C :\\ Downloads \\ 36_1474212 9631886006
84_4451886 . snappy ";
SnappyInputStream sis = new SnappyInputStream ( new
FileInputStream ( fileName ));
```

```
byte [] buffer = new byte [ 4096 ];
int len = 0 ;
while (( len = sis . read ( buffer )) != - 1 ) {
    System . out . println ( new String ( buffer , 0 , len ));
}
```

## Python library

1. Download and install [python-snappy](#).
2. Run the decompression code.

The sample decompression code is as follows:

```
import snappy
compressed = open ('/ tmp / temp . snappy '). read ()
snappy . uncompress ( compressed )
```



### Note:

The following commands cannot be used to decompress a .snappy file shipped to OSS. The two commands can be used only in Hadoop mode (hadoop\_stream\_decompress) or stream mode (stream\_decompress).

```
$ python -m snappy -c uncompress ed_file compressed
_file . snappy
$ python -m snappy -d compressed _file . snappy
uncompress ed_file
```

## Decompression tool for Linux

In the Linux environment, Log Service also provides a tool to decompress a .snappy file. Click to download [snappy\\_tool](#).

```
./ snappy_too l 03_1453457 0065480787 22_44148 . snappy
03_1453457 0065480787 22_44148
compressed . size : 2217186
snappy :: Uncompress return : 1
uncompress ed . size : 25223660
```

## 3.6 Advanced RAM authorization

Before perform the OSS shipping task, the owner of the OSS bucket must configure [quick authorization](#). After the authorization is complete, Log Service of the current account has the permission to write to OSS bucket.

This document describes the RAM authorization for OSS shipping tasks in different scenarios.



- If you need more fine-grained access control for OSS buckets, see [Modify the authorization policy](#).
- If a Log Service project and OSS bucket are not created with the same Alibaba Cloud account, see [Cross-account shipping](#).
- If a sub-account must ship log data to OSS bucket that belongs to another Alibaba Cloud account, see [Shipping between sub-account and main account](#).
- If a sub-account must ship log data of the current main account to the OSS bucket of the same account, see [#unique\\_9](#).

### Modify the authorization policy

After [quick authorization](#), the role AliyunLogDefaultRole is granted to AliyunLogRolePolicy by default, and has write permission for all OSS buckets of account B.

If you need more fine-grained access control, revoke the AliyunLogRolePolicy authorization from the AliyunLogDefaultRole. See *OSS authorization* to create a more fine-grained permission policy, and authorize the AliyunLogDefaultRole.

### Cross-account shipping

If your Log Service project and OSS bucket are not created with the same Alibaba Cloud account, you must configure the authorization policy in following way.

For example, Log Service data of the account A must be shipped to the OSS bucket created by the account B.

1. Using [quick authorization](#) account B creates the role AliyunLogDefaultRole, and grants write permission to OSS.
2. In the RAM console, click Role Management on the left-side navigation pane. Then, select AliyunLogDefaultRole, and click the role name to see the basic information.

In the role description, `Service` configuration indicates the legal user of the role. For example, `log . aliyuncs . com` indicates that the current account can obtain the role to get OSS write permission.

### 3. In `Service` configuration, you can modify the role description to add

`A_ALIYUN_ID @ log . aliyuncs . com`. ID of the main account A can be viewed in the Account Management > Security Settings.

For example, ID of the account A is 165421896534\*\*\*\*, and modified description is as follows:

```
{
  "Statement": [
    {
      "Action": "sts : AssumeRole ",
      "Effect": "Allow ",
      "Principal": {
        "Service": [
          "1654218965 34 ****@ log . aliyuncs . com ",
          "log . aliyuncs . com "
        ]
      }
    }
  ],
  "Version": " 1 "
}
```

This role description indicates that account A has the permission to use Log Service to obtain the temporary token to operate the resources of the account B. For more information about the role description, see [#unique\\_23](#).

### 4. The account A creates a shipping task. When configuring the task, RAM role column must be filled with the RAM role identifier ARN of the OSS bucket owner, that is, the RAM role `AliyunLogDefaultRole` created by account B.

The ARN of the RAM role can be viewed in the basic information. The format is as follows: `acs : ram :: 13234 : role / logrole`.

## Shipping between sub-account and main account

If the sub-account `a_1` of the main account A must use this role to create a shipping rule to ship logs to the OSS bucket of the account B. In this case, the main account A must grant the `PassRole` permission to the sub-account `a_1`.

The configuration is as follows:

1. Account B configures quick authorization and adds a description to the role. For more information, see [Cross-account shipping](#).

2. The main account A logs on to the RAM console and grants AliyunRAMFullAccess permission to the sub-account a\_1.
  - a. On the User Management page, click Authorization on the right side of the sub-account a\_1.
  - b. Search for AliyunRAMFullAccess in the authorizable policies, and add it to selected policies. Then click Confirm.

After successful authorization, a\_1 has all RAM permissions.

To control the permission range of a\_1, the main account A can grant a\_1 only the permissions required for shipping logs to OSS by modifying `Action` and `Resource` parameters.

The contents of the `Resource` must be replaced with the ARN of AliyunLogDefaultRole. The example of authorization policy is as follows:

