ALIBABA CLOUD

Alibaba Cloud

Log Service

Data Transformation

Document Version: 20220708

Alibaba Cloud

Legal disclaimer

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

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

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

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

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

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

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

Document conventions

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

Table of Contents

1.Data transformation overview

Data transformation is a fully managed feature that provides high availability and scalability in Log Service. You can use the data transformation feature to standardize, enrich, transfer, mask, and filter data.

Transformation process

Log Service transforms data in the following three steps:

1. A consumer group reads data from a source Logstore.
2. Log Service transforms each data entry based on a transformation rule.
3. Log Service writes transformed data to a destination Logstore.

   After the data is transformed, you can view the data in the destination Logstore.

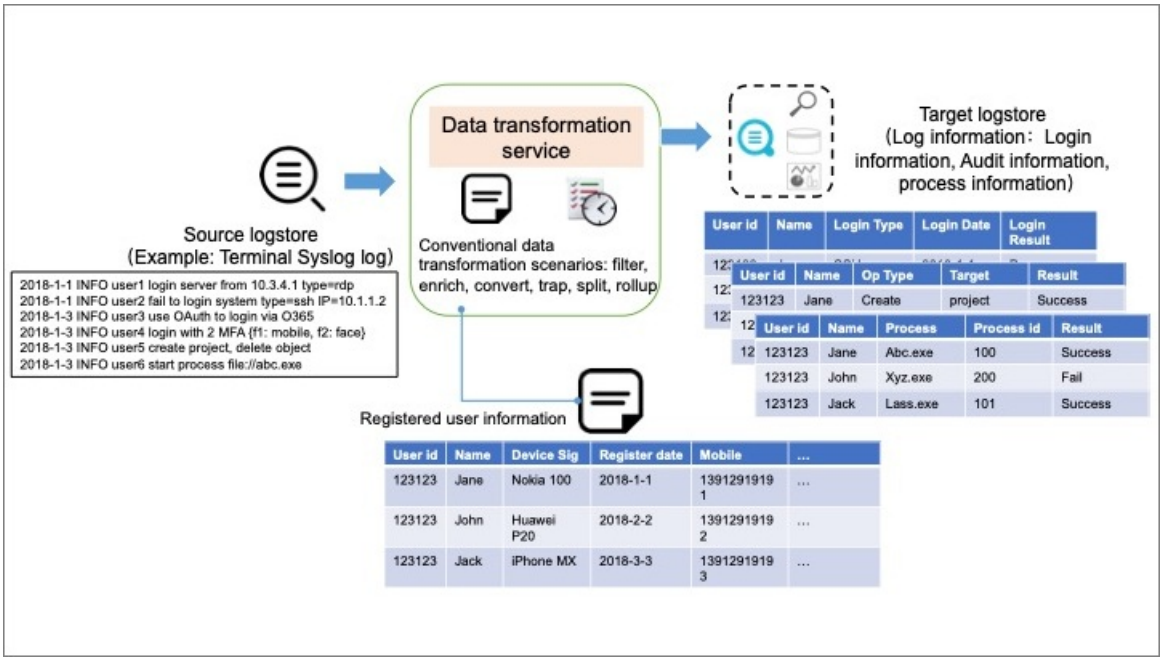
Features

You can use the data transformation feature to standardize, enrich, transfer, mask, and filter data.

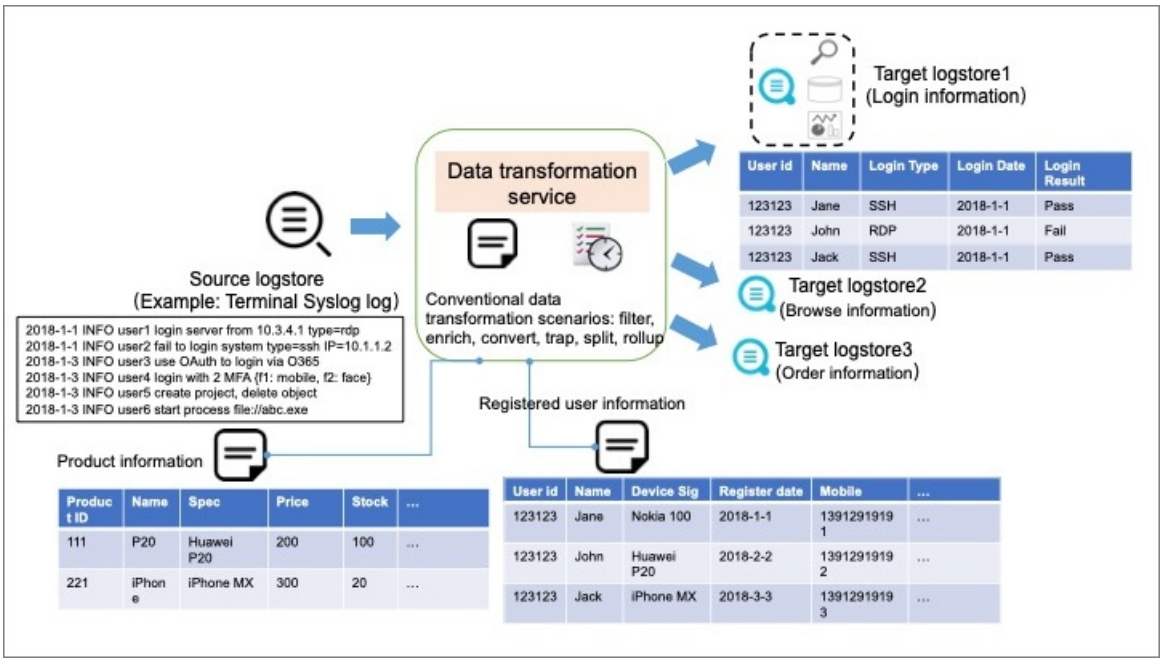
- Data standardization: You can extract fields from different formats of logs and convert the logs into structured data for stream processing and computing in data warehouses.
- Data enrichment: You can join the fields of logs and the fields of dimension tables to add dimensions for data analysis. For example, you can join order logs and user information tables to analyze data.
- Data transfer: You can transfer logs from regions outside mainland China to one region by using the global acceleration feature. This way, global logs can be managed in a centralized manner.
- Data masking: You can mask the sensitive information of data, such as passwords, mobile phone numbers, and addresses.
- Data filtering: You can filter the logs of specific services for further analysis.

Scenarios

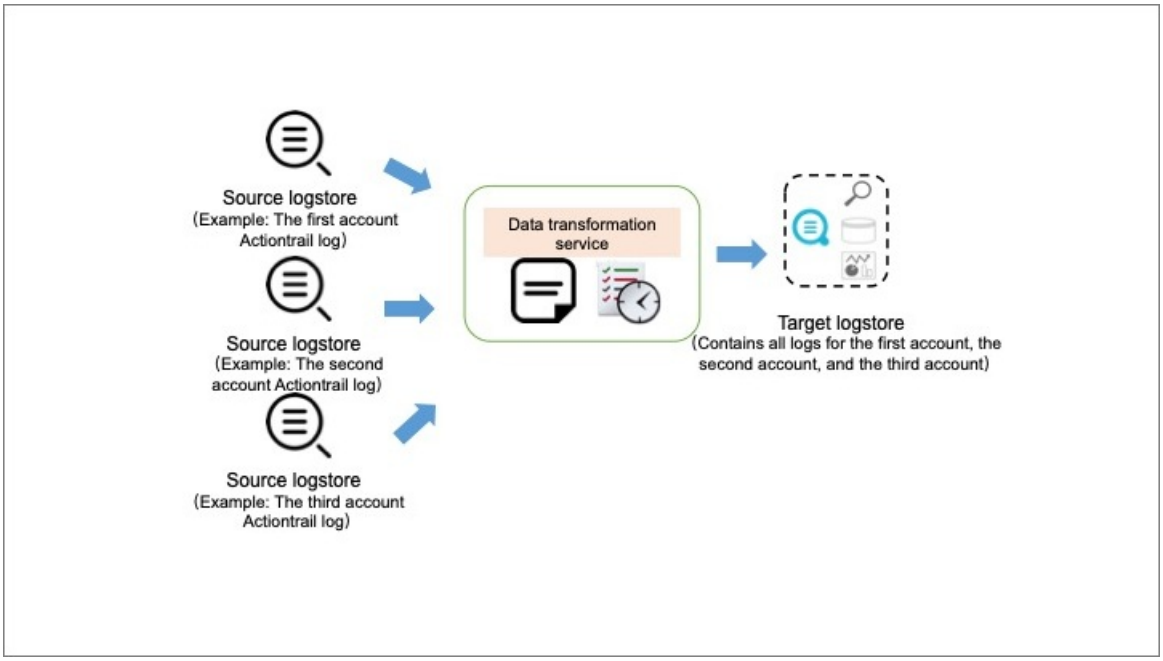
- Data standardization: Log data is read from a source Logstore, transformed, and then written to a destination Logstore.



- Data transfer: Log data is read from a source Logstore, transformed, and then written to multiple destination Logstores.



- Multi-source data aggregation: Log data is read from multiple source Logstores, transformed, and then written to a destination Logstore.



Transformation syntax

Log Service provides more than 200 built-in functions and more than 400 regular expressions. You can also use the domain-specific language (DSL) for Log Service to create user-defined functions (UDFs) based on your business requirements. For more information, see Syntax introduction.

Benefits

- Allows you to use the DSL for Log Service to orchestrate functions as needed. You can use the orchestrated functions to filter, standardize, enrich, transfer, and mask data.
- Processes data in real time and allows you to view data within seconds. The feature scales in or out the computation capability based on the size of data and provides a high throughput.
- Applies to log analysis scenarios and provides out-of-the-box functions.
- Integrates with dashboards, exception logs, and alerts in real time.
- Offers a fully managed and maintenance-free service that can be integrated with the big data services of Alibaba Cloud and open source ecosystems.

Billing

- You are charged for the server and network resources that are consumed when you use the data transformation feature. For more information, see Billable items.
- To reduce costs, you can disable the indexing feature for a source Logstore and set a short data retention period. For more information, see Performance guide and Cost optimization guide.

2.Terms

This topic introduces the terms that are related to the data transformation feature.

Basic terms

- ETL

  Extract, transform, and load (ETL) is a process during which data is extracted from business systems, cleansed, transformed, and loaded. This process unifies and standardizes data from different sources. Log Service can load data from a source Logstore, transform data, and then write transformed data to destination Logstores. Log Service can also load data from Object Storage Service (OSS) buckets, ApsaraDB RDS instances, or other Logstores.

- event, data, and log

  In data transformation, events and data are represented by logs. For example, the event time is equivalent to the log time, and the `drop_event_fields` function discards log fields.

- log time

  The log time indicates the point in time at which an event occurs. The log time is also known as the event time. The log time is indicated by the reserved field `__time__` in Log Service. The value of this field is extracted from the time information in logs. The value is a UNIX timestamp representing the number of seconds that have elapsed since the epoch time January 1, 1970, 00:00:00 UTC. Data type: integer. Unit: seconds.

- log receiving time

  The log receiving time indicates the point in time at which a log is received by a server of Log Service. By default, this time is not saved in logs. However, if you turn on Log Public IP for a Logstore, this time is recorded in the log tag field `__receive_time__`. In the data transformation process, the complete name of this field is `__tag__:__receive_time__`. The value is a UNIX timestamp representing the number of seconds that have elapsed since the epoch time January 1, 1970, 00:00:00 UTC. Data type: integer. Unit: seconds.

  > **Note** In most scenarios, logs are sent to Log Service in real time, and the **log time** is the same as the **log receiving time**. If you import historical logs, the log time is different from the log receiving time. For example, if you import logs generated during the last 30 days by using an SDK, the **log receiving time** is the current time and is different from the **log time**.

- tag

  Logs have tags. Each tag field is prefixed with `__tag__:`. Log Service supports two types of tags.

  - Custom tags: the tags that you add when you call the PutLogs operation to write data.
  - System tags: the tags that are added by Log Service, including `__client_ip__` and `__receive_time__`.

Configuration-related terms

- source Logstore

  The data transformation feature reads data from a source Logstore for transformation.

You can configure only one source Logstore for a data transformation task. However, you can configure the same source Logstore for different data transformation tasks.

- destination Logstore

  The data transformation feature writes transformed data to destination Logstores.

  You can configure one or more destination Logstores for a data transformation task. Data can be written to destination Logstores in static or dynamic mode. For more information, see Distribute data to multiple destination Logstores.

- DSL for Log Service

  The domain-specific language (DSL) for Log Service is a Python-compatible scripting language, and is used for data transformation in Log Service. The DSL for Log Service is built on top of Python. The DSL provides more than 200 built-in functions to simplify common data transformation tasks. The DSL also allows you to use custom Python extensions. For more information, see Language introduction.

- transformation rule

  A transformation rule is a data transformation script that is orchestrated by using the DSL for Log Service.

- data transformation task

  A data transformation task is the minimum scheduling unit of data transformation. You must configure a source Logstore, one or more destination Logstores, a transformation rule, a transformation time range, and other parameters for a data transformation task.

Rule-related terms

- resource

  Resources refer to third-party data sources that are referenced during data transformation. The data sources include but are not limited to on-premises resources, Object Storage Service (OSS), ApsaraDB RDS, and Logstores other than the source and destination Logstores. The resources may be referenced to enrich data. For more information, see Resource functions.

- dimension table

  A dimension table contains dimension information that can be used to enrich data. A dimension table is an external table. For example, a dimension table can contain the information of users, products, and geographical locations of a company. In most scenarios, dimension tables are included in resources and may be dynamically updated.

- enrichment or mapping

  If the information contained in a log cannot meet your requirements, you can map one or more fields in the log by using a dimension table to obtain more information. This process is called enrichment or mapping.

  For example, a request log contains the status field that specifies the HTTP status code. You can map the field to the status_desc field to obtain the HTTP status description by using the following table.

Before enrichment	After enrichment
status	status_desc
200	Success

Before enrichment	After enrichment
300	Redirect
400	Permission error
500	Server error

If a source log contains the user_id field, you can map the field by using a dimension table that contains account details to obtain more information. For example, you can obtain the user name, gender, registration time, and email address for each user ID. Then, you can add the information to the source log and write the log to the destination Logstores. For more information, see Mapping and enrichment functions.

- event splitting

  If a log contains multiple pieces of information, the log can be split into multiple logs. This process is called event splitting.

  For example, a log contains the following information:

```
__time__: 1231245
__topic: "win_logon_log"
content:
[ {
  "source": "192.0.2.1",
  "dest": "192.0.2.1"
  "action": "login",
  "result": "pass"
},{
  "source": "192.0.2.2",
  "dest": "192.0.2.1"
  "action": "logout",
  "result": "pass"
}
]
```

  The log can be split into two logs.

```
__time__: 1231245
__topic: "win_logon_log"
content:
{
  "source": "192.0.2.1",
  "dest": "192.0.2.1"
  "action": "login",
  "result": "pass"
}
```

```
__time__: 1231245
__topic: "win_logon_log"
content:
{
  "source": "192.0.2.2",
  "dest": "192.0.2.1"
  "action": "logout",
  "result": "pass"
}
```

- grok

  Grok uses patterns to replace complex regular expressions.

  For example, the `grok("%{IPV4}")` pattern indicates a regular expression that is used to match IPv4 addresses and is equivalent to the expression `"(?<![0-9])(?:(?:[0-1]?[0-9]{1,2}|2[0-4][0-9]|25[0-5])[.](?:[0-1]?[0-9]{1,2}|2[0-4][0-9]|25[0-5])[.](?:[0-1]?[0-9]{1,2}|2[0-4][0-9]|25[0-5])[.](?:[0-1]?[0-9]{1,2}|2[0-4][0-9]|25[0-5]))(?![0-9])"`. For more information, see Grok function.

- content capturing by using a regular expression

  You can use a regular expression to capture specified content in a field and include the content in a new field.

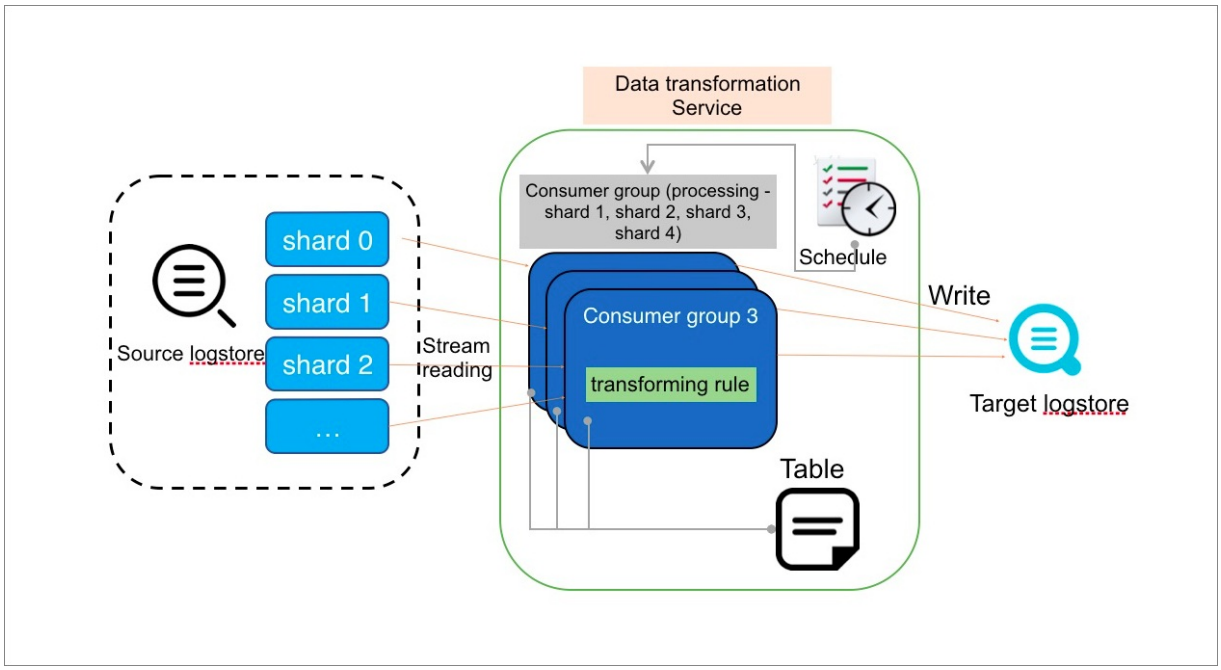
  For example, the function `e_regex("content", "(?P<email>[a-zA-Z][a-zA-Z0-9_.+-=:]+@\w+\.com)")` extracts the email address from the `content` field and includes the extracted email address in the `email` field. The email address information is extracted by using a common regular expression. We recommend that you use the following grok pattern to simplify the regular expression: `e_regex("content", grok("%{EMAILADDRESS:email}")`. For more information, see Regular expressions.