```
{
  "Statement": [
    {
      "Action": "ram : PassRole ",
      "Effect": "Allow ",
      "Resource": "acs : ram :: 11111111 : role / aliyunlogd
        efaultrrole "
    }
  ],
  "Version": " 1 "
}
```

- c. The sub-account a\_1 creates a shipping task. When configuring the task, RAM role column must be filled with the RAM role identifier ARN of the OSS ARN of the OSS bucket owner, that is, the RAM role AliyunLogDefaultRole created by account B.

## 4 Ship data to MaxCompute

---

### 4.1 Ship data to MaxCompute by using DataWorks

You can not only ship logs to OSS storage, but also ship log data to MaxCompute by using the Data Integration function of DataWorks. Data Integration is a stable, efficient, and elastically scalable data synchronization platform provided by the Alibaba Group to external users. It provides offline batch data access channels for Alibaba Clouds big data computing engines (including MaxCompute, AnalyticDB, and OSPS).

For details about the regions in which this feature is available, see [DataWorks](#).

#### Scenarios

- Data synchronization between data sources (LogHub and MaxCompute) across regions
- Data synchronization between data sources (LogHub and MaxCompute) with different Alibaba Cloud accounts
- Data synchronization between data sources (LogHub and MaxCompute) with the same Alibaba Cloud account
- Data synchronization between data sources (LogHub and MaxCompute) with a public cloud account and an AntCloud account

#### Prerequisites

1. Log Service, MaxCompute, and DataWorks have been activated.
2. Log Service has successfully collected log data and LogHub has data to ship.
3. An Access Key pair is enabled for the data source account.
4. RAM authorization is configured when shipping across accounts is involved.

For details, see [Perform authorization for log shipping across accounts](#) in this document.

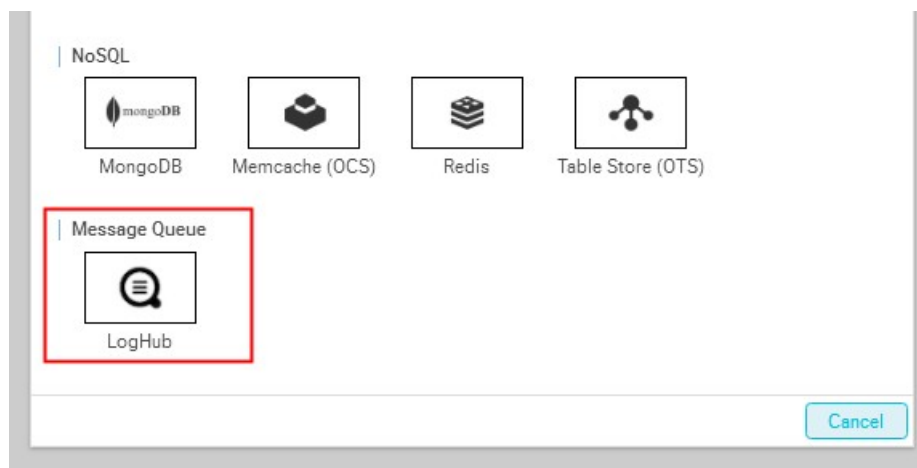
#### Procedure

##### Step 1 Create a data source

1. On the DataWorks console, open the Data Integration page. Click the Data Source tab on the navigation bar on the left.

2. On the Data Source page, click New Data Source in the upper right corner. The New Data Source page appears.
3. Click LogHub in the Message Queue list. The New LogHub data source page appears.

Figure 4-1: Add a data source



4. Set the configuration items for the data source.

The following table describes the configuration items:

Configuration items	Description
Data source name	A data source name may consist of letters, digits, and underscores. It must begin with a letter or underscore and cannot contain more than 60 characters.
Data source description	A brief description of the data source, containing up to 80 characters.
LOG Endpoint	Endpoint of Log Service, determined by your region, in the format of <code>http://yyy.com</code> . For more information, see <a href="#">#unique_26</a> .
LOG Project	A Log Service project in MaxCompute to which the log data is sent. It must be an existing project. Must be a project that has been created.

Configuration items	Description
Access Id/Access Key	The Access Key of the data source account is equivalent to a logon password. You can enter the Access Key of the primary account or subaccount of the data source. After successful configuration, the current account is granted access to the account logs in the data source and thus can ship logs of the data source account through a synchronization task.

Figure 4-2: Create a LogHub data source

5. Click Test Connectivity Click Finish after Connectivity test is successful appears in the upper right corner of the page.

## Step 2 Configure a synchronization task

Click Synchronization Task in the navigation bar on the left and click Step 2 Create a synchronization task to configure the synchronization task.

Select Wizard Mode to configure the task on a visualized page more easily; or select Script Mode to configure your synchronization task with more customization.

### Wizard mode

The configuration items of the task synchronization node include Select a Source, Select a Target, Field Mapping, and Channel Control.

## 1. Select a source.

**Data source:** Select the data source configured in Step 1. Set the configuration items according to the following table:

Configuration items	Description
Data source	Select the name of the LogHub data source.
Logstore	Name of the table from which the incremental data is exported. You must enable the Stream feature on the table when creating the table or using UpdateTable API later.
Log start time	Start time of data consumption. The parameter defines the left border of a time range (left closed and right open) in the format of yyyyMMddHHmmss (such as 20180111013000) and can work with the scheduling time parameter in DataWorks.
Log end time	End time of data consumption. The parameter defines the right border of a time range (left closed and right open) in the format of yyyyMMddHHmmss (such as 20180111013010) and can work with the scheduling time parameter in DataWorks.
Batch size	Number of data entries read each time. The default value is 256.

After the configuration items are set, click the Data Preview drop-down button to show the Data Preview details. Verify that log data has been obtained, and then click Next.



**Note:**

Data preview presents several data entries selected from the LogHub. The preview result may differ from the synchronization data that you configure, because the synchronization data is configured with log start time and end time.

Figure 4-3: Select a source

The screenshot shows a five-step configuration wizard. Step 1, 'Choose Source', is active and highlighted with a blue circle. The other steps are 'Select Target', 'Field Mapping', 'Channel Control', and 'Preview Stored'. Below the steps, a description reads: 'Reads data from a source data store. Viewing supported lists of [data source types](#)'. The configuration fields are as follows:

- \* data sources : docdoc (loghub) [dropdown arrow] [help icon]
- \* Logstore : wd2016 [dropdown arrow]
- \* log start time : \${startTime} [help icon]
- \* the end of the log time : \${endTime} [help icon]
- number of batch : 256 [help icon]

At the bottom, there is a 'data preview' link with a checkmark icon.



## 2. Select a target.

### a. Select a MaxCompute data source and target table.

If you have not created any MaxCompute table, click **Generate Target Table in One Click** on the right. Choose **Create Data Table** on the dialog box-up menu.

### b. Fill in Partition information.

Partition configuration supports regular expressions. For example, you can set the pt value of the partition “\*” to read data in all the pt partitions.

### c. Choose Clearing Rules.

You can choose to clear exiting data (overwrite mode) or retain existing data (insertion mode) before writing.

After the configuration, click **Next**.

Figure 4-4: Select a target

The screenshot shows a five-step progress bar at the top: 1. Choose Source, 2. Select Target (active), 3. Field Mapping, 4. Channel Control, and 5. Preview Stored. Below the progress bar, a text label reads: 'Writes data to the target/destination data store. Viewing support of [data type of destination](#)'. The configuration fields include: '\* data sources:' with a dropdown menu showing 'odps\_first (odps)'; '\* Table:' with a dropdown menu showing 'your\_table\_name' and a 'Create New Target Table' link; '\* Zoning Information:' with 'pt' in the first field, an equals sign, and a text box containing '\${bdp.system.bizdate}'; and 'Cleansing Rules:' with two radio buttons: 'Write before cleaning with available data Insert Overwrite' (selected) and 'Former reservations have been included in the data Insert Into'.

## 3. Map fields.

Select the mapping between fields. You need to configure the field mapping relationship. Source Table Fields on the left correspond one-to-one with Target Table Fields on the right. You can click **Same row mapping** to select or deselect **Same row mapping**.



### Note:

- If you need to manually add log fields as synchronous columns, use the [#unique\\_27/unique\\_27\\_Connect\\_42\\_section\\_gnq\\_k4f\\_vdb](#) configuration.
- You can enter constants. Each constant must be enclosed in a pair of single quotes, such as abc and 123.

- You can add scheduling parameters, such as `${bdp.system.bizdate}`.
- You can enter functions supported by relational databases, such as `now()` and `count(1)`.
- If the value you entered cannot be parsed, the type is displayed as Not identified
- 

Figure 4-5: Map fields



#### 4. Control the tunnel.

Configure the maximum job rate and dirty data check rules, as shown in the following figure:

Figure 4-6: Control the tunnel

#### Configuration item descriptions:

- **DMU: Data migration unit**, which measures the resources (including CPU, memory, and network bandwidth) 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 synchronization task.

## 5. Preview and save the configuration.

After completing the configuration, you can scroll up or down to view the task configurations. If no error exists, click Save.

Figure 4-7: Preview and save the configuration

1 Choose Source 2 Select Target 3 Field Mapping 4 Channel Control 5 Preview Stored

Please confirm and save the configured information, you can directly run or configure the scheduling properties, [data synchronization files](#)

choose source modified

- \* data sources : docdoc
- \* Logstore : wd2016
- \* log start time : \${startTime}
- \* the end of the log time : \${endTime}
- number of batch : 256

select target modified

- \* data sources : odps\_first
- \* Table: your\_table\_name
- \* Zoning Information: pt = \${bdp.system.bizdate}
- Cleansing Rules: ☒ Write before cleansing with available data Insert Overwrite

field mapping modified

Previous Save

## Script mode

To configure the task using a script, see the following script for reference:

```
{
  " type ": " job ",
  " version ": " 1 . 0 ",
  " configurat ion ": {
    " reader ": {
      " plugin ": " loghub ",
      " parameter ": {
        " datasource ": " loghub_lzz ",// Data source name . Use the
        data resource name you have added .
        " logstore ": " logstore - ut2 ",// Target Logstore name . A
        Logstore is a log data collection , storage , and query
        unit in the Log Service .
        " beginDateT ime ": "${ startTime }",// Start time of data
        consumptio n . The parameter defines the left border of
        a time range ( left closed and right open )
        " endDateTim e ": "${ endTime }",// End time of data
        consumptio n . The parameter defines the right border
        of a time range ( left closed and right open )
      }
    }
  }
}
```

```

" batchSize ": 256 ,// Number of data entries read each
time . The default value is 256 .
" splitPk ": "",
" column ": [
" key1 ",
" key2 ",
" key3 "
]
},
" writer ": {
" plugin ": " odps ",
" parameter ": {
" datasource ": " odps_first ",// Data source name . Use the
data resource name you have added .
" table ": " ok ",// Target table name
" truncate ": true ,
" partition ": "",// Shard informatio n
" column ": [// Target column name
" key1 ",
" key2 ",
" key3 "
]
}
},
" setting ": {
" speed ": {
" mbps ": 8 ,/ Maximum job rate
" concurrent ": 7 // Number of concurrent jobs
}
}
}
}
}

```

### Step 3 Run the task

You can run the task in either of the following ways:

- Directly run the task (one-time running)

Click Run above the task to run the task directly on the data integration page. Set values for the custom parameters before running the task.

Figure 4-8: Running task configuration

The screenshot shows a dialog box titled "running tasks configuration". It contains two sections. The first section, "system variable parameters", has a field "bdp.system.bizdate" with the value "20180531". The second section, "since the definition of variables and parameters", has two fields: "startTime" with the value "20180127101000" and "endTime" with the value "20180127103000". At the bottom right, there are "Ok" and "Cancel" buttons.

As shown in the preceding figure, LogHub records between 10:10 and 17:30 are synchronized to MaxCompute.

- Schedule the task

Click Submit to submit the synchronization task to the scheduling system. The scheduling system automatically and periodically runs the task from the second day according to the configuration attributes.

Set the schedule interval to 5 minutes, at a schedule period from 00:00 to 23:59, with `startTime=${yyyymmddhh24miss-10/24/60}` 10 minutes before the system to `endTime=${yyyymmddhh24miss-5/24/60}` 5 minutes before the system.

Figure 4-9: Scheduling configuration

The screenshot shows a 'Commit' dialog box with the following sections and fields:

- cycle attributes**
  - \* Movement Type : ☒ Cycle Control
  - \* Automatic Heavy ☐ Automatic Heavy Run ?
  - Run :
  - \* Date of Entry Into : 1970-01-01 - 2117-05-28
  - Force :
  - \* Scheduling Cycle : minutes hours **days** week month
  - \* The Starting And : 00:00
  - Ending Time:
- since the definition of variables and parameters ?**
  - startTime :
  - endTime :
- dependent attributes**
  - \* Add Dependent : dpdefault\_382549 please select the ...
- | Name of Project                 | The Mandate of The Name | Action |
|---------------------------------|-------------------------|--------|
| do not rely on upstream mandate |                         |        |

At the bottom right, there are 'Ok' and 'Cancel' buttons. The status bar at the bottom shows 'status=11'.

### Perform authorization for log shipping across accounts

To configure a log shipping task across accounts, perform authorization on the RAM.

- Perform authorization for log shipping across accounts

To ship data between primary accounts, you can enter the Access Key of the primary account of the data source in the step Add LogHub Data Source.

Authorization is successful if the connectivity test passes.

For example, to ship log data under account A to a MaxCompute table of account B through the DataWorks service activated with account B, configure a data integration task with account B and enter the Access Key of the primary account of account A in the step Add LogHub Data Source. After successful configuration, account B has the permission to read all log data under account A.

- Subaccount authorization

If you do not want to reveal the Access Key of the primary account or need to ship the log data collected by a subaccount, configure explicit authorization for the subaccount.

- Assign management permissions to the subaccount

If you need to ship all log data under a primary account through a subaccount, perform the following steps for authorization and Access Key configuration.

1. Use primary account A to assign Log Service management permissions ( AliyunLogFullAccess and AliyunLogReadOnlyAccess ) to subaccount A1. For details, see [#unique\\_9](#).
2. Configure a data integration task with account B, and enter the Access Key of the subaccount of the data source in the step Add LogHub Data Source.

After successful configuration, account B has the permission to read all log data under account A.

- Assign the customization permission to the subaccount

If you need to ship specified log data under a primary account through a subaccount, perform the following steps for authorization and Access Key configuration.

1. Configure a custom authorization policy for subaccount A1 with the primary account A. For details on related authorization operations, see [#unique\\_28](#) and [#unique\\_29](#).
2. Configure a data integration task with account B, and enter the Access Key of the subaccount of the data source in the step Add LogHub Data Source.

When the above steps are successfully completed, account B has the permission to read specified log data under account A.

Example of custom authorization policy:

In this way, account B can synchronize only project\_name1 and project\_name2 data in Log Service through subaccount A1.