3.Data transformation basics

The data transformation feature uses consumer groups to consume log data and uses transformation rules to transform log data. More than 200 built-in functions are available for you to orchestrate transformation rules. This topic describes how log data consumption is scheduled during data transformation and how the rules engine for data transformation works.

Scheduling basics

The data transformation feature of Log Service uses a consumer group to consume log data from the source Logstore in the streaming mode, transforms each log entry based on the specified transformation rule, and then writes the transformed log data to one or more destination Logstores.



- **Scheduling mechanism**

  For each transformation rule, the data transformation scheduler starts one or more running instances. Each running instance behaves as a consumer to consume data from one or more shards of the source Logstore. The scheduler determines the number of concurrent running instances based on the memory and CPU resources that are used by running instances. The maximum number of running instances that the scheduler can start is the same as the number of shards in the source Logstore.

- **Running instance**

  Running instances read source log data from the shards that are allocated to them based on your configurations. The data is transformed based on the transformation rule and is then written to one or more destination Logstores. You can configure transformation rules to enrich log data by using external resources. Based on the consumer group mechanism, running instances record data consumption checkpoints in shards. The checkpoints are useful if consumption is unexpectedly interrupted. After an interruption ends, running instances can continue to consume data from the last checkpoint.

- **Task termination**
  - By default, if you do not set the end time of a transformation task, running instances do not exit and the task does not stop.

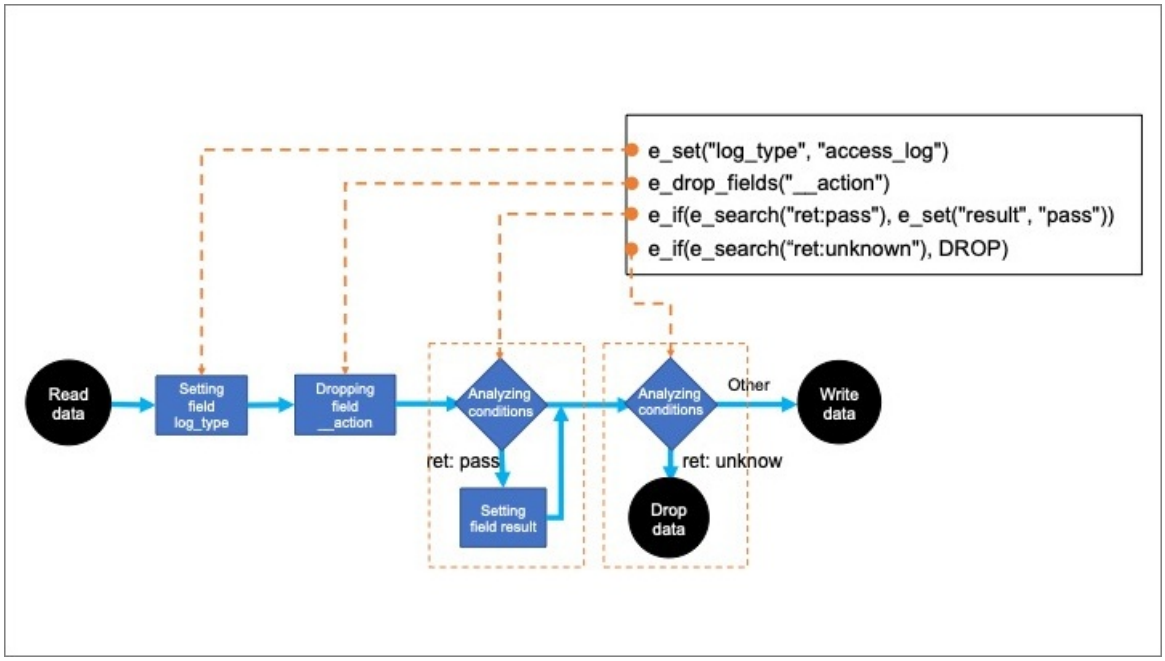
- If you set the end time of a transformation task, running instances consume log data until the end time. When the task reaches the end time, the instances exit and the task stops.
- By default, if a task is stopped and then restarted, running instances continue to consume data from the last recorded checkpoint.

Rules engine basics: basic operations

You can use the built-in functions that are written in the domain specific language (DSL) for Log Service to orchestrate transformation rules. Each function is a transformation step. The rules engine calls the functions of a transformation rule in sequence. For example, the following transformation rule is orchestrated by using four functions. The functions are four steps that are used to transform data:

```
e_set("log_type", "access_log")
e_drop_fields("__action")
e_if(e_search("ret: pass"), e_set("result", "pass"))
e_if(e_search("ret: unknown"), DROP)
```

The following figure shows the transformation logic.



- **Basic logic**

  The rules engine calls each function that is defined in the rule in sequence. Each function processes and modifies each log entry, and returns a transformed log entry.

  For example, the `e_set("log_type", "access_log")` function adds the *log_type* field to each log entry. The value of the field is `access_log`. Then, the next function receives each transformed log entry that contains the `log_type` field.

- **Conditional expressions**

  You can set conditions in steps. If a log entry does not meet a condition in a step, this step is skipped for the log entry.

For example, the `e_if(e_search("ret: pass"), e_set("result", "pass"))` function first checks whether the value of the `ret` field in a log entry contains *pass*. If no, this step is skipped for the log entry. If yes, the function sets the value of the `result` field in the log entry to *pass*.

- **Transformation termination**

  If a function does not return a transformed log entry, the log entry is discarded.

  For example, the `e_if(e_search("ret: unknown"), DROP)` function discards a log entry in which the value of the `ret` field is *unknown*. After the log entry is discarded, the rules engine no longer calls subsequent functions to transform this log entry, and starts to transform the next log entry.

Rules engine basics: data output, duplication, and splitting

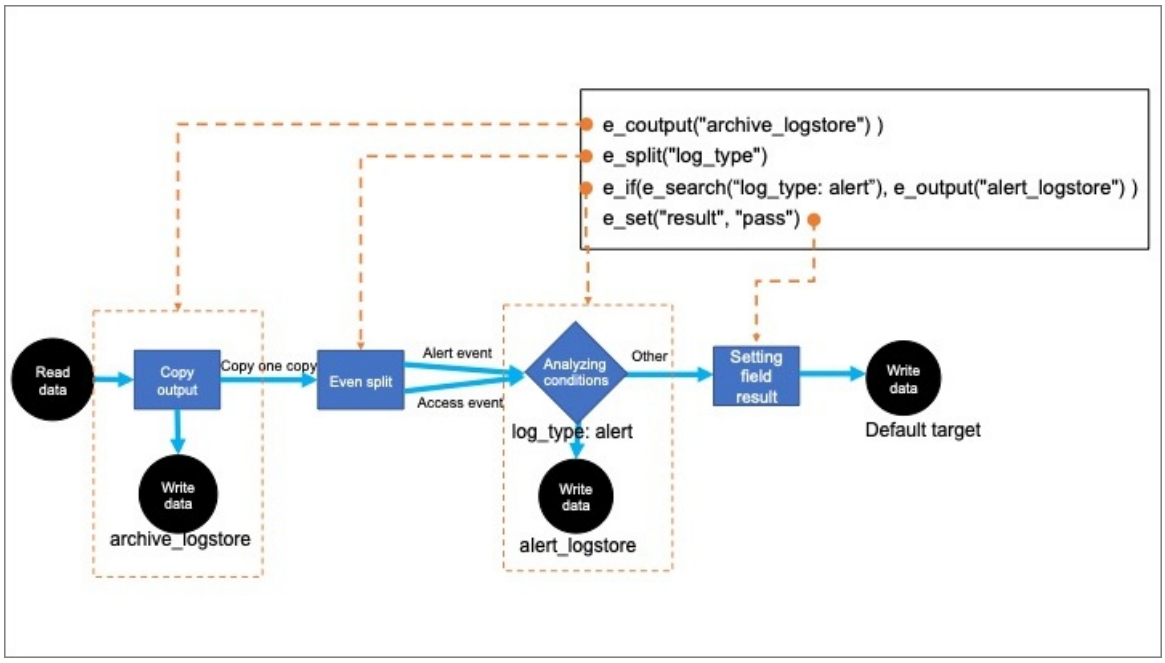
The rules engine also supports log output, duplication, and splitting. For example, the following transformation rule is orchestrated by using four functions. The functions are four steps that are used to transform data:

```
e_coutput("archive_Logstore") )
e_split("log_type")
e_if(e_search("log_type: alert"), e_output("alert_Logstore") )
e_set("result", "pass")
```

The following example is a sample log entry to be transformed:

```
log_type: access,alert
content: admin login to database.
```

The following figure shows the transformation logic.



- **Log output**

Log output can be considered as a special way to stop transforming a log entry. As shown in the preceding figure, if the value of the `log_type` field in a log entry is *alert*, the `e_output("alert_Logstore")` function in step 3 is called to write the log entry to the specified destination Logstore. Then the log entry is discarded and the subsequent function is not called.

- **Log duplication and output**

  The `e_coutput` function duplicates a log entry and writes the duplicated log entry to one or more destination Logstores. Then, the rules engine continues to call subsequent functions to transform the log entry. As shown in the preceding figure, log entries that are duplicated in step 1 are written to the destination Logstore named `archive_Logstore`.

- **Log splitting**

  If the values of the `log_type` field in a log entry are *access* and *alert*, the `e_split("log_type")` function in step 2 is called to split the log entry into two log entries. The two log entries are the same except the value of the `log_type` field. The value of the field is *access* in a log entry and is *alert* in the other.

  The log entries that are generated after the splitting are transformed by the subsequent functions.

4.Supported regions

This topic describes the regions that are supported by the data transformation feature.

You can use the data transformation feature in the following regions.

Region	
China (Hangzhou)	China (Hong Kong)
China East 1 Finance	Japan (Tokyo)
China (Shanghai)	South Korea (Seoul)
China East 2 Finance	Singapore (Singapore)
China (Qingdao)	Australia (Sydney)
China (Beijing)	Malaysia (Kuala Lumpur)
China North 2 Finance	Indonesia (Jakarta)
China North 2 Ali Gov 1	Philippines (Manila)
China (Zhangjiakou)	Thailand (Bangkok)
China (Hohhot)	UAE (Dubai)
China (Ulanqab)	US (Silicon Valley)
China (Shenzhen)	Germany (Frankfurt)
China South 1 Finance	US (Virginia)
China (Heyuan)	India (Mumbai)
China (Guangzhou)	UK (London)
China (Chengdu)	N/A

You can go to the Project Overview page of a project in the Log Service console to view the endpoints for the region where the project resides. For more information about the Log Service endpoint for each region, see Endpoints.

5.Grant permissions

5.1. Authorization overview

Before you can use the data transformation feature, you must have the permissions to perform related operations for data transformation and to access the required data.

- If you use an Alibaba Cloud account, you need only to authorize a data transformation task to access the required data before you can run the task.

  > **Notice** To ensure the security of your cloud resources, we recommend that you use a Resource Access Management (RAM) user.

- If you use a RAM user, you must authorize the user to perform the related operations and authorize a data transformation task to access the required data before you can run the task.

Permissions to perform related operations

The operations include creating, deleting, modifying, and viewing a data transformation task. The operations also include previewing transformation results.

- You can use an Alibaba Cloud account to perform the operations. The account has the management permissions on Log Service that are specified by the AliyunLogFullAccess policy. If you use an Alibaba Cloud account to run a data transformation task, you do not need to grant the permissions to perform related operations.
- You can also use a RAM user to perform the operations. If you use a RAM user to run a data transformation task, you must grant the user the permissions to perform related operations by using an Alibaba Cloud account. We recommend that you use a RAM user. For more information, see Authorize a RAM user to manage a data transformation task.

Permissions to access the required data

The following procedure describes how a data transformation task works. Steps 1 and 3 involve data access. The system needs to read data from a source Logstore and write transformed data to one or more destination Logstores.

1. Read data from the source Logstore.
2. Transform the data.
3. Write the transformed data to the destination Logstores.

You can grant permissions to access the required data by using a default role, custom role, or AccessKey pair.

- Default role: You can authorize a data transformation task to assume the system role AliyunLogETLRole to read data from a source Logstore and write transformed data to one or more destination Logstores. For more information, see Access data by using a default role.

  The AliyunLogETLRole role has access permissions on Logstores.

- Custom role: You can authorize a data transformation task to assume a custom role to read data from a source Logstore and write transformed data to one or more destination Logstores. For more information, see Access data by using a custom role.

You must use an Alibaba Cloud account to grant the access permissions on Logstores to the custom role.

- AccessKey pair: You can authorize a data transformation task to use the AccessKey pair of an Alibaba Cloud account or a RAM user to read data from a source Logstore and write transformed data to one or more destination Logstores. For more information, see Access data by using an AccessKey pair.
  - By default, an Alibaba Cloud account has access permissions on Logstores.
  - By default, a RAM user does not have access permissions on Logstores. You must use an Alibaba Cloud account to grant the access permissions to the RAM user.

5.2. Authorize a RAM user to manage a data transformation task

This topic describes how to authorize a RAM user to manage a data transformation task.

Prerequisites

A RAM user is created. For more information, see Step 1: Create a RAM user.

Context

You can use your Alibaba Cloud account to authorize a RAM user to manage a data transformation task.

- Create, delete, or modify a data transformation task.
- Read the data in a source Logstore to preview the results of a data transformation task.

> **Note** An authorized RAM user can manage a data transformation task in the Log Service console. The permissions that are granted to a RAM user to manage a transformation task are different from the permissions that are granted to a data transformation task to access the data in a Logstore. The AccessKey pair of the current RAM user may be used to manage a data transformation task and access the data in a Logstore during a data transformation task. In this case, you must combine the content of the permission policy in this topic with the content of the permission policy in Access data by using an AccessKey pair.

You can authorize a RAM user to transform data in Log Service by using one of the following modes:

- Simple mode: allows you to grant full access permissions on Log Service to the RAM user. You do not need to modify the related parameters.
- Custom mode: allows you to create custom permission policies and grant the required permissions to the RAM user. This mode requires complex configurations and provides fine-grained access control.

Simple mode

Use your Alibaba Cloud account to log on to the RAM console. Attach the AliyunRAMReadOnlyAccess policy and AliyunLogFullAccess policy to the RAM user. For more information, see Create a RAM user and authorize the RAM user to access Log Service.

Custom mode

1. Use your Alibaba Cloud account to log on to the RAM console.

2. Create a policy.
   i. In the left-side navigation pane, choose **Permissions > Policies**.
   ii. On the Policies page, click **Create Policy**.
   iii. On the **Create Custom Policy** page, set the parameters and click **OK**. The following table describes the parameters.

Parameter	Description
Policy Name	The name of the policy.
Configuration Mode	Select **Script**.
	The content of the policy. Replace the content in the editor with the following script. Replace *Project name* with the name of the project in which a data transformation task is created. Replace *Logstore name* with the name of the related Logstore. **Note** If you want to use the AccessKey pair of the current RAM user to read and write Logstore data, you must add the related policy to the following sample script. For more information, see Access data by using an AccessKey pair. `{ "Version":"1", "Statement":[{ "Effect":"Allow", "Action":["log:CreateLogStore", "log:CreateIndex", "log:UpdateIndex", "log:Get*"], "Resource":"acs:log:*:*:project/Project name/logstore/internal-etl-log" }, { "Action":["log:List*"], "Resource":"acs:log:*:*:project/Project name/logstore/*", "Effect":"Allow" }, { "Action":["log:Get*", "log:List*"], "Resource":[`

Policy Document Parameter	Description
	"acs:log:*:*:project/Project name/logstore/Logstore name"

```
            ],
            "Effect":"Allow"
        },
        {
            "Effect":"Allow",
            "Action":[
                "log:GetDashboard",
                "log:CreateDashboard",
                "log:UpdateDashboard"
            ],
            "Resource":"acs:log:*:*:project/Project name/dashboard/internal-etl-insight*"
        },
        {
            "Effect":"Allow",
            "Action":"log:CreateDashboard",
            "Resource":"acs:log:*:*:project/Project name/dashboard/*"
        },
        {
            "Effect":"Allow",
            "Action":[
                "log:*"
            ],
            "Resource":"acs:log:*:*:project/Project name/job/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ram:PassRole",
                "ram:GetRole",
                "ram:ListRoles"
            ],
            "Resource": "*"
         }
    ]
}
```

3. Grant the required permissions to the RAM user.
   i. In the left-side navigation pane, choose **Identities > Users**.
   ii. On the Users page, find the RAM user and click **Add Permissions** in the Actions column.
   iii. In the Add Permissions panel, click **System Policy** in the Select Policy section, and then select the **AliyunRAMReadOnlyAccess** policy.
   iv. Click **Custom Policy** in the Select Policy section, and then select the policy that you created in Step .
   v. Click **OK**.