```
{
  " Version ": " 1 ",
  " Statement ": [
    {
      " Action ": [
        " log : Get *",
```

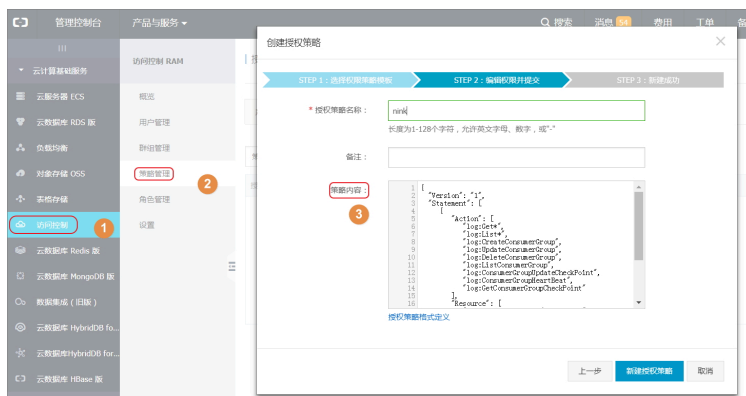


```

" log : List *",
" log : CreateCons umerGroup ",
" log : UpdateCons umerGroup ",
" log : DeleteCons umerGroup ",
" log : ListConsum erGroup ",
" log : ConsumerGr oupUpdateC heckPoint ",
" log : ConsumerGr oupHeartBe at ",
" log : GetConsume rGroupChec kPoint "
],
" Resource ": [
" acs : log :*: project / project_na me1 ",
" acs : log :*: project / project_na me1 /*",
" acs : log :*: project / project_na me2 ",
" acs : log :*: project / project_na me2 /*"
],
" Effect ": " Allow "
}
}
}

```

Figure 4-10: Custom authorization policy



## 5 Send logs to an SIEM system

---

### 5.1 Overview

This topic describes some of the requirements and features involved if you use the POST method to send logs to an SIEM system.

#### Terms

- **SIEM:** an abbreviation of Security Information and Event Management. It is a security management approach that can be implemented with Splunk and IBM QRadar.
- **Splunk HEC:** the Splunk HTTP Event Collector. It is an HTTP or HTTPS interface that is used to receive logs.

#### Recommended specifications for the target server

- **Hardware specifications**
  - Linux OS (for example, Ubuntu x64)
  - At least one 8-core CPU running a minimum of 2.0 GHz
  - At least 16 GB memory
  - At least one NIC that transmits 1 GB/s
  - At least one disk of 2 GB.



#### Note:

- Although 16 GB memory is the minimum specification required, we recommend that you upgrade to 32 GB of memory for better performance.
- Although 2 GB is the minimum storage specification required for the disk, we recommend that you upgrade to 10 GB storage for better performance.

- Network

The bandwidth value of the network that connects your server with Alibaba Cloud must be greater than the value of the rate at which data are generated in Alibaba Cloud. Otherwise, the generated data cannot be consumed in real time.

We recommend that you set the bandwidth to 4 MB/s (32 Mbps) for the scenario that meets the following conditions:

- Data are generated at an even rate.
- The peak rate of data generation is twice of the even rate.
- Log data of 1 TB are generated each day.
- The log data to be transmitted are compressed by a 5:1 compression ratio.

- Languages

Both Python and Java can be used to compile the required program.

In the two examples ([#unique\\_32](#) and [#unique\\_33](#)), Python is used.

For information about how the example in which Java is used, see [#unique\\_34](#).

## Consumer library

The consumer library provides the advanced mode to consume logs in Log Service. Specifically, the consumer library uses consumer groups to abstract and manage the data consumption. In contrast, SDK is used to directly read data. With the consumer library, you can focus on how the data is consumed and sent. You can ignore to the details about how Log Service works, or the load balancing and failover event between consumers.

One Logstore in Log Service contains multiple shards. The consumer Library assign shards to the consumers of one consumer group by following these rules:

- Each shard can only be assigned to one consumer.
- Multiple shards can be assigned to the same consumer.



### Note:

- These rules take effect each time when shards are assigned to consumers in a consumer group.

- When a new consumer is added in a consumer group, the shards assigned to the consumers before will be reassigned to all the consumers in the consumer group. This is a method of load balancing in the consumer group.
- The process in which shards are assigned to consumers is transparent.

In addition, the consumer library can also save check-points. This feature helps data consumption continue from the point where a failure occurred, avoiding re-consumption of the same data.

Spark Streaming, Storm, and Flink Connector all works on the base of the consumer library.

### Methods to send logs to an SIEM system

We recommend that you use consumer groups in Log Service to configure the required program to consume log data in real time, and then use send log data to an SIEM system through `HTTPS` or `Syslog`.

For more information, see [#unique\\_32](#) and [#unique\\_33](#).

## 5.2 Use HTTPS to send logs to an SIEM system

This topic uses an example to describe how to send Alibaba Cloud logs to an SIEM system (Splunk in this example) through the HTTP Event Collector (HEC).

### Prerequisites

- Splunk Enterprise Security is deployed on-premises to support the SIEM system, rather than in the cloud.
- This SIEM system cannot be accessed through the Internet because there are no corresponding interfaces configured for Internet access.
- PyPy3 is used to run the program, instead of CPython (the standard interpreter of Python).
- The Python SDK of Log Service is installed by using the `pypy3 -m pip install aliyun-log-python-sdk -U` command. For more information about Python SDK, see [User Guide](#).

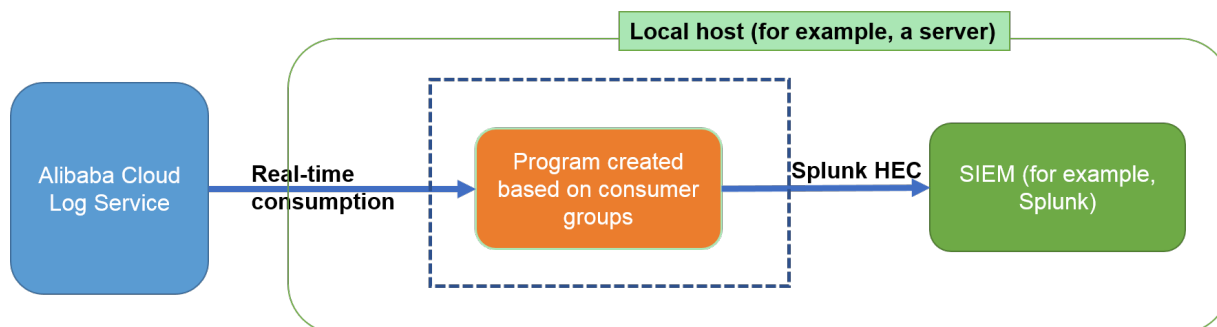


Note:

This topic uses a few of code examples. If you want to view all the latest code, you can obtain them at [Github](#), or [Github \(applicable to the Logstore that contains multiple data sources\)](#).

## Workflow

We recommend that you use the consumer groups in Log Service to configure the required program to consume log data in real time. Then, the program can send logs to the SIEM system (Splunk) by using the Splunk API (that is, HEC).



## Limits

For each Logstore in Log Service, up to ten consumer groups can be set. If the system displays the error message `ConsumerGroupQuotaExceed`, we recommend that you log on to the Log Service console to delete the consumer groups that you do not need.

## Control logic of the program

The following code shows the control logic of the program:

```

def main():
    option, settings = get_monitor_option()

    logger.info("*** start to consume data ...")
    worker = ConsumerWorker(SyncData, option, args=(
        settings,))
    worker.start(join=True)

if __name__ == '__main__':
  
```

```
main ()
```

## Configurations and a code example of the program

- Configurations
  - The log file of the program: This file is used to debug the program or diagnose any exceptions that may occur.
  - The basic configurations of the program: These include the configurations used to connect to Log Service and those of the consumer group.
  - The advanced configurations of the consumer group: These configurations are used to debug performance. We recommend that you do not modify them.
  - The parameters and options related to the SIEM (Splunk).
- A code example of the program

You must follow the annotations of the code to customize it to meet your needs.

```
# encoding : utf8
import os
import logging
from logging.handlers import RotatingFileHandler

root = logging.getLogger()
handler = RotatingFileHandler("{0}_{1}.log".format(
    os.path.basename(__file__), current_process().pid),
    maxBytes=100 * 1024 * 1024, backupCount=5)
handler.setFormatter(logging.Formatter(fmt='[% (asctime)s] - [% (threadName)s] - {%(module)s : %(funcName)s : %(lineno)d} %(levelname)s - %(message)s', datefmt='%Y-%m-%d %H:%M:%S'))
root.setLevel(logging.INFO)
root.addHandler(handler)
root.addHandler(logging.StreamHandler())

logger = logging.getLogger(__name__)

def get_option():
    #####
    # Basic options
    #####

    # Load parameters and options of SLS from
    # environment variables.
    endpoint = os.environ.get('SLS_ENDPOINT', '')
    accessKeyId = os.environ.get('SLS_AK_ID', '')
    accessKey = os.environ.get('SLS_AK_KEY', '')
    project = os.environ.get('SLS_PROJECT', '')
    logstore = os.environ.get('SLS_LOGSTORE', '')
    consumer_group = os.environ.get('SLS_CG', '')

    # This parameter specifies the starting time when
    # data consumption is initialized.
    # You can use begin or end when you set
    # this parameter. The time can be set to use a
    # specific ISO format.
    cursor_start_time = "2018-12-26 0:0:0"
```

```
#####
# Advanced options
#####

# Do not configure the consumer name , especially
when you need to run this program in parallel .
consumer_name = "{ 0 }-{ 1 }".format ( consumer_group ,
current_process (). pid )

# Set the heartbeat interval . If the server does
not receive the heartbeat report from the specified
shard within two times of the specified interval ,
the server determines the consumer is off line and
will reassign its task to another consumer .
# We recommend that you set a greater interval
when the network provides low performance .
heartbeat_interval = 20

# The maximum interval between two times of data
consumption . If data are generated quickly , do
not adjust this setting .
data_fetch_interval = 1

# Create a consumer group that contains a
consumer .
option = LogHubConfig ( endpoint , accessKeyId ,
accessKey , project , logstore , consumer_group , consumer_name ,
                           cursor_position = CursorPosition .
SPECIAL_TIMER_CURSOR ,
                           cursor_start_time = cursor_start_time
,
                           heartbeat_interval = heartbeat_
interval ,
                           data_fetch_interval = data_fetch
_interval )

# Splunk options
settings = {
    " host " : " 10 . 1 . 2 . 3 " ,
    " port " : 80 ,
    " token " : " a023nsdu12 3123123 " ,
    ' https ' : False ,
    # optional .
bool
    ' timeout ' : 120 ,
    ' ssl_verify ' : True ,
    " sourcetype " : "" ,
    " index " : "" ,
    " source " : "" ,
    # optional . int
    # optional . bool
    # optional .
    # optional .
}

return option , settings
```

An example of the code used to consume and send data

The following code shows the settings about for to obtain data from Log Service and how to send the data to the Splunk:

```
from aliyun . log . consumer import *
from aliyun . log . pulllog_response import PullLogResponse
```

```

from multiprocessing import current_process
import time
import json
import socket
import requests

class SyncData ( ConsumerProcessorBase ):
    """
    This consumer obtains data from Log Service and
    send the data to Splunk .
    """
    def __init__ ( self , splunk_setting ):
        """ Initiate the Splunk server and its network
        connectivity ."""
        super ( SyncData , self ). __init__ ()

        assert splunk_setting , ValueError ( " You need to
        configure settings of remote target " )
        assert isinstance ( splunk_setting , dict ), ValueError
        ( " The settings should be dict to include necessary
        address and credentials . " )

        self . option = splunk_setting
        self . timeout = self . option . get ( " timeout " , 120 )