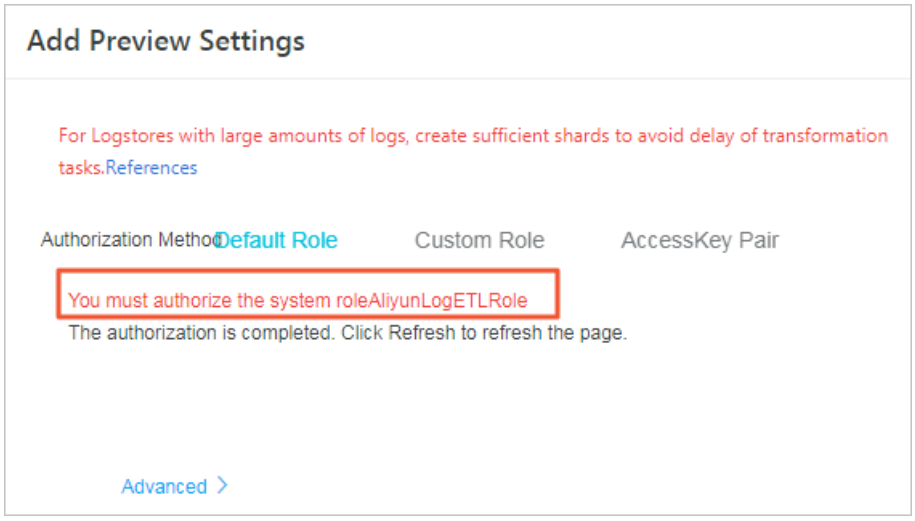
5.3. Authorize Log Service

5.3.1. Access data by using a default role

You can authorize a data transformation task to assume the system role AliyunLogETLRole to read data from a source Logstore and write transformed data to one or more destination Logstores. The AliyunLogETLRole role has access permissions on Logstores.

Procedure

In the **Create Data Transformation Rule** panel, click **You must authorize the system role AliyunLogETLRole** below **Default Role**. Then, configure other parameters as prompted to complete the authorization. For more information, see Create a data transformation job.



> **Note**
> - If you use a Resource Access Management (RAM) user, you must use an Alibaba Cloud account to assign the AliyunLogETLRole role to the RAM user.
> - If you use an Alibaba Cloud account that has assumed the role, you can skip this operation.

5.3.2. Access data by using a custom role

You can assign a custom role to a data transformation task to read data from a source Logstore and write transformed data to one or more destination Logstores. This topic describes how to grant access permissions on Logstores to a custom role.

Prerequisites

A Resource Access Management (RAM) role is created. For more information, see Create a RAM role for a trusted Alibaba Cloud service.

Grant the RAM role the permissions to read data from a source Logstore

After you use an Alibaba Cloud account to authorize the RAM role, the RAM role has permissions to read data from the source Logstore. When you create a data transformation task, you can use the RAM role. For more information, see Create a data transformation job.

1. Log on to the RAM console by using an Alibaba Cloud account.
2. Create a policy.

   The policy is used to allow the RAM role to read data from a source Logstore.

   i. In the left-side navigation pane, choose **Permissions > Policies**.

   ii. On the **Policies** page, click **Create Policy**.

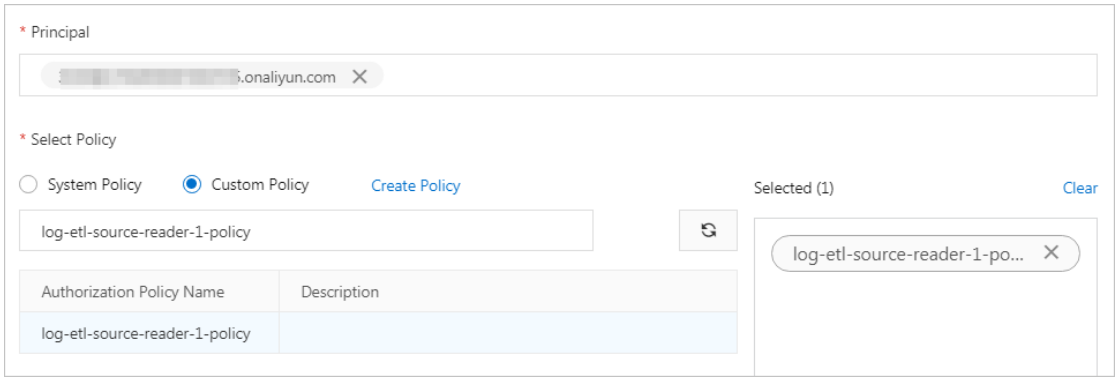
   iii. On the **Create Custom Policy** page, configure the following parameters and click **OK**.

Parameter	Description
Policy Name	The name of the policy. In this example, enter log-etl-source-reader-1-policy.
Configuration Mode	Select **Script**.
	The content of the policy. Replace the content in the editor with one of the following scripts based on your business requirements. ■ Policy that uses exact match The source project name is log-project-prod. The source Logstore name is access_log. Replace the project and Logstore names based on your business requirements. `{ "Version": "1", "Statement": [{ "Action": ["log:ListShards", "log:GetCursorOrData", "log:GetConsumerGroupCheckPoint", "log:UpdateConsumerGroup", "log:ConsumerGroupHeartBeat", "log:ConsumerGroupUpdateCheckPoint", "log:ListConsumerGroup", "log:CreateConsumerGroup"], "Resource": ["acs:log:*:*:project/log-project-prod/logstore/access_log", "acs:log:*:*:project/log-project-prod/logstore/access_log/*"], "Effect": "Allow" }] }`

Parameter	Description
Policy Document	■ Policy that uses fuzzy match The source project names can be log-project-dev-a, log-project-dev-b, or log-project-dev-c. The source Logstore names can be app_a_log, app_b_log, or app_c_log. Replace the project and Logstore names based on your business requirements. `{ "Version": "1", "Statement": [{ "Action": ["log:ListShards", "log:GetCursorOrData", "log:GetConsumerGroupCheckPoint", "log:UpdateConsumerGroup", "log:ConsumerGroupHeartBeat", "log:ConsumerGroupUpdateCheckPoint", "log:ListConsumerGroup", "log:CreateConsumerGroup"], "Resource": ["acs:log:*:*:project/log-project-dev-*/logstore/app_*_log", "acs:log:*:*:project/log-project-dev-*/logstore/app_*_log/*"], "Effect": "Allow" }] }` For more information about authorization scenarios, see Use custom policies to grant permissions to a RAM user.

Parameter	Description

3. Attach the policy to the RAM role.
   i. In the left-side navigation pane, choose **Identities > Roles**.
   ii. On the **RAM Roles** page, find the RAM role and click **Add Permissions** in the Actions column.
   iii. In the Select Policy section, click **Custom Policy**, select the policy that you created in Step , and then click **OK**. In this example, the policy is log-etl-source-reader-1-policy.



   iv. Confirm the authorization result and click **Complete**.
4. Obtain the Alibaba Cloud Resource Name (ARN) of the RAM role.

   In the **Basic Information** section of the RAM role, obtain the ARN. Example: acs:ram::13234:role/logsource.

Grant the RAM role the permissions to write data to destination Logstores within the same Alibaba Cloud account

If the source and destination Logstores belong to the same Alibaba Cloud account, you can use an Alibaba Cloud account to authorize the RAM role. Then, the RAM role has the permissions to write transformed data to the destination Logstores. When you create a data transformation task, you can use the RAM role. For more information, see Create a data transformation job.

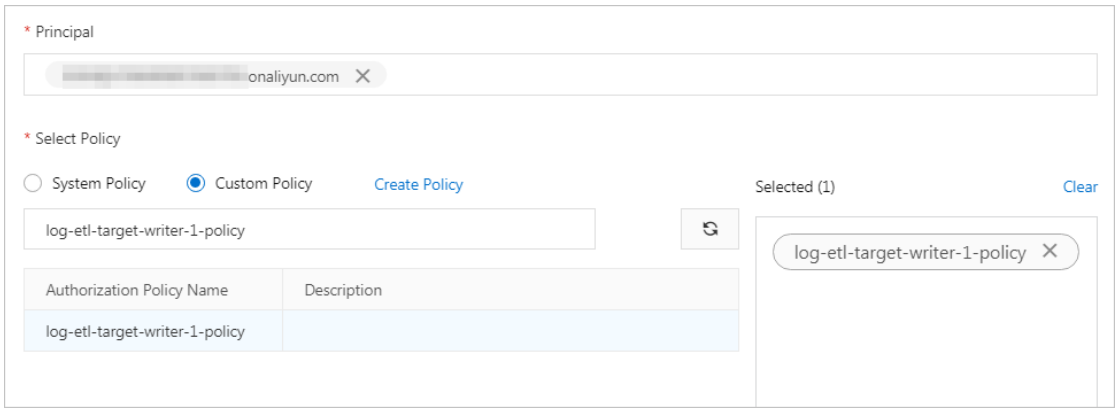
1. Log on to the RAM console by using an Alibaba Cloud account.
2. Create a policy.
   i. In the left-side navigation pane, choose **Permissions > Policies**.
   ii. On the **Policies** page, click **Create Policy**.
   iii. On the **Create Custom Policy** page, configure the following parameters and click **OK**.

Parameter	Description
Policy Name	The name of the policy. In this example, enter log-etl-target-writer-1-policy.
Configuration Mode	Select **Script**.

Parameter	Description

Policy Document	The content of the policy. Replace the content in the editor with one of the following scripts based on your business requirements.
	■ Policy that uses exact match
	The destination project name is log-project-prod. The destination Logstore name is access_log_output. Replace the project and Logstore names based on your business requirements.
	`{ "Version": "1", "Statement": [ { "Action": [ "log:Post*", "log:BatchPost*" ], "Resource": "acs:log:*:*:project/log-project-prod/logstore/access_log_output", "Effect": "Allow" } ] }`
	■ Policy that uses fuzzy match
	The destination project names can be log-project-dev-a, log-project-dev-b, or log-project-dev-c. The destination Logstore names can be app_a_log_output, app_b_log_output, or app_c_log_output. Replace the project and Logstore names based on your business requirements.
	`{ "Version": "1", "Statement": [ { "Action": [ "log:Post*", "log:BatchPost*" ], "Resource": "acs:log:*:*:project/log-project-dev-*/logstore/app_*_log_output", "Effect": "Allow" } ] }`
	For more information about authorization scenarios, see Use custom policies to grant permissions to a RAM user.

3. Attach the policy to the RAM role.
   i. In the left-side navigation pane, choose **Identities > Roles**.
   ii. On the **RAM Roles** page, find the RAM role and click **Add Permissions** in the Actions column.
   iii. In the Select Policy section, click **Custom Policy**, select the policy that you created in Step , and then click **OK**. In this example, the policy is log-etl-target-writer-1-policy.



   iv. Confirm the authorization result and click **Complete**.
4. Obtain the ARN of the RAM role.

   In the **Basic Information** section of the RAM role, obtain the ARN. Example: acs:ram::13234:role/logtarget.

Grant the RAM role the permissions to write data to destination Logstores across Alibaba Cloud accounts

If the source and destination Logstores belong to different Alibaba Cloud accounts, perform the following steps to grant permissions to the RAM role. For example, a data transformation task is created to read data from a source Logstore that belongs to Alibaba Cloud Account A and write transformed data to a destination Logstore that belongs to Alibaba Cloud Account B.

> **Notice** Before you perform the following steps, you must use Alibaba Cloud Account B to authorize the RAM role to access destination Logstores within the same Alibaba Cloud account. For more information, see Grant the RAM role the permissions to write data to destination Logstores within the same Alibaba Cloud account.

1. Use Alibaba Cloud Account B to log on to the RAM console.
2. In the left-side navigation pane, choose **Identities > Roles**.
3. On the Roles page, click the name of the RAM role.
4. On the page that appears, click the **Trust Policy Management** tab. Then, click **Edit Trust Policy**.
5. Modify the policy.

   Add *ID of Alibaba Cloud Account A to which the source Logstore belongs* to the Service element. Replace *ID of Alibaba Cloud Account A to which the source Logstore belongs* with the ID of your Alibaba Cloud account. You can view the ID of your Alibaba Cloud account in the Account Management console. The following policy allows Alibaba Cloud Account A to obtain a temporary token to manage the cloud resources of Alibaba Cloud Account B.

```
{
    "Statement": [
        {
            "Action": "sts:AssumeRole",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ID of Alibaba Cloud Account A to which the source Logstore belongs
@log.aliyuncs.com"
                ]
            }
        }
    ],
    "Version": "1"
}
```

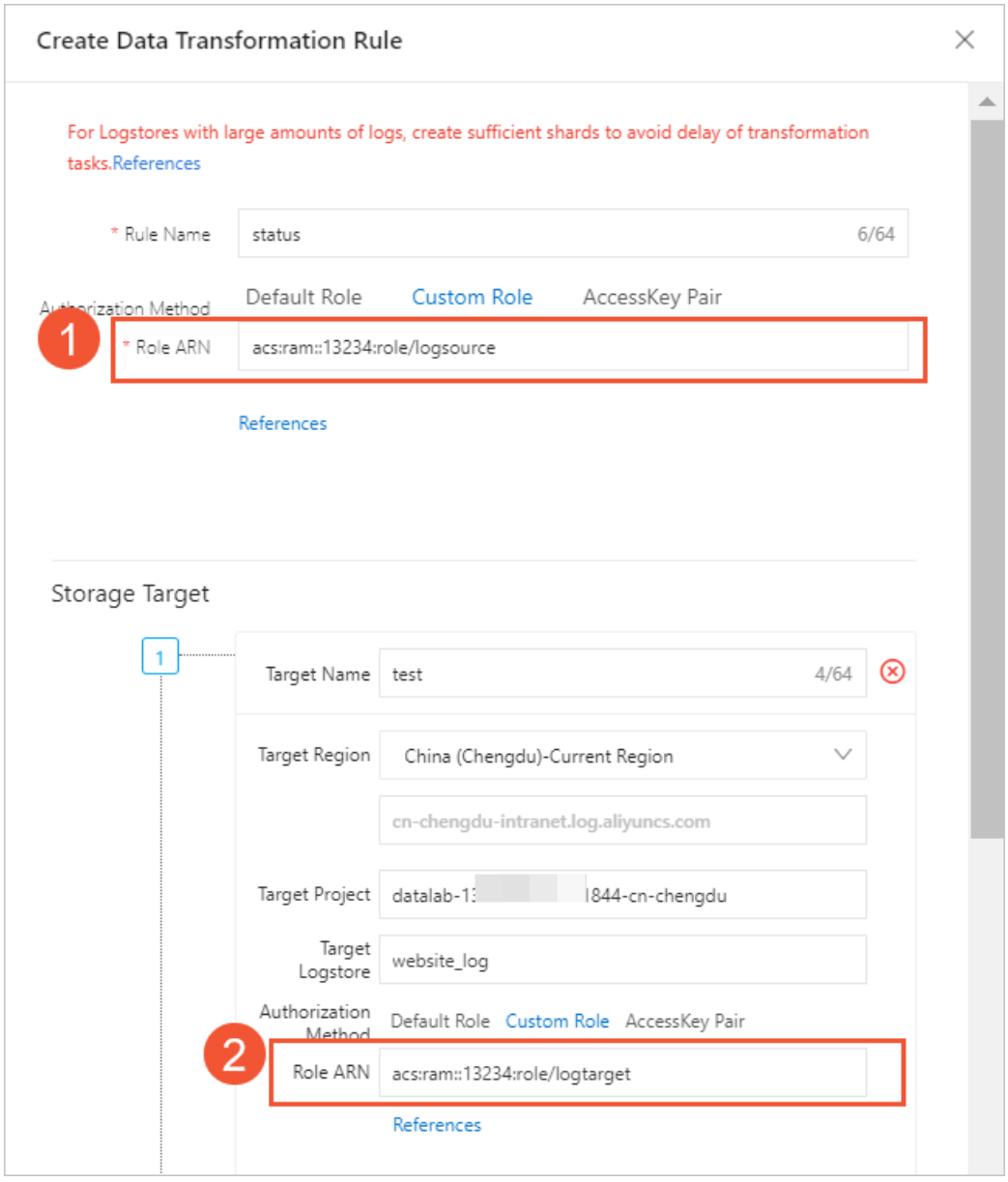
6. Obtain the ARN of the RAM role.

   In the **Basic Information** section of the RAM role, obtain the ARN. Example: acs:ram::13234:role/logtarget.

What's next

When you create a data transformation task, specify the ARN of the RAM role. For more information, see Create a data transformation job.

- In Section 1, enter the ARN of the RAM role that has permissions to read data from a source Logstore. For more information, see Grant the RAM role the permissions to read data from a source Logstore.
- In Section 2, enter the ARN of the RAM role that has permissions to write data to destination Logstores. For more information, see Grant the RAM role the permissions to write data to destination Logstores within the same Alibaba Cloud account or Grant the RAM role the permissions to write data to destination Logstores across Alibaba Cloud accounts.



5.3.3. Access data by using an AccessKey pair

You can authorize a data transformation task to use the AccessKey pair of an Alibaba Cloud account or a Resource Access Management (RAM) user to read data from a source Logstore and write transformed data to one or more destination Logstores. The AccessKey pair of an Alibaba Cloud account has access permissions on Logstores and can be directly used. If you use a RAM user, you must grant the RAM user the access permissions on Logstores. For more information, see the following procedures.

Prerequisites

A RAM user is created. For more information, see Create a RAM user.

> **Notice**
>
> - When you create the RAM user, select **Programmatic Access** for **Access Mode**. Then, record the AccessKey pair of the RAM user.
> - The AccessKey secret is displayed only when you create the RAM user. The AccessKey secret cannot be queried. We recommend that you record the AccessKey secret for subsequent use and keep it confidential.

Grant the RAM user the permissions to read from a source Logstore

After the RAM user is authorized by an Alibaba Cloud account, the RAM user has permissions to read from the source Logstore. When you create a data transformation task, you can enter the AccessKey pair of the RAM user. For more information, see Create a data transformation job.