        # Test Splunk connectivity .
        s = socket . socket ()
        s . settimeout ( self . timeout )
        s . connect (( self . option [ " host " ] , self . option [ '
port ' ]))

        self . r = requests . session ()
        self . r . max_redirects = 1
        self . r . verify = self . option . get ( " ssl_verify " ,
True )
        self . r . headers [ ' Authorization ' ] = " Splunk {}".
format ( self . option [ ' token ' ])
        self . url = "{ 0 }://{ 1 }:{ 2 }/ services / collector /
event ". format ( " http " if not self . option . get ( ' https
' ) else " https " , self . option [ ' host ' ] , self . option [ '
port ' ])

        self . default_fields = {}
        if self . option . get ( " sourcetype " ):
            self . default_fields [ ' sourcetype ' ] = self . option
. get ( " sourcetype " )
        if self . option . get ( " source " ):
            self . default_fields [ ' source ' ] = self . option .
get ( " source " )
        if self . option . get ( " index " ):
            self . default_fields [ ' index ' ] = self . option .
get ( " index " )

        def process ( self , log_groups , checkpoint_tracker ):
            logs = PullLogResponse . loggroups_to_flatten_list (
log_groups , time_as_string = True , decode_bytes = True )
            logger . info ( " Get data from shard { 0 } , log
count : { 1 } ". format ( self . shard_id , len ( logs )))
            for log in logs :
                # Send data to Splunk .
                event = {}
                event . update ( self . default_fields )
                event [ ' time ' ] = log [ u ' __time__ ' ]
                del log [ ' __time__ ' ]

```



```

    json_topic = {"actiontrail_audit_event": ["event
"] }
    topic = log.get("__topic__", "")
    if topic in json_topic:
        try:
            for field in json_topic[topic]:
                log[field] = json.loads(log[field
])
            except Exception as ex:
                pass
            event['event'] = json.dumps(log)
            data = json.dumps(event, sort_keys=True)

            try:
                req = self.r.post(self.url, data =
data, timeout = self.timeout)
                req.raise_for_status()
            except Exception as err:
                logger.debug("Failed to connect to
remote Splunk server ({0}). Exception: {1}", self.url
, err)

            # TODO: Add the code to process
errors. For example, you can add the code to retry
or send a notification in response to an error.
            raise err

            logger.info("Complete send data to remote")
            self.save_checkpoint(checkpoint_tracker)

```

A example of the code used to start the program

Assume that the program is named `sync_data.py`. The code used to start the program is as follows:

```

export SLS_ENDPOINT =< Endpoint of your region >
export SLS_AK_ID =< YOUR AK ID >
export SLS_AK_KEY =< YOUR AK KEY >
export SLS_PROJECT =< SLS Project Name >
export SLS_LOGSTORE =< SLS Logstore Name >
export SLS_CG =< SLS consumer group name. You can set
it to sync_data.>

pypy3 sync_data.py

```

An example code of the Logstore that contains multiple sources

For the Logstore that contains multiple sources, you must set an executor to avoid process broken down. For more information about related examples, see [Send logs from a multi-source Logstore to Splunk](#).



**Note:**

**You must pay attention to the changes to the primary functions.**

```

executor , options , settings = get_option ()

    logger . info ("*** start to consume data ...")
    workers = []

    for option in options :
        worker = ConsumerWorker ( SyncData , option , args =(
settings ,) )
        workers . append ( worker )
        worker . start ()

    try :
        for i , worker in enumerate ( workers ) :
            while worker . is_alive () :
                worker . join ( timeout = 60 )
                logger . info (" worker project : { 0 } logstore : { 1
} exit unexpected , try to shutdown it ". format (
options [ i ]. project , options [ i ]. logstore ))
                worker . shutdown ()
            except KeyboardInterrupt :
                logger . info ("*** try to exit **** ")
                for worker in workers :
                    worker . shutdown ()

        # wait for all workers to shutdown before
        shutting down executor
        for worker in workers :
            while worker . is_alive () :
                worker . join ( timeout = 60 )

    executor . shutdown ()

```

### View and monitor data consumption

- You can log on to the Log Service console to view the status of a consumer group. For more information, see [#unique\\_36](#).
- You can log on to the CloudMonitor to view delays associated with consumer groups and configure corresponding alarms. For more information, see [Consumer group delay](#).

### Run multiple consumers

For a program that is based on consumer groups, you can start the program multiple times to run multiple consumers.

```

nohup pypy3 sync_data . py &
nohup pypy3 sync_data . py &
nohup pypy3 sync_data . py &
...

```



**Note:**

The data of one shard can be consumed by only one consumer. If a Logstore contains ten shards, then up to ten consumers can consume the data of all of the shards.

### Throughput test scenario

Assume that you have an instance with a single core CPU. The data of raw logs is consumed at a rate of 5 MB/s only if the following conditions are met:

- The bandwidth and rate to receive data is not limited.
- PyPy3 is used to run the following example codes.
- One consumer uses about 10% resources of a single-core CPU.

Therefore, ten consumers that use the same single-core CPU can consume the data of raw logs at the rate of 50 MB/s. Then, up to 4 TB data of raw logs can be consumed on such a CPU.

### High availability scenario

A consumer group saves check-points on the server end. When one consumer stops, another consumer automatically takes over the data consumption task and continues the task from the check-point where the preceding consumer stopped.

Therefore, to achieve high availability, start multiple consumers that are located in different servers. That way, if one of these servers stops or is damaged, a consumer located in one of the other servers can take over the consumption task and continue the task from the check-point where the preceding consumer stopped.

To ensure sufficient consumers are available, set the number of the consumers that you start in different servers to be greater than the number of the shards.

### HTTPS

To use HTTPS to encrypt the data transmitted between your configured program and Log Service, you must set the prefix of the service endpoint to `https ://`. For example, `https :// cn - beijing . log . aliyuncs . com`.

The server certificate \*.aliyuncs.com issued by GlobalSign. Generally, most Linux or Windows server automatically trusts this certificate. If a server does not trust this certificate, you can download the valid certificate at [Certificates](#).

## 5.3 Use Syslog to send logs to an SIEM system

This topic describes the required configurations and related information for how to use Syslog to send logs to your Security Information and Event Management (SIEM) software. Syslog is a log path through which most physical devices (such as switches, routers, and servers) send logs. In addition, almost SIEM software (including IBM Qradar and HP Arcsight) can receive logs that are sent with Syslog.

### Background information

- Syslog: This protocol defines the standard format of logs according to the standard protocols [RFC5424](#) and [RFC3164](#).

The RFC3164 protocol was released in 2001 and the RFC5424 protocol released in 2009 is an upgraded version. We recommend that you use the RFC5424 protocol. It has fixed many of the issues of the previous version and is compatible with the RFC3164 protocol.

- Syslog over TCP/TLS: If the TCP and TLS protocols are supported by your SIEM software, this method can be used to transfer data to your software. We recommend that you use this method.

Both the TCP and UDP protocols support Syslog. These two protocols can guarantee the stable data transmission. The [RFC5425](#) protocol also defines the TLS protocol.

- Syslog facility: This is a program component defined by the earlier versions of Unix. For more information, see [Facility](#). You can select `user - level messages` as the default facility.
- Syslog severity: This refers to [Message priorities](#). You can set the priority for the specified logs as needed. By default, `Informational : informational messages` is used.



#### Note:

This topic uses a few of code examples. If you want to view all the latest code, you can obtain them at [Github](#).

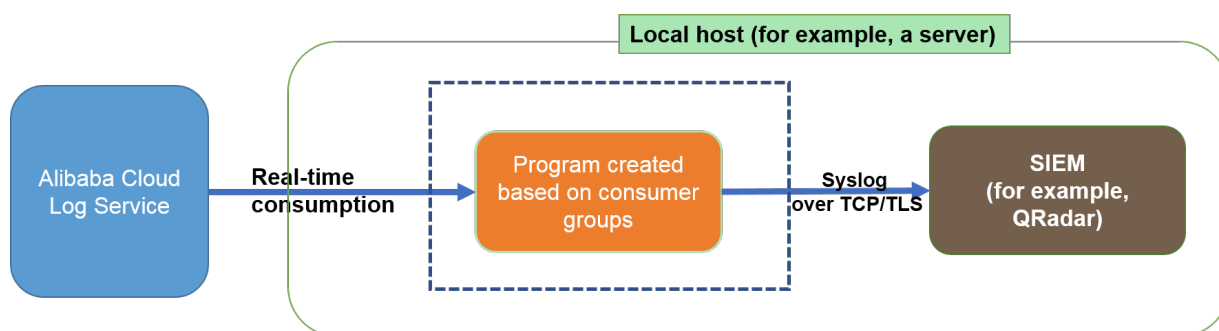
### Prerequisites

- PyPy3 is used to run the program, instead of CPython (the standard interpreter of Python).

- The Python SDK of Log Service is installed by using the `python3 -m pip install aliyun-log-python-sdk -U` command. For more information about Python SDK, see [User Guide](#).

## Workflow

We recommend that you configure the required program that is based on the consumer groups in Log Service. Then, the program can send logs to the SIEM system by using Syslog.



## Limits

Up to ten consumer groups can be set for each Logstore in Log Service. If the system displays the error message `ConsumerGroupQuotaExceed`, we recommend that you log on to the Log Service console to delete the consumer groups that you do not need.

## Control logic of the program

The following code shows the control logic of the program:

```
def main():
    option, settings = get_monitor_option()

    logger.info("*** start to consume data ...")
    worker = ConsumerWorker(SyncData, option, args=(
        settings,))
    worker.start(join=True)

if __name__ == '__main__':
```

```
main ()
```

## Configurations and a code example of the program

- Configurations
  - The log file of the program: This file is used to debug the program or diagnose any exceptions that may occur.
  - The basic configurations of the program: These include the configurations used to connect to Log Service and those of the consumer group.
  - The advanced configurations of the consumer group: These configurations are used to debug performance. We recommend that you do not modify them.
  - The parameters and options related to the Syslog server.



### Note:

If SIEM supports the Syslog path that is based on TCP or TLS, set the `proto` parameter of the program to TLS and set a valid SSL certificate.

- A code example of the program

You must follow the annotations of the code to customize it to meet your needs.

```
# encoding : utf8
import os
import logging
from logging.handlers import RotatingFileHandler

root = logging.getLogger()
handler = RotatingFileHandler("{0}_{1}.log".format(
    os.path.basename(__file__), current_process().pid),
    maxBytes = 100 * 1024 * 1024, backupCount = 5)
handler.setFormatter(logging.Formatter(
    fmt='[% (asctime)s] - [% (threadName)s] - [% (module)s : % (funcName)s : % (
    lineno)d] % (levelname)s - % (message)s ', datefmt='%Y-%m-%d %H:%M:%S'))
root.setLevel(logging.INFO)
root.addHandler(handler)
root.addHandler(logging.StreamHandler())

logger = logging.getLogger(__name__)

def get_option():
    #####
    # Basic options
    #####

    # Load parameters and options of SLS from
    environment variables.
    endpoint = os.environ.get('SLS_ENDPOINT', '')
    accessKeyId = os.environ.get('SLS_AK_ID', '')
    accessKey = os.environ.get('SLS_AK_KEY', '')
    project = os.environ.get('SLS_PROJECT', '')
    logstore = os.environ.get('SLS_LOGSTORE', '')
```

```

consumer_group = os.environ.get('SLS_CG', '')

# This parameter specifies the starting time when
data consumption is initialized.
# You can use begin or end when you set
this parameter. The time can be set to use a
specific ISO format.
cursor_start_time = "2018-12-26 0:0:0"

#####
# Advanced options
#####

# Do not configure the consumer name, especially
when you need to run this program in parallel.
consumer_name = "{0}-{1}".format(consumer_group,
current_process().pid)

# Set the heartbeat interval. If the server does
not receive the heartbeat report from the specified
shard within two times of the specified interval,
the server determines the consumer is off line and
will reassign its task to another consumer.
# We recommend that you set a greater interval
when the network provides low performance.
heartbeat_interval = 20

# The maximum interval between two times of data
consumption. If data are generated quickly, do
not adjust this setting.
data_fetch_interval = 1

# Create a consumer group that contains a
consumer.
option = LogHubConfig(endpoint, accessKeyId,
accessKey, project, logstore, consumer_group, consumer_name,
                        cursor_position = CursorPosition.SPECIAL_TIMER_CURSOR,
                        cursor_start_time = cursor_start_time,
                        heartbeat_interval = heartbeat_interval,
                        data_fetch_interval = data_fetch_interval)

# syslog options
settings = {
    "host": "1.2.3.4", # required.
    "port": 514, # required.
    "protocol": "tcp", # required. Valid: tcp
| udp | tls (only applicable to Python3)
    "sep": "||", # required. Set the
separator to separate key-value pairs.
    "cert_path": None, # optional. Set the
directory where the TLS certificate is located.