1. Log on to the RAM console by using an Alibaba Cloud account.
2. Create a policy.

   The policy is used to allow the RAM user to read data from the source Logstore.

   i. In the left-side navigation pane, choose **Permissions > Policies**.

   ii. On the **Policies** page, click **Create Policy**.

   iii. On the **Create Custom Policy** page, configure the following parameters and click **OK**.

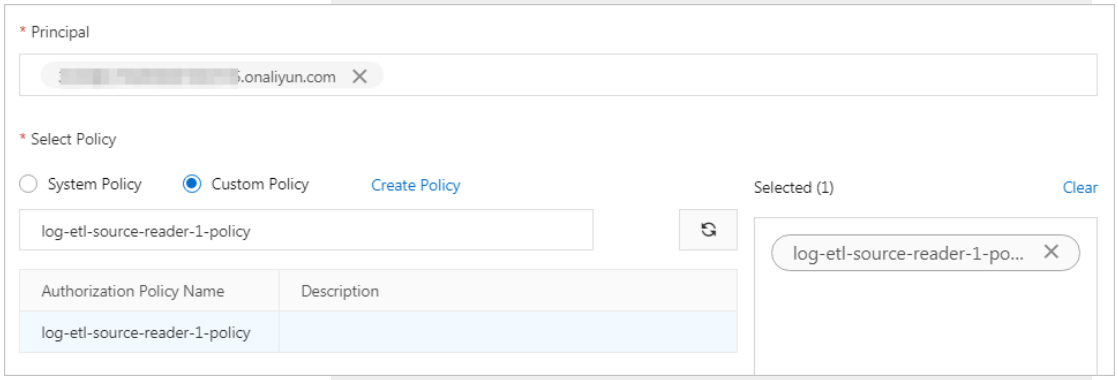
Parameter	Description
Policy Name	The name of the policy. In this example, enter log-etl-source-reader-1-policy.
Configuration Mode	Select **Script**.
	The content of the policy. Replace the content in the editor with one of the following scripts based on your business requirements.

Parameter	Description

Policy Document	■ Policy that uses exact match
	The source project name is log-project-prod. The source Logstore name is access_log. Replace the project and Logstore names based on your business requirements.
	{
	"Version": "1",
	"Statement": [
	{
	"Action": [
	"log:ListShards",
	"log:GetCursorOrData",
	"log:GetConsumerGroupCheckPoint",
	"log:UpdateConsumerGroup",
	"log:ConsumerGroupHeartBeat",
	"log:ConsumerGroupUpdateCheckPoint",
	"log:ListConsumerGroup",
	"log:CreateConsumerGroup"
	],
	"Resource": [
	"acs:log:*:*:project/log-project-prod/logstore/access_log",
	"acs:log:*:*:project/log-project-prod/logstore/access_log/*"
	],
	"Effect": "Allow"
	}
	]
	}

Parameter	Description
	■ Policy that uses fuzzy match The source project names can be log-project-dev-a, log-project-dev-b, or log-project-dev-c. The source Logstore names can be app_a_log, app_b_log, or app_c_log. Replace the project and Logstore names based on your business requirements.

3. Attach the policy to the RAM user.
   i. In the left-side navigation pane, choose **Identities > Users**.
   ii. On the Users page, find the RAM user and click **Add Permissions** in the Actions column.
   iii. In the Select Policy section, click the **Custom Policy** tab. From the list of custom policies, click the policy that you created in Step 2 and click OK. In this example, the policy is **log-etl-source-reader-1-policy**.



   iv. Confirm the authorization results. Then, click **Complete**.

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "log:Post*"
      ],
      "Resource": [
        "acs:log:*:*:project/log-project-dev-*/logstore/app_*_log",
        "acs:log:*:*:project/log-project-dev-*/logstore/app_*_log/*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

For more information about authorization scenarios, see Use custom policies to grant permissions to a RAM user.

Grant the RAM user the permissions to write to destination Logstores

After the RAM user is authorized by an Alibaba Cloud account, the RAM user has permissions to write to the destination Logstores. When you create a data transformation task, you can enter the AccessKey pair of the RAM user. For more information, see Create a data transformation job.

1. Log on to the RAM console by using an Alibaba Cloud account.
2. Create a policy.

   The policy is used to allow the RAM user to write data to the destination Logstores.
   i. In the left-side navigation pane, choose **Permissions > Policies**.
   ii. On the **Policies** page, click **Create Policy**.
   iii. On the **Create Custom Policy** page, configure the following parameters and click **OK**.

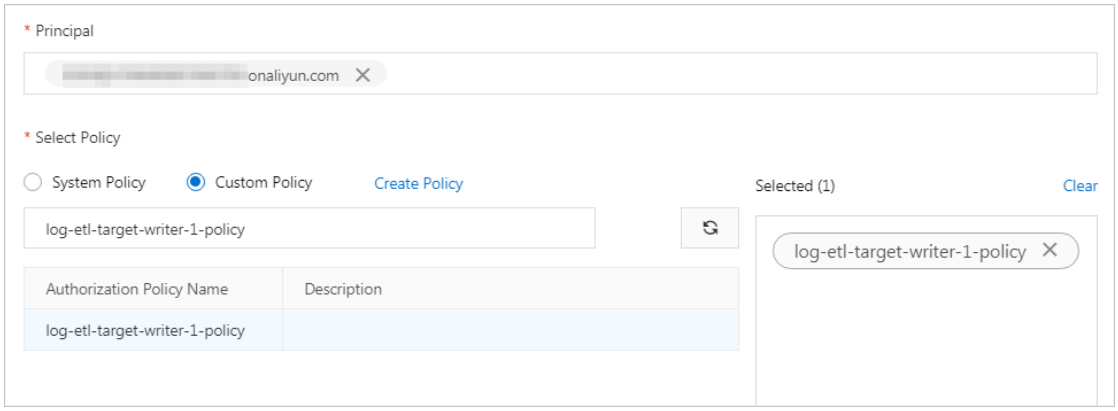
Parameter	Description
Policy Name	The name of the policy. In this example, enter log-etl-target-writer-1-policy.
Configuration Mode	Select **Script**.
	The content of the policy. Replace the content in the editor with one of the following scripts based on your business requirements.

Parameter	Description

Policy Document	■ Policy that uses exact match
	The destination project name is log-project-prod. The destination Logstore name is access_log_output. Replace the project and Logstore names based on your business requirements.
	`{ "Version": "1", "Statement": [ { "Action": [ "log:Post*", "log:BatchPost*" ], "Resource": "acs:log:*:*:project/log-project-prod/logstore/access_log_output", "Effect": "Allow" } ] }`
	■ Policy that uses fuzzy match
	The destination project names can be log-project-dev-a, log-project-dev-b, or log-project-dev-c. The destination Logstore names can be app_a_log_output, app_b_log_output, or app_c_log_output. Replace the project and Logstore names based on your business requirements.
	`{ "Version": "1", "Statement": [ { "Action": [ "log:Post*", "log:BatchPost*" ], "Resource": "acs:log:*:*:project/log-project-dev-*/logstore/app_*_log_output", "Effect": "Allow" } ] }`
	For more information about authorization scenarios, see Use custom policies to grant permissions to a RAM user.

3. Attach the policy to the RAM user.
   i. In the left-side navigation pane, choose **Identities > Users**.
   ii. On the Users page, find the RAM user and click **Add Permissions** in the Actions column.

iii. In the Select Policy section, click the **Custom Policy** tab. From the list of custom policies, click the policy that you created in Step 2 and click OK. In this example, the policy is **log-etl-target-writer-1-policy**.

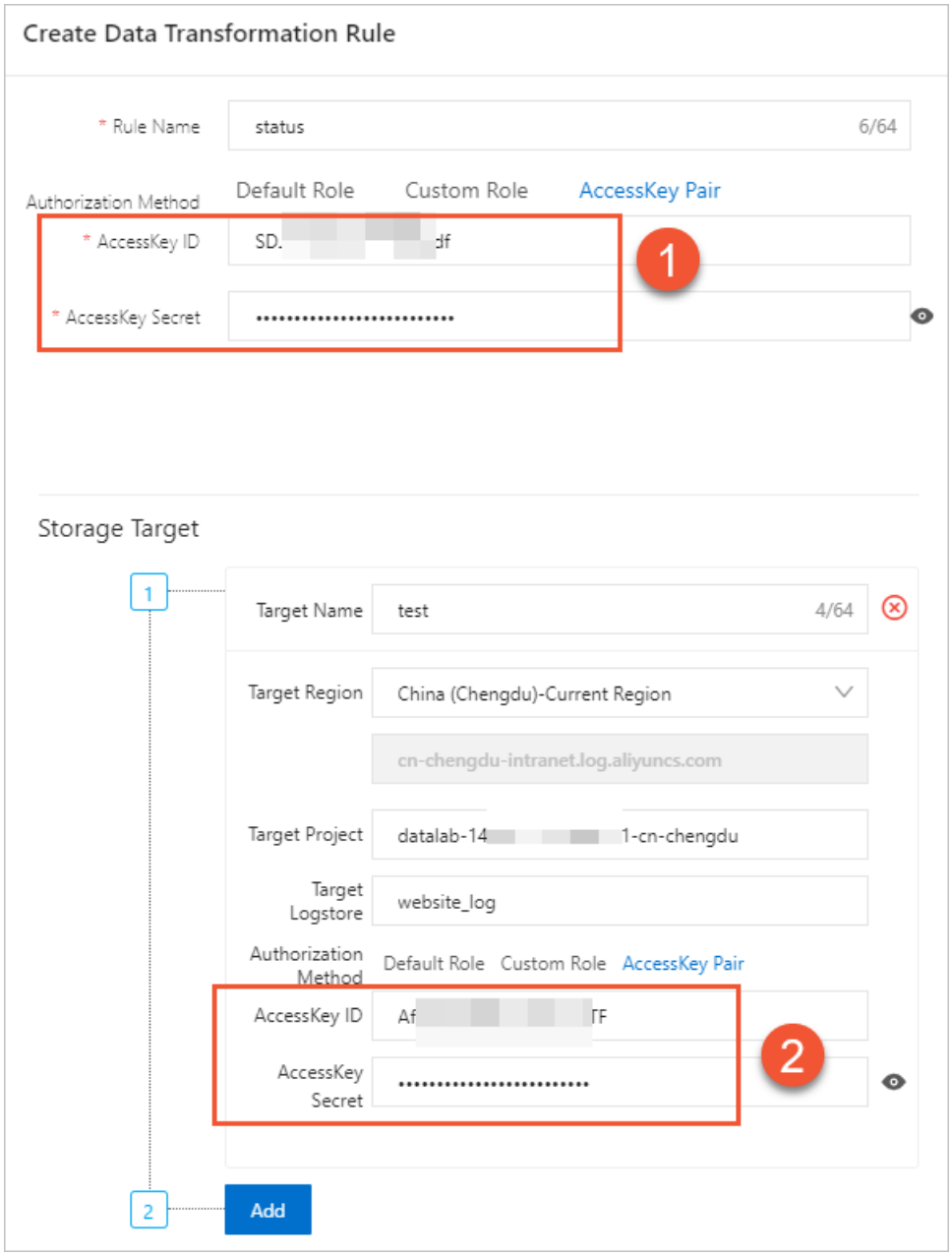


iv. Confirm the authorization results. Then, click **Complete**.

What's next

You can enter the AccessKey pair of the RAM user in a data transformation task. For more information, see Create a data transformation job.

- In Section 1, enter the AccessKey pair of the RAM user that has permissions to read from a source Logstore. For more information, see Grant the RAM user the permissions to read from a source Logstore.
- In Section 2, enter the AccessKey pair of the RAM user that has permissions to write to destination Logstores. For more information, see Grant the RAM user the permissions to write to destination Logstores.


Create Data Transformation Rule
* Rule Name
status
6/64
Authorization Method
Default Role
Custom Role
AccessKey Pair
* AccessKey ID
SD.
df
1
* AccessKey Secret
Storage Target
1
Target Name
test
4/64
Target Region
China (Chengdu)-Current Region
cn-chengdu-intranet.log.aliyuncs.com
Target Project
datalab-14
1-cn-chengdu
Target Logstore
website_log
Authorization Method
Default Role
Custom Role
AccessKey Pair
AccessKey ID
Af
TF
2
AccessKey Secret
2
Add

6.Quick start of data transformation

This topic describes a complete data transformation process to walk you through the feature and related operations. Website access logs are used as an example to describe the process.

Prerequisites

- A project named web-project is created. For more information, see Create a project.
- A Logstore named website_log is created in the web-project project, and the Logstore is used as the source Logstore. For more information, see Create a Logstore.
- Website access logs are collected and stored in the website_log Logstore. For more information, see Data collection methods.
- Destination Logstores are created in the web-project project. The following table lists the details about the destination Logstores.

Destination Logstore	Description
website-success	Logs for successful access are stored in the website-success Logstore, which is configured in the target-success storage destination.
website-fail	Logs for failed access are stored in the website-fail Logstore, which is configured in the target-fail storage destination.
website-etl	Other access logs are stored in the website-etl Logstore, which is configured in the target0 storage destination.

- If you use a Resource Access Management (RAM) user, the user is granted the permissions to perform related operations for data transformation. For more information, see Authorize a RAM user to manage a data transformation task.
- Indexes are configured for the source and destination Logstores. For more information, see Configure indexes.

  Data transformation does not require indexes. However, if you do not configure indexes, you cannot perform query or analysis operations.

Context

All access logs of a website are stored in a Logstore. You need to specify different topics for the logs to distinguish between logs for successful access and logs for failed access. In addition, you need to distribute the two types of logs to different Logstores for analysis. Log sample:

```
body_bytes_sent:1061
http_user_agent:Mozilla/5.0 (Windows; U; Windows NT 5.1; ru-RU) AppleWebKit/533.18.1 (KHTML
, like Gecko) Version/5.0.2 Safari/533.18.5
remote_addr:192.0.2.2
remote_user:vd_yw
request_method:DELETE
request_uri:/request/path-1/file-5
status:207
time_local:10/Jun/2021:19:10:59
```

Step 1: Create a data transformation job

1.
2. Go to the data transformation page.
   i.
   ii.
   iii. On the query and analysis page, click **Data Transformation**.
3. In the upper-right corner of the page, select a time range for the required log data.

   Make sure that the **Raw Logs** tab displays log data.
4. In the editor, enter transformation statements.

```
e_if(e_search("status:[200,299]"),e_compose(e_set("__topic__","access_success_log"),e_o
utput(name="target-success")))
e_if(e_search("status:[400,499]"),e_compose(e_set("__topic__","access_fail_log"),e_outp
ut(name="target-fail")))
```

   The e_if function indicates that the specified operations are performed if the condition is met. For more information, see e_if.

   - Condition: `e_search("status:[200,299]")`

     If the value of the status field meets the condition, the operations 1 and 2 are performed. For more information, see e_search.
   - Operation 1: `e_set("__topic__","access_success_log")`

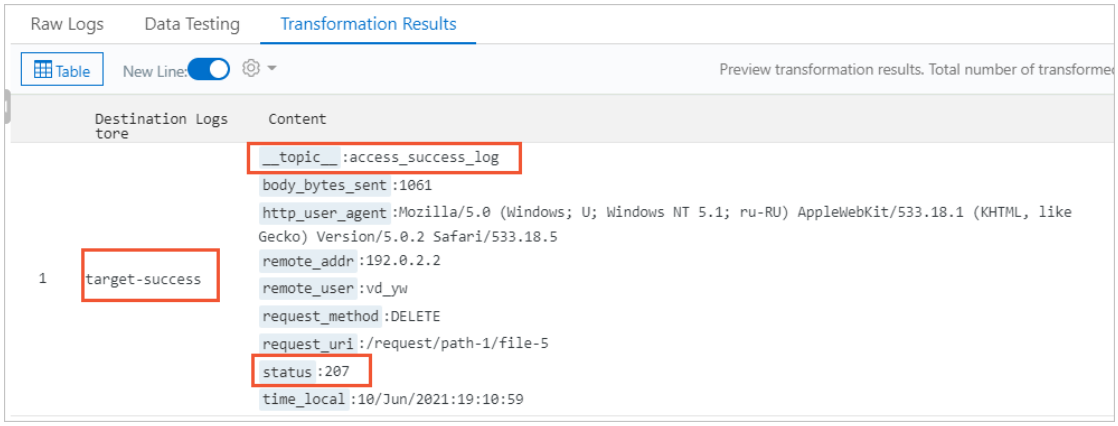
     The function adds the __topic__ field and assigns the value access_success_log to the field. For more information, see e_set.
   - Operation 2: `e_output(name="target-success", project="web-project", logstore="website-success")`

     The function stores the transformed data in the website-success Logstore. For more information, see Event processing functions.
5. Preview transformation results.
   i. Select **Quick**.

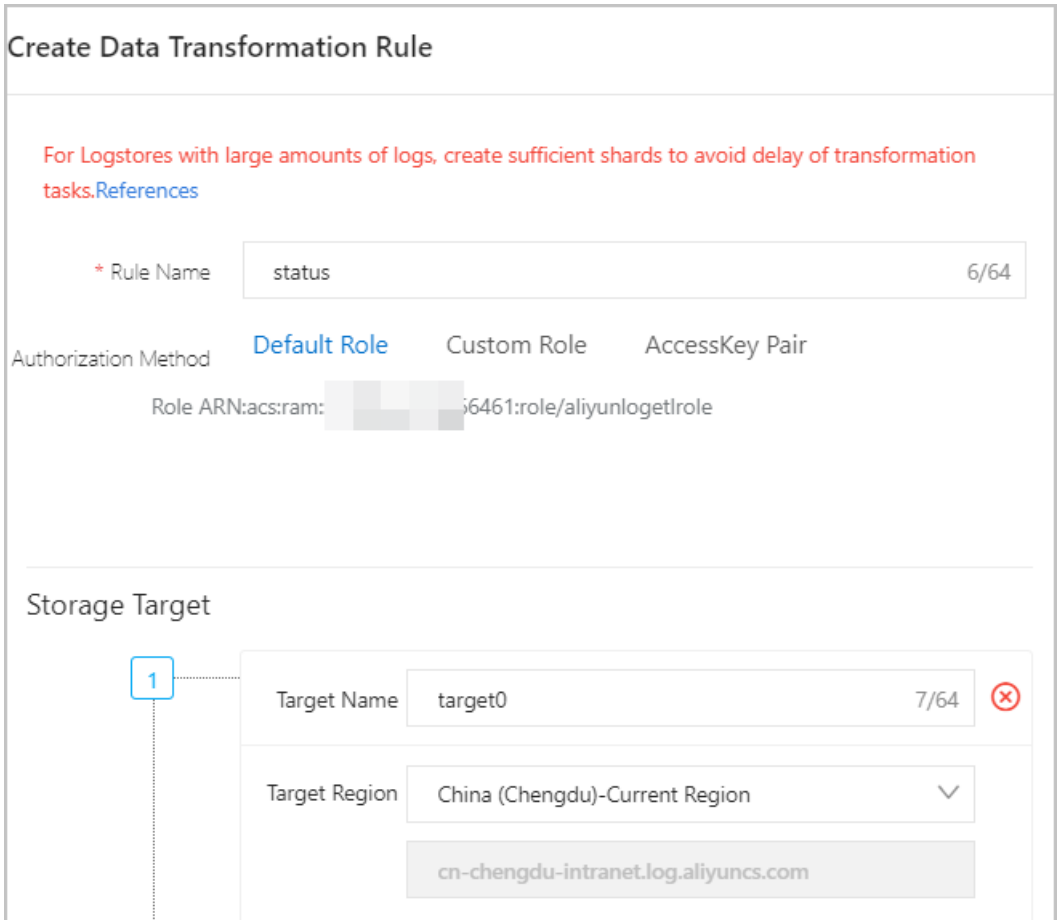
      You can select either Quick or Advanced. For more information, see Preview mode overview.

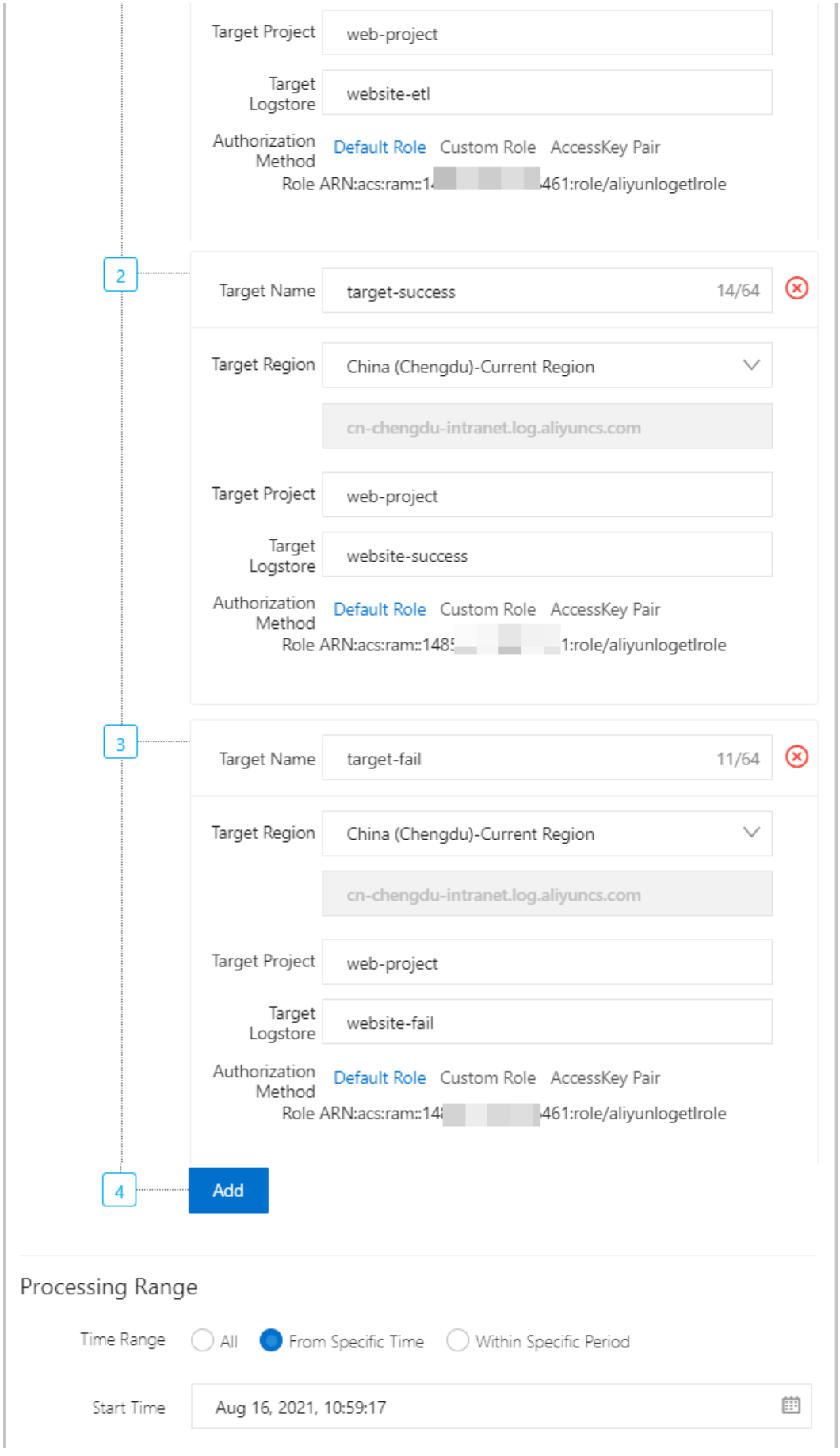
ii. Click **Preview Data**.
View the results.

> **Note** During the preview, logs are written to a Logstore named internal-etl-log instead of the destination Logstores. If this is your first time to preview transformation results, Log Service automatically creates the internal-etl-log Logstore in the current project. This Logstore is dedicated. You cannot modify the configurations of this Logstore or write other data to this Logstore. This Logstore is not charged.



6. Create a data transformation job.
   i. Click **Save as Transformation Rule**.
   ii. In the **Create Data Transformation Rule** panel, configure the following parameters.




Target Project
web-project
Target Logstore
website-etl
Authorization Method
Default Role
Custom Role
AccessKey Pair
Role ARN:acs:ram::1
461:role/aliyunlogetlrole
2
Target Name
target-success
14/64
Target Region
China (Chengdu)-Current Region
cn-chengdu-intranet.log.aliyuncs.com
Target Project
web-project
Target Logstore
website-success
Authorization Method
Default Role
Custom Role
AccessKey Pair
Role ARN:acs:ram::148
1:role/aliyunlogetlrole
3
Target Name
target-fail
11/64
Target Region
China (Chengdu)-Current Region
cn-chengdu-intranet.log.aliyuncs.com
Target Project
web-project
Target Logstore
website-fail
Authorization Method
Default Role
Custom Role
AccessKey Pair
Role ARN:acs:ram::14
461:role/aliyunlogetlrole
4
Add
Processing Range
Time Range
All
From Specific Time
Within Specific Period
Start Time
Aug 16, 2021, 10:59:17

Parameter	Description
Rule Name	The name of the transformation rule.

Parameter	Description
Authorization Method	The method used to authorize the data transformation job to read data from the source Logstore. Valid values: ■ Default Role: authorizes the data transformation job to assume the system role AliyunLogETLRole to read data from the source Logstore. You must click **You must authorize the system role AliyunLogETLRole**. Then, you must configure other parameters as prompted to complete the authorization. For more information, see Access data by using a default role. Note ■ If the authorization is complete within your Alibaba Cloud account, you can skip this operation. ■ If you use an Alibaba Cloud account that has assumed the role, you can skip this operation. ■ Custom Role: authorizes the data transformation job to assume a custom role to read data from the source Logstore. You must grant the custom role the permissions to read from the source Logstore. Then, you must enter the Alibaba Cloud Resource Name (ARN) of the custom role in the **Role ARN** field. For more information about authorization, see Access data by using a custom role. ■ AccessKey Pair: authorizes the data transformation job to use the AccessKey pair of an Alibaba Cloud account or a RAM user to read data from the source Logstore. ■ Alibaba Cloud account: The AccessKey pair of an Alibaba Cloud account has permissions to read from the source Logstore. You can directly enter the AccessKey ID and AccessKey secret of the Alibaba Cloud account in the **AccessKey ID** and **AccessKey Secret** fields. For more information about how to obtain an AccessKey pair, see AccessKey pair. ■ RAM user: You must grant the RAM user the permissions to read from the source Logstore. Then, you can enter the AccessKey ID and AccessKey secret of the RAM user in the **AccessKey ID** and **AccessKey Secret** fields. For more information about authorization, see Access data by using an AccessKey pair.
Storage Target	

Parameter	Description
Target Name	The name of the storage destination. Storage Target includes Target Project and Target Logstore. Make sure that the value of this parameter is the same as the value of name configured in . **Note** By default, Log Service uses the storage destination that is numbered 1 to store the logs that do not meet the specified conditions. In this example, set the value to target0.
Target Region	The region of the project to which the destination Logstore belongs. If you want to perform data transformation across regions, we recommend that you use HTTPS for data transmission. This ensures the privacy of log data. For cross-region data transformation, the data is transmitted over the Internet. If the Internet connections are unstable, data transformation latency may occur. You can select **DCDN Acceleration** to accelerate the cross-region data transmission. Before you can select DCDN Acceleration, make sure that the global acceleration feature is enabled for the project. For more information, see Enable the global acceleration feature. **Note** You are charged for the amount of Internet traffic that is generated when data after compression is transmitted across regions. For more information, see Billable items.
Target Project	The name of the project to which the destination Logstore belongs.
Target Logstore	The name of the destination Logstore.
	The method used to authorize the data transformation job to write transformed data to the destination Logstore. Valid values: ■ Default Role: authorizes the data transformation job to assume the system role AliyunLogETLRole to write transformed data to the destination Logstore. You must click **You must authorize the system role AliyunLogETLRole**. Then, you must configure other parameters as prompted to complete the authorization. For more information, see Access data by using a default role. **Note** ■ If you use a RAM user, you must use an Alibaba Cloud account to assign the AliyunLogETLRole role to the user. ■ If you use an Alibaba Cloud account that has assumed the role, you can skip this operation.

Parameter	Description
Authorization Method	■ Custom Role: authorizes the data transformation job to assume a custom role to write transformed data to the destination Logstore. You must grant the custom role the permissions to write to the destination Logstore. Then, you must enter the ARN of the custom role in the **Role ARN** field. For more information about authorization, see Access data by using a custom role. ■ AccessKey Pair: authorizes the data transformation job to use the AccessKey pair of an Alibaba Cloud account or a RAM user to write transformed data to the destination Logstore. ■ Alibaba Cloud account: The AccessKey pair of an Alibaba Cloud account has permissions to write to the destination Logstore. You can directly enter the AccessKey ID and AccessKey secret of the Alibaba Cloud account in the **AccessKey ID** and **AccessKey Secret** fields. For more information about how to obtain an AccessKey pair, see AccessKey pair. ■ RAM user: You must grant the RAM user the permissions to write to the destination Logstore. Then, you can enter the AccessKey ID and AccessKey secret of the RAM user in the **AccessKey ID** and **AccessKey Secret** fields. For more information about authorization, see Access data by using an AccessKey pair.
Processing Range	
Time Range	The time range within which the data is transformed. Valid values: **Note** The value of Time Range is based on the time when logs are received. ■ **All**: transforms data in the source Logstore from the first log entry until the job is manually stopped. ■ **From Specific Time**: transforms data in the source Logstore from the log entry that is received at the specified start time until the job is manually stopped. ■ **Within Specific Period**: transforms data in the source Logstore from the log entry that is received at the specified start time to the log entry that is received at the specified end time.

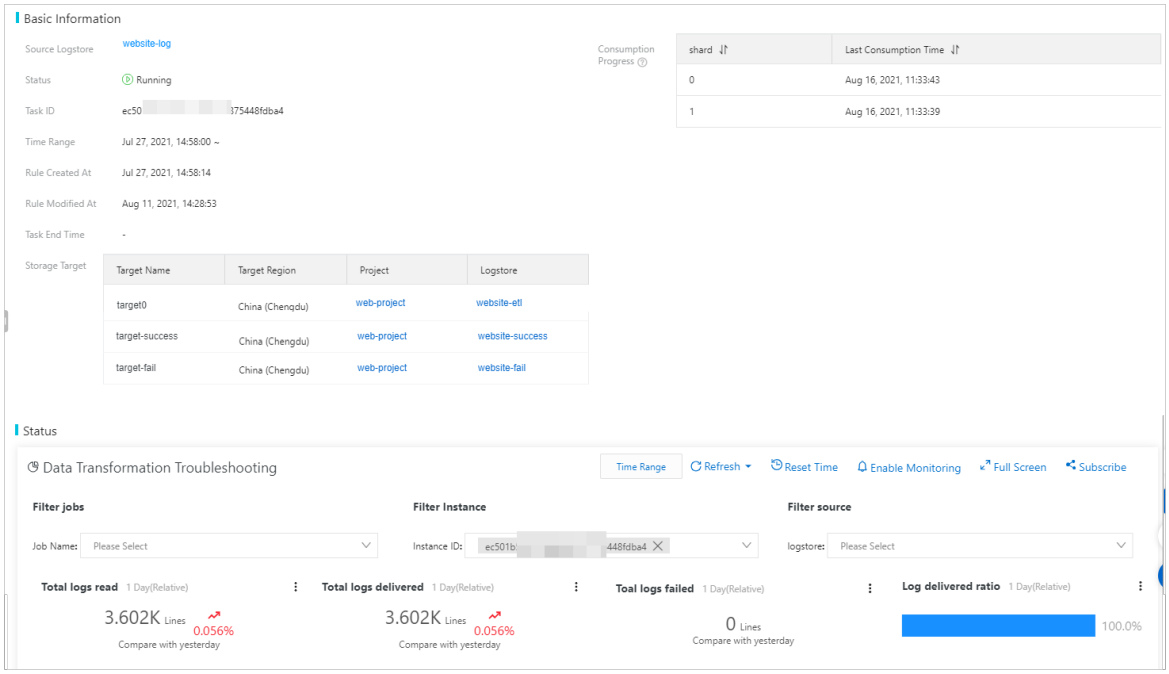
iii. Click **OK**.

After logs are distributed to the destination Logstores, you can perform query and analysis operations on the destination Logstores. For more information, see Query and analyze logs.

Step 2: View the data transformation job

1. In the left-side navigation pane, choose **Jobs > Data Transformation**.
2. In the data transformation job list, find and click the job.
3. On the **Data Transformation Overview** page, view the details of the job.

   You can view the details and status of the job. You can also modify, start, stop, or delete the job. For more information, see Manage a data transformation job.



7.Configure preview modes

7.1. Preview mode overview

You can use the preview feature to debug data transformation scripts. This feature supports the Quick preview mode and Advanced preview mode. This topic describes the two preview modes.

Quick preview mode

In Quick preview mode, you can check whether the syntax of a data transformation script is valid and whether data is efficiently transformed as expected. You can also use the raw logs of a Logstore or custom data as test data. You can perform the preceding operations in Quick preview mode free of charge. For more information, see Quick preview.

In Quick preview mode, Log Service cannot access the actual data that is specified by resource functions. The resource functions include res_local, res_rds_mysql, res_log_logstore_pull, and res_oss_file. If you use a resource function in a data transformation rule, you can enter test data on the **Dimension Table** tab to preview the data.

Advanced preview mode

In Advanced preview mode, Log Service accesses the specified Logstore and reads data from the Logstore to test the data and simulate the data transformation process. In Advanced preview mode, the specified transformation rule must be granted the required permissions to read data. The transformation speed in Advanced preview mode is slower than the transformation speed in Quick preview mode. You are charged for the traffic that is generated during the preview process. The traffic fee is included in the data transformation fee.

We recommend that you use the Quick preview mode to check whether the data is transformed as expected and the Advanced preview mode to check whether a resource function is configured as required. For more information, see Advanced preview.

7.2. Quick preview

In Quick preview mode, you can check whether the syntax of a data transformation script is valid and whether data is efficiently transformed as expected. You can perform the preceding operations in Quick preview mode free of charge. This topic describes how to configure the Quick preview mode.

Prerequisites

Data is collected. For more information, see Log collection methods.

Procedure

1.
2. Go to the data transformation page.
   i.
   ii.
   iii. On the query and analysis page, click **Data Transformation**.
3. In the upper-right corner of the page, select a time range for the required log data.

Make sure that the **Raw Logs** tab displays log data.

4. In the edit box, enter a data transformation statement.

   For more information, see Data processing syntax.

   > **Note** You can add comments to the data transformation statement to debug the statement line by line.

5. Preview data.

   i. In the upper-right corner of the page, click **Quick**.

   ii. Click the **Data Testing** tab.

   iii. On the **Data Testing** tab, enter test data.

   Test data includes basic data and dimension table data.

   - On the **Data** tab, you can enter or specify basic data.

     You can click the **Raw Logs** tab, find a log entry, and then click **Import Test Data**. The log entry is added as test data. You can also enter a test log entry.

     > **Note**
     > - The size of the test data for a single preview cannot exceed 1 MB.
     > - Each test log entry is separated by a blank line.
     > - The Markdown syntax is used to indicate cross-line field values. Triple backticks (```) are used to indicate a whole field.
     > - The test data that you enter or specify on the **Data** tab must be in the key-value pair or JSON format. Colons (:) are used to connect the field names and field values of data in the key-value pair format.