    "timeout": 120, # optional. Set the
timeout. The default timeout is 120 seconds.
    "facility": syslogclient.FAC_USER, #
optional. You can refer to the values of other
syslogclient.FAC_*.

```

```

        " severity ": syslogclie nt . SEV_INFO , #
optional . You can refer to the values of other
syslogclie nt . SEV_* .
        " hostname ": None , # optional . Set the
host name . By default , the name of the host that
you use is used .
        " tag ": None # optional . Set the
tag . By default , a hyphen (-) is used .
    }

    return option , settings

```

An example of the code used to consume and send data

The following code shows the settings about for to obtain data from Log Service and how to send the data to the Syslog server:



#### Note:

You can customize the format of the code according to your needs.

```

from syslogclie nt import SyslogClie ntRFC5424 as
SyslogClie nt

class SyncData ( ConsumerPr ocessorBas e ):
    """
    This consumer obtains data from Log Service and
    send the data to the Syslog server .
    """
    def __init__ ( self , splunk_set ting ):
        """ Initiate the Syslog server and its network
        connectivi ty . """
        super ( SyncData , self ). __init__ () # remember to
        call base ' s ' init

        assert target_set ting , ValueError ( " You need to
        configure settings of remote target " )
        assert isinstance ( target_set ting , dict ) , ValueError
        ( " The settings should be dict to include necessary
        address and confidenti als . " )

        self . option = target_set ting
        self . protocol = self . option [ ' protocol ' ]
        self . timeout = int ( self . option . get ( ' timeout ' ,
120 ))
        self . sep = self . option . get ( ' sep ' , " || " )
        self . host = self . option [ " host " ]
        self . port = int ( self . option . get ( ' port ' , 514 ))
        self . cert_path = self . option . get ( ' cert_path ' , None
        )

        # try connection
        with SyslogClie nt ( self . host , self . port , proto
= self . protocol , timeout = self . timeout , cert_path = self .
cert_path ) as client :
            pass

        def process ( self , log_groups , check_poin t_tracker ):
            logs = PullLogRes ponse . loggroups_ to_flatter n_list (
            log_groups , time_as_st r = True , decode_byt es = True )

```



```

        logger . info (" Get  data  from  shard { 0 }, log
count : { 1 }". format ( self . shard_id , len ( logs )))
    try :
        with SyslogClient ( self . host , self . port ,
proto = self . protocol , timeout = self . timeout , cert_path =
self . cert_path ) as client :
            for log in logs :
                # Put your sync code here to send
to remote .
                # the format of log is just a
dict with example as below ( Note , all strings are
unicode ) :
                # Python2 : { " __time__ " : " 12312312 " , "
__topic__ " : " topic " , u " field1 " : u " value1 " , u " field2 " :
u " value2 " }
                # Python3 : { " __time__ " : " 12312312 " , "
__topic__ " : " topic " , " field1 " : " value1 " , " field2 " : " value2
"}
                # suppose we only care about audit
log
                timestamp = datetime . fromtimestamp ( int (
log [ u ' __time__ ' ]))
                del log [ ' __time__ ' ]

                io = six . StringIO ()
                first = True
                # TODO : Modify the content to be formatted
as needed . The data is transmitted by using key -
value pairs which are separated by ||.
                for k , v in six . iteritems ( log ) :
                    io . write (" { 0 } { 1 } = { 2 } ". format ( self
. sep , k , v ))

                data = io . getvalue ()

                # TODO : Modify the facility or severity as
needed .
                client . log ( data , facility = self . option
. get ( " facility " , None ) , severity = self . option . get ( "
severity " , None ) , timestamp = timestamp , program = self .
option . get ( " tag " , None ) , hostname = self . option . get ( "
hostname " , None ))

    except Exception as err :
        logger . debug (" Failed to connect to remote
syslog server ( { 0 } ). Exception : { 1 } ". format ( self . option
, err ))

        # TODO : Add the code to process errors . For
example , you can add the code to retry or send
a notification in response to an error .
        raise err

    logger . info (" Complete send data to remote ")

```

```
self . save_check_point ( check_poin t_tracker )
```

A example of the code used to start the program

Assume that the program is named `sync_data . py` . The code used to start the program is as follows:

```
export SLS_ENDPOINT = < Endpoint of your region >
export SLS_AK_ID = < YOUR AK ID >
export SLS_AK_KEY = < YOUR AK KEY >
export SLS_PROJECT = < SLS Project Name >
export SLS_LOGSTORE = < SLS Logstore Name >
export SLS_CG = < SLS consumer group name . You can set it to sync_data .>

pypy3 sync_data . py
```

View and monitor data consumption

- You can log on to the Log Service console to view the status of a consumer group. For more information, see [#unique\\_36](#).
- You can log on to the CloudMonitor to view delays associated with consumer groups and configure corresponding alarms. For more information, see [Consumer group delay](#).

Run multiple consumers

For a program that is based on consumer groups, you can start the program multiple times to run multiple consumers.

```
nohup pypy3 sync_data . py &
nohup pypy3 sync_data . py &
nohup pypy3 sync_data . py &
...
```



#### Note:

The data of one shard can be consumed by only one consumer. If a Logstore contains ten shards, then up to ten consumers can consume the data of all of the shards.

Throughput test scenario

Assume that you have an instance with a single core CPU. The data of raw logs is consumed at a rate of 5 MB/s only if the following conditions are met:

- No limit of the bandwidth and rate to receive data exists.
- PyPy3 is used to run the following example codes.
- One consumer uses about 10% resources of a single-core CPU.

Therefore, ten consumers that use the same single-core CPU can consume the data of raw logs at the rate of 50 MB/s. Then, up to 4 TB data of raw logs can be consumed on such a CPU.

#### High availability scenario

A consumer group saves check-points on the server end. When one consumer stops, another consumer automatically takes over the data consumption task and continues the task from the check-point where the preceding consumer stopped.

Therefore, to obtain high availability, you can start multiple consumers that are located in different servers. That way, if one of these servers stops or is damaged, a consumer located in one of the other servers can take over the consumption task and continue the task from the check-point where the preceding consumer stopped.

To ensure sufficient consumers are available, set the number of the consumers that you start in different servers to be greater than the number of the shards.