- Example 1: The test data includes two log entries. The first log entry is in the key-value pair format and contains a cross-line field named traceback. The second log entry is in the JSON format.

```
time_local: 25/May/2020:01:56:22
user agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/534.18
(KHTML, like Gecko) Chrome/11.0.661.0 Safari/534.18
"request:method": GET
```
```
traceback: Traceback (most recent call last):
  File "traceback_print_exc.py", line 20, in <module>
    produce_exception()
  File "/home/user/code/test.py", line 16, in produce_exception
    produce_exception(recursion_level-1)
  File "/home/user/code/test.py", line 18, in produce_exception
    raise RuntimeError()
RuntimeError
```
```
{
  "time_local": "25/May/2020:01:56:22",
  "user agent": "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/53
4.18 (KHTML, like Gecko) Chrome/11.0.661.0 Safari/534.18",
  "request:method": "GET",
  "remote user": "john"
}
```

- Example 2: The test data includes three log entries in the JSON format.

```
[
  {
    "time_local": "25/May/2020:01:56:22",
    "user agent": "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/
534.18 (KHTML, like Gecko) Chrome/11.0.661.0 Safari/534.18",
    "request:method": "GET",
    "remote user": "john"
  },
  {
    "time_local": "25/May/2020:01:56:22",
    "user agent": "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/
534.18 (KHTML, like Gecko) Chrome/11.0.661.0 Safari/534.18",
    "request:method": "GET",
    "remote user": "john"
  },
  {
    "time_local": "25/May/2020:01:56:22",
    "user agent": "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/
534.18 (KHTML, like Gecko) Chrome/11.0.661.0 Safari/534.18",
    "request:method": "GET",
    "remote user": "john"
  }
]
```

- On the **Dimension Table** tab, you can enter or specify dimension table data.

  You can use dimension tables to preview the resources that are accessed by resource functions. You can enter sample dimension table data for preview and debugging.

  > **Note** If you enter test data on the **Dimension Table** tab and use the res_rds_mysql or res_log_logstore_pull function to transform the data, the data must be in the CSV format. If you use the res_oss_file or res_local function, the data must be in the CSV or JSON format.

  Example: The test data includes two log entries. The first log entry is in the CSV format, and the second log entry is in the JSON format.

  ```
  ip,country,province
  127.0.0.1,China,Shanghai
  192.168.0.0,China,Zhejiang
  [
    {
      "ip": "127.0.0.1",
      "country": "China",
      "province": "Shanghai"
    },
    {
      "ip": "192.168.0.0",
      "country": "China",
      "province": "Zhejiang"
    }
  ]
  ```

iv. Click **Preview Data**.

> **Note** A maximum of 100 log entries can be returned for each data transformation test.

After you configure the preview settings, you can preview the data transformation results on the **Transformation Results** tab.

- If the data fails to be transformed because the syntax of the transformation statement or the permissions are invalid, troubleshoot the failure as prompted.
- If the data is transformed as expected, you can save the transformation statement as a rule. For more information, see Create a data transformation job.

Quick preview example

- Transformation statement

  Enter the following transformation statement in the editor:

```
# e_set("insert_field", "test_value")
e_table_map(
    res_rds_mysql(
        address="rm-uf6wjk5****.mysql.rds.aliyuncs.com",
        username="test_username",
        password="****",
        database="test_db",
        table="test_table",
    ),
    "ip",
    ["country", "province"],
)
```

- Test data

  On the **Data Testing > Data** tab, enter the following content:

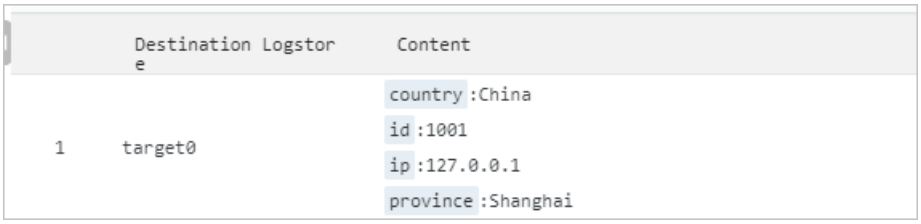
```
{
    "id": "1001",
    "ip": "127.0.0.1"
}
```

- Dimension table data

  On the **Data Testing > Dimension Table** tab, enter the following content:

```
ip,country,province
127.0.0.1,China,Shanghai
192.168.0.0,China,Zhejiang
```

- Preview transformation results



	Destination Logstore	Content
1	target0	country :China id :1001 ip :127.0.0.1 province :Shanghai

7.3. Advanced preview

In Advanced preview mode, Log Service accesses the specified Logstore and reads data from the Logstore to test the data and simulate the data transformation process. This topic describes how to configure the Advanced preview mode.

Procedure

1.
2. Go to the data transformation page.
   i.
   ii.
   iii. On the query and analysis page, click **Data Transformation**.
3. In the upper-right corner of the page, select a time range for the required log data.

Make sure that the **Raw Logs** tab displays log data.

4. In the edit box, enter a data transformation statement.

   For more information, see Data processing syntax.

   > **Note** You can add comments to the data transformation statement to debug the statement line by line.

5. Preview data.
   i. In the upper-right corner of the page, click **Advanced**.
   ii. Click **Preview Data**.
   iii. In the **Add Preview Settings** panel, set the parameters and click **OK**. The following table describes the parameters.

   When you preview data for the first time, you must set the parameters. After you set the parameters, you can click **Modify Preview Settings** to modify the parameters.

Parameter	Description

Parameter	Description
Authorization Method	You can authorize a data transformation task to read data from the current source Logstore by using one of the following methods: ■ Default Role: authorizes the data transformation job to assume the system role AliyunLogETLRole to read data from the source Logstore. You must click **You must authorize the system role AliyunLogETLRole**. Then, you must configure other parameters as prompted to complete the authorization. For more information, see Access data by using a default role. **Note** ■ If the authorization is complete within your Alibaba Cloud account, you can skip this operation. ■ If you use an Alibaba Cloud account that has assumed the role, you can skip this operation. ■ Custom Role: authorizes the data transformation job to assume a custom role to read data from the source Logstore. You must grant the custom role the permissions to read from the source Logstore. Then, you must enter the Alibaba Cloud Resource Name (ARN) of the custom role in the **Role ARN** field. For more information about authorization, see Access data by using a custom role. ■ AccessKey Pair: authorizes the data transformation job to use the AccessKey pair of an Alibaba Cloud account or a RAM user to read data from the source Logstore. ■ Alibaba Cloud account: The AccessKey pair of an Alibaba Cloud account has permissions to read from the source Logstore. You can directly enter the AccessKey ID and AccessKey secret of the Alibaba Cloud account in the **AccessKey ID** and **AccessKey Secret** fields. For more information about how to obtain an AccessKey pair, see AccessKey pair. ■ RAM user: You must grant the RAM user the permissions to read from the source Logstore. Then, you can enter the AccessKey ID and AccessKey secret of the RAM user in the **AccessKey ID** and **AccessKey Secret** fields. For more information about authorization, see Access data by using an AccessKey pair.

Parameter	Description
Advanced Parameter Settings	Log Service allows you to set the passwords that are required in the transformation statement in the key-value pair format. For example, you can set a password that is used to connect to a database in the key-value pair format. You can reference passwords in the transformation statement by using the `${key}` variable. You can click the plus sign (+) to add more key-value pairs. For example, you can add config.vpc.vpc_id.test1:vpc-uf6mskb0b****n9yj, which indicates the ID of the virtual private cloud (VPC) to which an ApsaraDB RDS instance belongs. 

After you configure the preview settings, you can preview the data transformation results on the **Transformation Results** tab.

- If the data fails to be transformed because the syntax of the transformation statement or the permissions are invalid, troubleshoot the failure as prompted.
- If the data is transformed as expected, you can save the transformation statement as a rule. For more information, see Create a data transformation job.

Advanced preview example

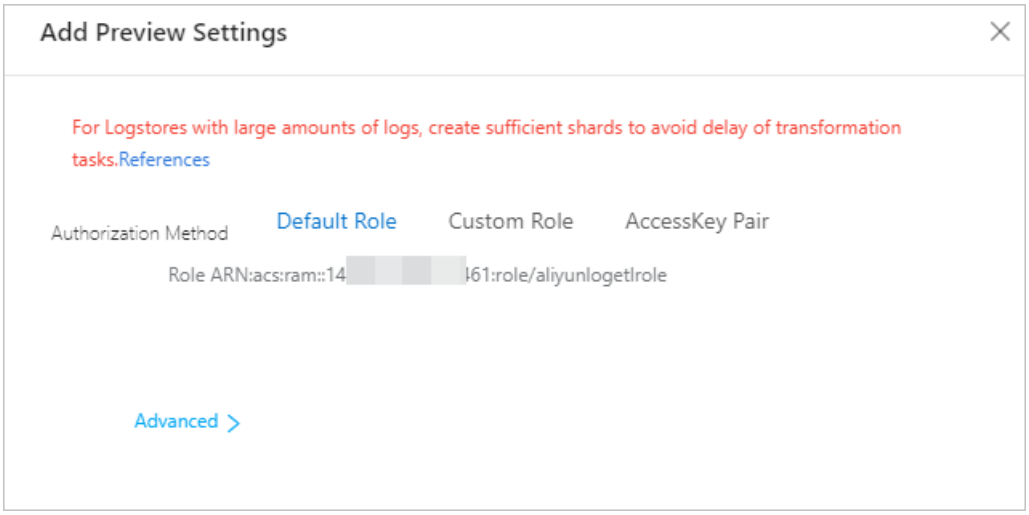
- Transformation statement

```
# e_set("insert_field", "test_value")
e_table_map(
    res_rds_mysql(
        address="rm-uf6wjk5****.mysql.rds.aliyuncs.com",
        username="test_username",
        password="*****",
        database="test_db",
        table="test_table",
    ),
    "ip",
    ["country", "province"],
)
```

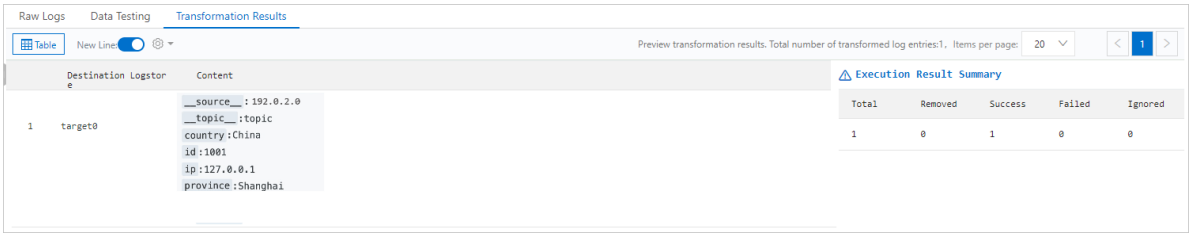
- Raw log entry

```
{
  "__source__": "192.0.2.0",
  "__time__": 1624523917,
  "__topic__": "topic",
  "id": "1001",
  "ip": "127.0.0.1"
}
```

- Advanced preview settings



- Transformation result



8.Create a data transformation job

Log Service allows you to create a data transformation job to read data from a source Logstore and write transformed data to one or more destination Logstores. You can also query and analyze the transformed data to create more value. This topic describes how to create a data transformation job in the Log Service console.

Prerequisites

- Log data is collected and stored in Log Service. For more information, see Data collection methods.
- If you use a Resource Access Management (RAM) user, the user is granted the permissions to perform related operations for data transformation. For more information, see Authorize a RAM user to manage a data transformation task.

Procedure

1.
2. Go to the data transformation page.
   i.
   ii.
   iii. On the query and analysis page, click **Data Transformation**.
3. In the upper-right corner of the page, select a time range for the required log data.

   Make sure that the **Raw Logs** tab displays log data.
4. In the editor, enter transformation statements.

   For more information about the statement syntax, see Language introduction.
5. Preview transformation results.
   i. Select **Quick**.

      You can select either Quick or Advanced. For more information, see Preview mode overview.
   ii. Click **Preview Data**.

      View the results.
      - If data fails to be transformed because the syntax of the transformation statements or configured permissions are invalid, troubleshoot the failure as prompted.
      - If the transformed data is returned as expected, go to .
6. Create a data transformation job.
   i. Click **Save as Transformation Rule**.
   ii. In the **Create Data Transformation Rule** panel, configure the following parameters and click **OK**.

Parameter	Description
Rule Name	The name of the transformation rule.

Parameter	Description
Authorization Method	The method used to authorize the data transformation job to read data from the source Logstore. Valid values: ■ Default Role: authorizes the data transformation job to assume the system role AliyunLogETLRole to read data from the source Logstore. You must click **You must authorize the system role AliyunLogETLRole**. Then, you must configure other parameters as prompted to complete the authorization. For more information, see Access data by using a default role. **Note** ■ If the authorization is complete within your Alibaba Cloud account, you can skip this operation. ■ If you use an Alibaba Cloud account that has assumed the role, you can skip this operation. ■ Custom Role: authorizes the data transformation job to assume a custom role to read data from the source Logstore. You must grant the custom role the permissions to read from the source Logstore. Then, you must enter the Alibaba Cloud Resource Name (ARN) of the custom role in the **Role ARN** field. For more information about authorization, see Access data by using a custom role. ■ AccessKey Pair: authorizes the data transformation job to use the AccessKey pair of an Alibaba Cloud account or a RAM user to read data from the source Logstore. ■ Alibaba Cloud account: The AccessKey pair of an Alibaba Cloud account has permissions to read from the source Logstore. You can directly enter the AccessKey ID and AccessKey secret of the Alibaba Cloud account in the **AccessKey ID** and **AccessKey Secret** fields. For more information about how to obtain an AccessKey pair, see AccessKey pair. ■ RAM user: You must grant the RAM user the permissions to read from the source Logstore. Then, you can enter the AccessKey ID and AccessKey secret of the RAM user in the **AccessKey ID** and **AccessKey Secret** fields. For more information about authorization, see Access data by using an AccessKey pair.
Storage Target	

Parameter	Description
Target Name	The name of the storage destination. Storage Target includes Target Project and Target Logstore. You can create multiple storage destinations to store the transformed data in different destination Logstores. ■ You can also use the name parameter of the e_output or e_coutput function in the transformation statements to specify the name of the storage destination. For more information, see e_output and e_coutput. ■ If you do not include the e_output function in the transformation statements, the job writes the transformed data to the Logstore in the storage destination that is numbered 1 by default. If you want to configure only one destination Logstore, you do not need to include the e_output function in the transformation statements. ■ If you include the e_output or e_coutput function and set the name, project, and logstore parameters for the function, the job runs based on the parameter settings in the functions even if you configure the Target Project and Target Logstore parameters in this step.
Target Region	The region of the project to which the destination Logstore belongs. If you want to perform data transformation across regions, we recommend that you use HTTPS for data transmission. This ensures the privacy of log data. For cross-region data transformation, the data is transmitted over the Internet. If the Internet connections are unstable, data transformation latency may occur. You can select **DCDN Acceleration** to accelerate the cross-region data transmission. Before you can select DCDN Acceleration, make sure that the global acceleration feature is enabled for the project. For more information, see Enable the global acceleration feature. **Note** You are charged for the amount of Internet traffic that is generated when data after compression is transmitted across regions. For more information, see Billable items.
Target Project	The name of the project to which the destination Logstore belongs.
Target Logstore	The name of the destination Logstore.

Parameter	Description
Authorization Method	The method used to authorize the data transformation job to write transformed data to the destination Logstore. Valid values: ■ Default Role: authorizes the data transformation job to assume the system role AliyunLogETLRole to write transformed data to the destination Logstore. You must click **You must authorize the system role AliyunLogETLRole**. Then, you must configure other parameters as prompted to complete the authorization. For more information, see Access data by using a default role. **Note** ■ If you use a RAM user, you must use an Alibaba Cloud account to assign the AliyunLogETLRole role to the user. ■ If you use an Alibaba Cloud account that has assumed the role, you can skip this operation. ■ Custom Role: authorizes the data transformation job to assume a custom role to write transformed data to the destination Logstore. You must grant the custom role the permissions to write to the destination Logstore. Then, you must enter the ARN of the custom role in the **Role ARN** field. For more information about authorization, see Access data by using a custom role. ■ AccessKey Pair: authorizes the data transformation job to use the AccessKey pair of an Alibaba Cloud account or a RAM user to write transformed data to the destination Logstore. ■ Alibaba Cloud account: The AccessKey pair of an Alibaba Cloud account has permissions to write to the destination Logstore. You can directly enter the AccessKey ID and AccessKey secret of the Alibaba Cloud account in the **AccessKey ID** and **AccessKey Secret** fields. For more information about how to obtain an AccessKey pair, see AccessKey pair. ■ RAM user: You must grant the RAM user the permissions to write to the destination Logstore. Then, you can enter the AccessKey ID and AccessKey secret of the RAM user in the **AccessKey ID** and **AccessKey Secret** fields. For more information about authorization, see Access data by using an AccessKey pair.
Processing Range	

Parameter	Description
Time Range	The time range within which the data is transformed. Valid values: ■ **All**: transforms data in the source Logstore from the first log entry until the job is manually stopped. ■ **From Specific Time**: transforms data in the source Logstore from the log entry that is received at the specified start time until the job is manually stopped. ■ **Within Specific Period**: transforms data in the source Logstore from the log entry that is received at the specified start time to the log entry that is received at the specified end time. **Note** The value of **Time Range** is based on the time when logs are received.
Advanced	
Advanced Parameter Settings	You may need to specify passwords, such as database passwords, in transformation statements. Log Service allows you to add a key-value pair to save the passwords. You can specify the `${key}` variable in the statements to reference the passwords. You can click the plus sign (+) to add more key-value pairs. For example, you can add config.vpc.vpc_id.test1:vpc-uf6mskb0b****n9yj, which indicates the ID of the virtual private cloud (VPC) to which an ApsaraDB RDS instance belongs. 

What's next

After the data transformation job is created, you can perform the following operations:

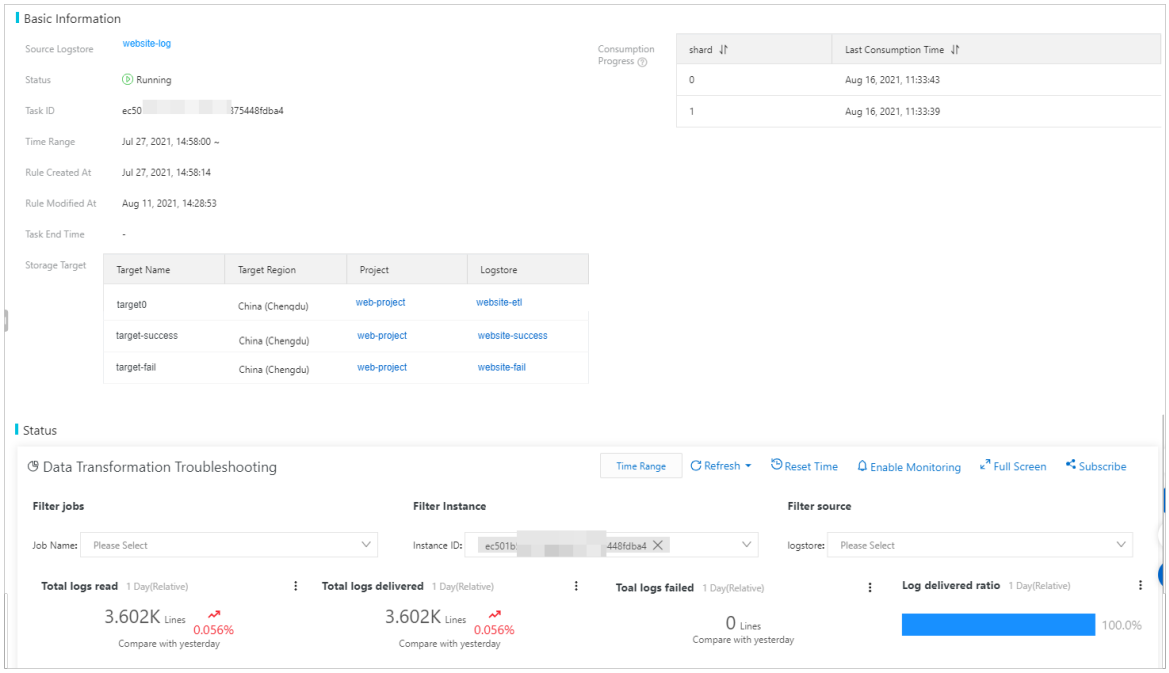
- On the **Data Transformation Overview** page, view the details and status of the job. You can also perform other operations, such as modifying or stopping the job. For more information, see Manage a data transformation job.
- In a destination Logstore, perform query and analysis operations. For more information, see Query and analyze logs.

9.Manage a data transformation job

This topic describes how to manage a data transformation job in the Log Service console. You can view the details and status of a job in the console. You can also start, stop, modify, and delete the job and configure alerts.

View the details of a job

1.
2. In the Projects section, find the project to which the job belongs and click the project.
3. In the left-side navigation pane, choose **Jobs > Data Transformation**.
4. In the data transformation job list, find and click the job.
5. On the **Data Transformation Overview** page, view the details of the job.



Parameter	Description
Source Logstore	The name of the Logstore from which data is read for transformation.
Status	The status of the job. For more information, see View the status of a job.
Task ID	The ID of the job.
Time Range	The time range within which the job is run.
Rule Created At	The time when the transformation rule was created.

Parameter	Description
Rule Modified At	The time when the transformation rule was last modified.
Task End Time	The time when the job ends. **Note** If you do not set End Time when you create the job, the value of this parameter is empty.
Storage Target	The name of the storage destination, the name of the Logstore where transformed data is stored, and the name of the project to which the Logstore belongs.
Consumption Progress	The data consumption progress of the job.
Status	The dashboard that is provided by Log Service to display the statistics about the execution of the transformation rule. The statistics include the overall information, transformation rate, consumption latency, consumption rate, active shards, and exceptions. For more information, see Data transformation dashboard.

View the status of a job

You can view the status of a data transformation job on the **Data Transformation Overview** page. The following table lists the job states and the operations supported in each state.

State/Action	Stop the job	Start the job	Rerun the job	Modify the job	Delete the job
Starting	Not supported	Not supported	Not supported	Supported	Supported
Running	Supported	Not supported	Supported	Supported	Supported
Stopping	Not supported	Not supported	Not supported	Supported	Supported
Terminated	Not supported	Supported	Supported	Supported	Supported
Success	Not supported	Not supported	Supported	Supported	Supported
Failed	Not supported	Not supported	Supported	Supported	Supported

Stop a job

If a job is in the **Running** state, you can click **Stop** on the **Data Transformation Overview** page of the job.

Note If a data transformation job is stopped, the system records the checkpoint at which the job is stopped. If the job is started again, data transformation is resumed from this checkpoint. If you want to resume data transformation from the beginning, you must rerun the job. For more information, see Rerun a job.

Start a job

If a job is in the **Terminated** state, you can click **Start** on the **Data Transformation Overview** page of the job.

Rerun a job

You can rerun a job regardless of the job status. You can choose **More > Rerun** on the **Data Transformation Overview** page of the job.

> **Note** If you rerun a job, the job is run from the beginning. If you want to continue with a job from the checkpoint at which the job was stopped, you must stop the job and then start the job.

Modify a transformation rule

> **Note** After you modify a transformation rule:
>
> - If you want to use the new transformation rule to continue with the job, you must stop the job and then start the job.
> - If you want to use the new transformation rule to rerun the job, you must run the job from the beginning. For more information, see Rerun a job.

1. On the **Data Transformation Overview** page, click **Edit Rules**.
2. Modify the transformation rule based on your business requirements.

   For more information about the rule syntax, see Language introduction.
3. Preview transformation results.
   i. Select **Quick**.

      You can select either Quick or Advanced. For more information, see Preview mode overview.
   ii. Click **Preview Data**.
   iii. View the results.
      - If data fails to be transformed because the syntax of the transformation rule or configured permissions are invalid, troubleshoot the failure as prompted.
      - If the transformed data is returned as expected, go to .
4. Click **Modify Processing Rule**.
5. Modify the settings and click **OK**.

   For more information about the parameters, see Create a data transformation job.

Delete a job

> **Warning** After you delete a data transformation job, the job cannot be recovered. Exercise caution when you delete a job.

You can delete a job by using one of the following methods:

- In the data transformation job list, move the pointer over the job that you want to delete and choose > **Delete**.

- On the **Data Transformation Overview** page, choose **More > Delete**.

Configure alerts

In the Data Transformation Troubleshooting dashboard of the Data Transformation Overview page, you can view the metrics of the data transformation job, subscribe to the dashboard, or configure monitoring and alerts for the metrics. For more information, see Enable monitoring for data transformation tasks.

10.Data transformation dashboard

After you create a data transformation task, a dashboard is automatically created for the task. You can view the metrics of the task on the dashboard. This topic describes the metrics of a data transformation task and how to view the metrics on the dashboard.

View the dashboard

After you create a transformation task, you can view the task on the **Data Transformation Overview** page of the task.

1.
2. In the Projects section, find the project to which the job belongs and click the project.
3. In the left-side navigation pane, choose **Jobs > Data Transformation**.
4. Click the target data transformation task. The Data Transformation Overview page of the task appears. In the **Status** section of the page, view the metrics of the task.

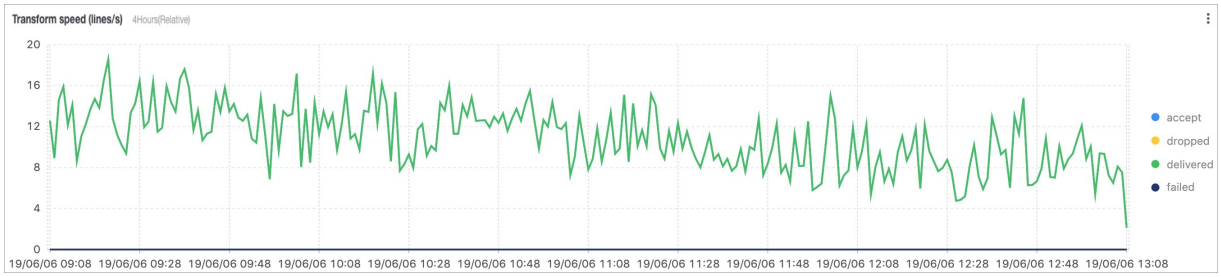
Overall metrics

The following figure shows the overall metrics.

- Total logs read: the total number of log entries that are read from all shards of the source Logstore.
- Total logs delivered: the total number of log entries that are read from all shards of the source Logstore and sent to the destination Logstore.
- Total logs failed: the total number of log entries that are read from all shards of the source Logstore and failed to be transformed.
- Log delivered ratio: the ratio of the log entries that are sent to the destination Logstore to the log entries that are read from the source Logstore.

Transform speed

This metric indicates the number of log entries that are transformed every minute.

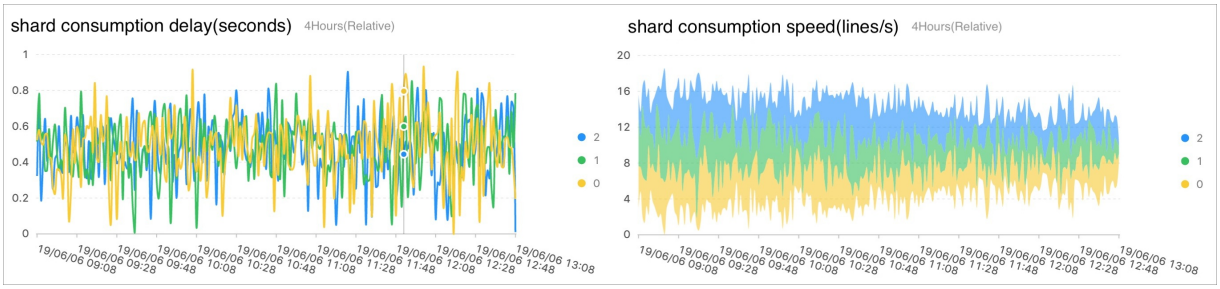


- accept: the number of log entries that are read from all shards of the source Logstore.
- dropped: the number of log entries that are read from all shards of the source Logstore and dropped as expected.
- delivered: the number of log entries that are read from all shards of the source Logstore and sent to the destination Logstore.

- failed: the number of log entries that are read from all shards of the source Logstore and failed to be transformed.

Shard consumption delay and shard consumption speed

These two metrics indicate the consumption delay and speed of each shard within a minute when the transformation task reads data from the source Logstore.



- Consumption delay: the difference between the time when logs in a shard are consumed and the time when the latest logs are written to the shard. The two time points are calculated based on the log receiving time.
- Shard consumption speed: the average number of log entries that are read by a shard per second within a minute.

> **Note** The delay of real-time consumption of logs is about 1 second. When historical logs are transformed, the shard consumption delay is high in the beginning, but is eventually reduced to a low level.

Active shard

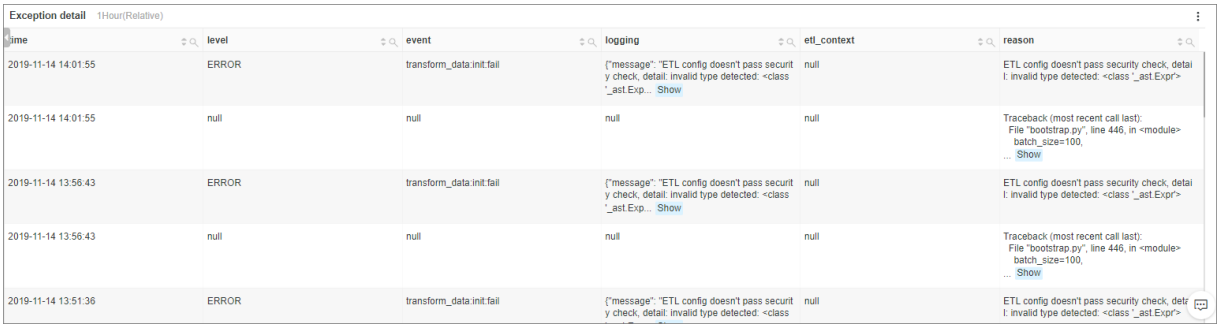
This metric indicates the number of accepted, dropped, delivered, and failed log entries per second in each shard within a specified period of time.


Active shard 1Hour(Relative)

project	logstore	shard	accept lines/s	dropped lines/s	delivered lines/s	failed lines/s
etl-test-beijing	test-1	2	346.1	0.0	346.1	0.0
etl-test-beijing	test-1	1	344.033	0.0	344.033	0.0
etl-test-beijing	test-1	0	342.9	0.0	342.9	0.0

Exception detail

In the reason field, you can view the code issues that may cause errors.


Exception detail 1Hour(Relative)

time	level	event	logging	etl_context	reason
2019-11-14 14:01:55	ERROR	transform_data:init:fail	{"message": "ETL config doesn't pass security check, detail: invalid type detected: <class '_ast.Exp... Show	null	ETL config doesn't pass security check, detail: invalid type detected: <class '_ast.Expr'>
2019-11-14 14:01:55	null	null	null	null	Traceback (most recent call last): File "bootstrap.py", line 446, in <module> batch_size=100, ... Show
2019-11-14 13:56:43	ERROR	transform_data:init:fail	{"message": "ETL config doesn't pass security check, detail: invalid type detected: <class '_ast.Exp... Show	null	ETL config doesn't pass security check, detail: invalid type detected: <class '_ast.Expr'>
2019-11-14 13:56:43	null	null	null	null	Traceback (most recent call last): File "bootstrap.py", line 446, in <module> batch_size=100, ... Show
2019-11-14 13:51:36	ERROR	transform_data:init:fail	{"message": "ETL config doesn't pass security check, detail: invalid type detected: <class	null	ETL config doesn't pass security check, detail: invalid type detected: <class '_ast.Expr'>

You can also access the **internal-etl-log** Logstore in the current project and use the ERROR or WARNING keyword to view complete error logs.

11.Enable monitoring for data transformation tasks

After you enable monitoring for data transformation tasks, Log Service sends alert notifications when exceptions occur during data transformation. This helps you handle exceptions in a timely manner. This topic describes how to enable monitoring for data transformation tasks.

Prerequisites

A data transformation task is created. For more information, see Create a data transformation job.

Context

- After you create a data transformation rule, Log Service automatically creates a dashboard named Data Transformation Troubleshooting for the data transformation task. We recommend that you take note of the following metrics on the Data Transformation Troubleshooting dashboard:
  - System metrics: the data consumption delay and relevant exceptions.
  - Application metrics: the number of received log entries and number of delivered log entries.

  For more information, see Data transformation dashboard.
- Log Service provides built-in monitoring rules, action policy, and alert templates for data transformation. You can use built-in resources based on the following rules:
  - Alert Center provides built-in monitoring rules for data transformation. You can enable the alert instance of a monitoring rule to configure alerts. You do not need to write SQL statements. You can enable the alert instances of various monitoring rules, such as the rule that triggers an alert when delay, exceptions, or failures occur during data transformation. For more information, see Monitoring rules for data transformation.
  - You can specify notification methods and alert templates in the built-in action policy for data transformation.
  - You can specify the content of alert notifications in a built-in alert template for data transformation.

Configuration process

You can use built-in resources or custom resources to configure alerts based on the following process:

- Use built-in resources.

  To configure alerts in an efficient manner, perform the following operations:

  i. Create a DingTalk chatbot.

     Configure a DingTalk chatbot to receive alert notifications.

  ii. Configure an action policy.

     Specify the webhook URL of the preceding DingTalk chatbot for the built-in action policy for data transformation. Log Service sends alert notifications by using the webhook URL.

  iii. Enable alert instances.
- Use custom resources.

To create custom resources and use them to configure recipients, alert templates, and notification methods based on your business requirements, perform the following operations:

i. Create users.

   Configure users or user groups to receive alert notifications. For more information, see Create users and user groups.

ii. Create an alert template.

   Configure the content of alert notifications. For more information, see Create an alert template.

iii. Create an action policy.

   Configure notification methods, such as Voice Call, SMS Message, and Email. For more information, see Create an action policy.

iv. Enable alert instances.

The built-in resources that are provided by Log Service can be applied to most alerting scenarios. You can use built-in resources or custom resources based on your business requirements. In this example, built-in resources are used to configure alerts.

Step 1: Create a DingTalk chatbot

By default, the built-in action policy for data transformation uses DingTalk-Custom as the notification method to send alert notifications. Before you enable monitoring, you must create a DingTalk chatbot. After an alert is triggered, Log Service sends an alert notification to the specified DingTalk group by using the webhook URL of the DingTalk chatbot.

1. Create a DingTalk chatbot.
   i. Open DingTalk and go to a DingTalk group.
   ii. In the upper-right corner of the chat window, click the **Group Settings** icon and choose **Group Assistant > Add Robot**.
   iii. In the **ChatBot** dialog box, click the **+** icon in the **Add Robot** section.
   iv. In the Robot details dialog box, select **Custom (Custom message services via Webhook)** and click **Add**.
   v. In the **Add Robot** dialog box, enter a chatbot name in the **Chatbot name** field and select security options in the **Security Settings** section based on your business requirements. Then, select the **I have read and accepted DingTalk Custom Robot Service Terms of Service** check box and click **Finished**.

      > **Note** We recommend that you set the **Security Settings** parameter to **Custom Keywords**. You can set up to 10 keywords. The chatbot sends only messages that contain at least one of the specified keywords. We recommend that you specify **Alert** as a keyword.

   vi. Click **Copy** to copy the webhook URL.

Step 2: Configure an action policy

Modify the request URL of the DingTalk-Custom notification method for the built-in action policy for data transformation. This way, Log Service sends alert notifications by using the specified webhook URL of the DingTalk chatbot.

1.

2. Go to the Action Policy tab.
   i. In the Projects section, click the project in which you created the data transformation task.
   ii. In the left-side navigation pane, click **Alerts**.
   iii. Click **Open Alert Center** and choose **Alert Management > Action Policy**.
3. On the **Action Policy** tab, click **Modify** in the Actions column of the built-in action policy whose ID is **sls.app.etl.builtin**.
4. In the **Edit Action Policy** dialog box, click the **Primary Action Policy** tab and set the **Request URL** parameter under **DingTalk-Custom** to the webhook URL that is obtained in Step 1: Create a DingTalk chatbot. Then, click **OK**.

Step 3: Enable alert instances

1.
2.
3. In the left-side navigation pane, click **Alerts**.
4. Click **Open Alert Center**.
5. On the **Alert Rules/Incidents** tab, select **SLS Data Transformation** in the Type section.
6. In the monitoring rule list, find the monitoring rule whose alert instances you want to enable and click **Enable**.

   After you enable an alert instance, Log Service monitors all data transformation tasks in real time by default.

   - To enable multiple alert instances, click **Add**.
   - If you need to monitor only specific data transformation tasks, click **Settings** and specify the IDs of the data transformation tasks that you want to monitor.

   For more information, see Related operations.

   For information about the parameters of monitoring rules, see Monitoring rules for data transformation.

Related operations

Operation	Description
Configure whitelists	You can configure whitelists for specific monitoring rules. This way, alerts are not triggered by specified data transformation tasks.
Add alert instances	You can add an alert instance of a monitoring rule. You can add an alert instance and configure its settings to monitor specific data transformation tasks.
Disable alert instances	If you disable an alert instance, the status in the **Status** column of the alert instance changes to **Not Enabled**, and no more alerts are triggered based on the alert instance. The configurations of the alert instance are not deleted. If you want to re-enable the alert instance to monitor data transformation tasks, you do not need to reconfigure the parameters of the alert instance.

Operation	Description
Pause alert instances	If you pause an alert instance, no alerts are triggered within a specified period of time based on the alert instance.
Resume alert instances	You can resume paused alert instances.
Delete alert instances	If you delete an alert instance, the status in the **Status** column of the alert instance changes to **Not Created**. The configurations of the alert instance are deleted, such as the settings of data transformation tasks. If you want to re-enable the alert instance to monitor data transformation tasks, you must set the parameters of the alert instance again.
Modify alert instances	You can modify the parameters of an alert instance, such as the alert name, the IDs of data transformation tasks that you want to monitor, monitoring threshold, action policy, and severity.

Monitoring rules for data transformation

Log Service provides the following built-in monitoring rules for data transformation. For information about how to manage monitoring rules, see Related operations.

- Delay Monitoring during Data Transformation
- Exception Monitoring during Data Transformation
- Traffic Monitoring during Data Transformation (Absolute Value)
- Traffic Monitoring during Data Transformation (Compared with Previous Day)
- Failure Monitoring during Data Transformation

The following tables describe the functionalities, parameters, and associated dashboard metrics of the Log Service built-in monitoring rules for data transformation. The table also provides the handling methods that are used to clear alerts.

- Delay monitoring rule during data transformation

Item	Description
Rule name	Delay Monitoring during Data Transformation
Functionality	This rule monitors the delay that may occur when data is consumed from shards in data transformation tasks. If the delay during data transformation exceeds the value of the Monitoring Threshold parameter, an alert is triggered.

Item	Description
Parameters	◦ Data Transformation ID: the ID of the data transformation task that you want to monitor, for example, dd2de8e7e23f3e42ffbb32fe05710372. Default value: .*. This value indicates that all data transformation tasks are monitored. Separate multiple task IDs with vertical bars (|). ◦ Monitoring Threshold: If the delay during data consumption exceeds this value, an alert is triggered. Default value: 300. Unit: seconds. ◦ Action Policy: the action policy that is used to send alert notifications. The action policy contains notification methods and alert templates. The default value is the built-in action policy whose ID is sls.app.etl.builtin. This value indicates that alert notifications are sent by using the webhook URL of a DingTalk chatbot. ◦ Severity: the severity of the alert. Valid values: **Critical**, **High**, **Medium**, **Low**, and **Report**. Default value: **High**.
Associated dashboard	**Data Transformation Troubleshooting** > **shard consumption delay (seconds)**
Handling method	You can clear triggered alerts based on the following rules: i. If the amount of data in the source Logstore significantly increases, perform the following operations as needed: ■ If the value of the **Transform speed (lines/s)** metric increases at the same time and the value of the **shard consumption delay (seconds)** metric decreases, this indicates that the data transformation task is automatically scaling up resources due to the increasing data volume in the source Logstore. In this case, wait for 5 minutes and then check whether the delay is lower than the specified threshold. If not, proceed to the next step. ■ If the value of the **Transform speed (lines/s)** metric does not increase or the value of the **shard consumption delay (seconds)** metric continues to increase, this indicates that the number of shards in the source Logstore may be insufficient and the expansion of resources for data transformation is limited. In this case, you must split the shards in the source Logstore. For more information, see Split a shard. After you split the shards, wait for 5 minutes and then check whether the delay is lower than the specified threshold. If not, proceed to the next step. ii. If alerts are triggered based on the **Exception Monitoring during Data Transformation** rule, you must clear the alerts first. After you clear the alerts, wait for 5 minutes and then check whether the delay is lower than the specified threshold. If not, proceed to the next step. iii. If you cannot clear the alerts, prepare the information of the related project, Logstore, and the data transformation task ID, and then submit a ticket for Alibaba Cloud technical support.

- Exception monitoring rule during data transformation

Item	Description
Rule name	Exception Monitoring during Data Transformation
Functionality	This rule monitors exceptions in data transformation tasks. If an exception occurs during data transformation, an alert is triggered.
Parameters	◦ Data Transformation ID: the ID of the data transformation task that you want to monitor, for example, dd2de8e7e23f3e42ffbb32fe05710372. Default value: .*. This value indicates that all data transformation tasks are monitored. Separate multiple task IDs with vertical bars (|). ◦ Action Policy: the action policy that is used to send alert notifications. The action policy contains notification methods and alert templates. The default value is the built-in action policy whose ID is sls.app.etl.builtin. This value indicates that alert notifications are sent by using the webhook URL of a DingTalk chatbot. ◦ Severity: the severity of the alert. Valid values: **Critical**, **High**, **Medium**, **Low**, and **Report**. Default value: **High**.
Associated dashboard	**Data Transformation Troubleshooting** > **Exception detail**
Handling method	Fix exceptions based on the related error messages. ◦ If the error message contains Unauthorized, InvalidAccessKeyId, or SignatureNotMatch, the data transformation task does not have the required permissions to read data from the source Logstore or write data to the destination Logstore. For more information, see Authorization overview. ◦ If the error message contains ProjectNotExist or LogStoreNotExist, the related project or Logstore of the data transformation task does not exist. In this case, log on to the Log Service console to identify and fix the error. ◦ If the error message contains SettingError, the configuration of the data transformation rule is invalid. For example, if the specified parameters in a function is invalid or the configuration of an external resource such as Object Storage Service (OSS) or ApsaraDB RDS for MySQL is invalid, an error occurs. For more information, see Function overview. ◦ If the error message contains TransformError, the raw data in the source Logstore does not meet the logic of the current data transformation rule. This error may occur when new types of data are imported to the source Logstore. In this case, identify the raw data from the error message, update the data transformation task, and then try again. For more information, see Manage a data transformation job.

- Traffic monitoring rule during data transformation (absolute value)

Item	Description
Rule name	Traffic Monitoring during Data Transformation (Absolute Value)

Item	Description
Functionality	This rule monitors the average number of log entries that are transformed by data transformation tasks within 5 minutes. If the average number of log entries that are transformed is lower than the value of the Monitoring Threshold parameter, an alert is triggered.
Parameters	◦ Data Transformation ID: the ID of the data transformation task that you want to monitor, for example, dd2de8e7e23f3e42ffbb32fe05710372. Default value: .*. This value indicates that all data transformation tasks are monitored. Separate multiple task IDs with vertical bars (|). ◦ Monitoring Threshold: If the average number of transformed log entries is lower than this value, an alert is triggered. Default value: 40000. Unit: lines/s. ◦ Action Policy: the action policy that is used to send alert notifications. The action policy contains notification methods and alert templates. The default value is the built-in action policy whose ID is sls.app.etl.builtin. This value indicates that alert notifications are sent by using the webhook URL of a DingTalk chatbot. ◦ Severity: the severity of the alert. Valid values: **Critical**, **High**, **Medium**, **Low**, and **Report**. Default value: **High**.
Associated dashboard	**Data Transformation Troubleshooting** > **Transform speed (lines/s)**
Handling method	You can clear triggered alerts based on the following rules: i. If the value change trend of the Transform speed (lines/s) metric is consistent with the increase or decrease trend of the data volume in the source Logstore, this indicates that the number of transformed log entries is limited by the data volume in the source Logstore. If not, proceed to the next step. ii. If alerts are triggered based on the **Delay Monitoring during Data Transformation** rule, you must clear the alerts first. After you clear the alerts, wait for 15 minutes. If the delay is less than 1 minute but the amount of the transformed data is inconsistent with the increase or decrease trend of the data volume in the source Logstore, proceed to the next step. iii. If you cannot clear the alerts, prepare the information of the related project, Logstore, and the data transformation task ID, and then submit a ticket for Alibaba Cloud technical support.

- Traffic monitoring rule during data transformation (compared with previous day)

Item	Description
Rule name	Traffic Monitoring during Data Transformation (Compared with Previous Day)
Functionality	This rule monitors the increase rate and decrease rate of the transformed data compared with the same period of the previous day in data transformation tasks. If the increase rate is greater than the value of the Daily Increase Threshold or the decrease rate is greater than the value of the Daily Decrease Threshold parameter, an alert is triggered.

Item	Description
Parameters	◦ Data Transformation ID: the ID of the data transformation task that you want to monitor, for example, dd2de8e7e23f3e42ffbb32fe05710372. Default value: .*. This value indicates that all data transformation tasks are monitored. Separate multiple task IDs with vertical bars (|). ◦ Daily Increase Threshold: If the daily increase rate of transformed data is greater than this value, an alert is triggered. Default value: 40%. ◦ Daily Decrease Threshold: If the daily decrease rate of transformed data is greater than this value, an alert is triggered. Default value: 20%. ◦ Action Policy: the action policy that is used to send alert notifications. The action policy contains notification methods and alert templates. The default value is the built-in action policy whose ID is sls.app.etl.builtin. This value indicates that alert notifications are sent by using the webhook URL of a DingTalk chatbot. ◦ Severity: the severity of the alert. Valid values: **Critical**, **High**, **Medium**, **Low**, and **Report**. Default value: **High**.
Associated dashboard	**Data Transformation Troubleshooting** > **Transform speed (lines/s)**
Handling method	You can clear triggered alerts based on the following rules: i. If the value change trend of the Transform speed (lines/s) metric is consistent with the increase or decrease trend of the data volume in the source Logstore, this indicates that the number of transformed log entries is limited by the data volume in the source Logstore. If not, proceed to the next step. ii. If alerts are triggered based on the **Delay Monitoring during Data Transformation** rule, you must clear the alerts first. After you clear the alerts, wait for 15 minutes. If the delay is less than 1 minute but the amount of the transformed data is inconsistent with the increase or decrease trend of the data volume in the source Logstore, proceed to the next step. iii. If you cannot clear the alerts, prepare the information of the related project, Logstore, and the data transformation task ID, and then submit a ticket for Alibaba Cloud technical support.

- Monitoring rule for the number of log entries that fail to be transformed during data transformation

Item	Description
Rule name	Failure Monitoring during Data Transformation
Functionality	This rule monitors the failures of data transformation tasks within 15 minutes. If the number of log entries that fail to be transformed during data transformation exceeds the value of the Monitoring Threshold parameter, an alert is triggered.

Item	Description
Parameters	◦ Data Transformation ID: the ID of the data transformation task that you want to monitor, for example, dd2de8e7e23f3e42ffbb32fe05710372. Default value: .*. This value indicates that all data transformation tasks are monitored. Separate multiple task IDs with vertical bars (|). ◦ Monitoring Threshold: If the number of log entries that fail to be transformed exceeds this value, an alert is triggered. Default value: 10. ◦ Action Policy: the action policy that is used to send alert notifications. The action policy contains notification methods and alert templates. The default value is the built-in action policy whose ID is sls.app.etl.builtin. This value indicates that alert notifications are sent by using the webhook URL of a DingTalk chatbot. ◦ Severity: the severity of the alert. Valid values: **Critical**, **High**, **Medium**, **Low**, and **Report**. Default value: **High**.
Associated dashboard	**Data Transformation Troubleshooting** > **Total logs failed**
Handling method	You can clear triggered alerts based on the following rules: i. Clear the alerts by using the method that is provided by the **Exception Monitoring during Data Transformation** rule. If no error message is reported, proceed to the next step. ii. If you cannot clear the alerts, prepare the information of the related project, Logstore, and the data transformation task ID, and then submit a ticket for Alibaba Cloud technical support.

12.Data processing syntax

12.1. Language introduction

LOG domain specific language (DSL) is a Python-compatible script language that is used for the data processing feature of Log Service. Based on Python, LOG DSL provides over 200 built-in functions to simplify data processing.

Flexible orchestration

LOG DSL allows you to flexibly combine functions to implement complex logic.

Global processing functions

LOG DSL provides about 30 global processing functions. You can configure the parameters of global processing functions to control operations in data processing steps. Global processing functions can use expression functions as parameters. As a special type of global processing functions, process control functions can also use other global processing functions as parameters.

- Process control functions
  - Allow you to control processes through condition-based judgment such as `if-else`, `if`, `switch`, and `compose`.
  - Allow you to call simple search functions such as `e_search` to process logs of different types flexibly.
- Event processing functions

  Allow you to discard, retain, split, write, and replicate events.
- Field processing functions

  Allow you to retain, delete, and rename fields.
- Value assignment function

  Allows you to assign the results of any expression functions or their combinations as values to fields.
- Value extraction functions
  - Allow you to extract values or key-value pairs from fields based on regular expressions, Grok patterns, Syslog standard, key-value pair delimiters, and value delimiters such as commas (,), vertical bars (|), and tabs (\t).
  - Allow you to extract and enrich JSON data.
- Mapping and enrichment functions
  - Allow you to map fields to or search for data in dictionaries and tables.
  - Allow you to obtain dimension tables for enrichment from resources such as rule configuration data, Object Storage Service (OSS) buckets, Relational Database Service (RDS) databases, and Logstores.
  - Allow you to refresh external resources based on full or incremental logs.

Expression functions

LOG DSL provides over 200 built-in expression functions for you to convert events or control global processing functions. The expression functions cover most data processing scenarios. LOG DSL provides the following expression functions:

- Event search function: allows you to search for events based on regular expressions, strings, wildcard characters, value comparison, and logical operations such as AND, OR, and NOT. The syntax is similar to that of Lucene.
- Basic processing functions: support field value extraction, control, comparison, containment judgment, and multi-field operations.
- Conversion functions: convert values among different types, such as numbers, dictionaries, and lists.
- Arithmetic functions: support basic calculation, multi-value calculation, mathematical calculation, and mathematical parameters.
- String functions: support multi-field operations and other operations such as encoding, decoding, sorting, reversing, replacing, normalizing, searching, judging, slicing, and formatting strings.
- Date and time functions: convert date and time strings, obtain date and time attributes, obtain date and time, obtain UNIX timestamps, obtain date and time as strings, and modify and compare date and time.
- Regular expression functions: extract, match, judge, replace, and split fields based on regular expressions.
- Grok function: supports over 400 built-in Grok patterns and pattern replacement.
- JSON, Protobuf, and XML functions: extract and filter data.
- Encoding and decoding functions: encode and decode text in the SHA1, SHA256, SHA512, MD5, HTML, URL, and Base64 formats.

Dynamic distribution

LOG DSL can distribute data to different destination Logstores based on the specified logic. The names of the destination Logstores can be obtained through dynamic computing or from external resources.

Flexible data enrichment

- Allows you to obtain data from local or external resources such as OSS buckets, RDS databases, and Logstores, and use the data to enrich logs.
- Supports common mapping for tables and dictionaries and advanced mapping based on table search.
- Automatically refreshes external resources that are loaded.

12.2. Syntax introduction

The domain-specific language (DSL) for Log Service provides different types of functions to meet the data transformation and processing needs in different scenarios.

Language modes

The DSL for Log Service is compatible with Python. In standard mode, the DSL can be regarded as a subset of Python. Except for basic data structures and expressions, other syntax rules are orchestrated by using functions. If you want to use user-defined functions (UDFs), submit a ticket.

Category	Python syntax	Standard mode
Data structure	Number, string, and Boolean	Supported, except for strings in the `"""` form.
	Tuple, list, set, and dictionary	Supported, except for the `set` structure, such as `{1,2,3}`.
	Object	Only built-in extended data structures such as table and datetime objects are supported.
Basic syntax	Operators, such as + (addition), - (subtraction), * (multiplication), and / (division)	Not supported directly. You must use the operators in functions.
	Comments	Supported.
	Variable assignment	Not supported. You must call functions to pass values to variables.
	Condition evaluation and loops	Not supported directly. You must use built-in functions to implement condition evaluation and loops.
Function	Standard built-in functions of Python	Not supported. You can use more than 200 built-in functions provided by the DSL for Log Service.
	Function calls	Supported, except for function calls with parameter unpacking.
	UDFs, such as def or lambda	Not supported. You can use more than 200 global processing and expression functions provided by the DSL for Log Service. You can also combine these functions to suite your needs.
Module	Import and use of the Python standard library	Not supported.
	Creation of threads and processes	Not supported.
	Import of third-party libraries	Not supported.

Category	Python syntax	Standard mode
	External network connection or external command call	Supported. The DSL for Log Service provides built-in resource connectors.

Function categories

In standard mode of the DSL for Log Service, all operations are implemented by calling functions. The DSL provides more than 200 built-in functions, which are categorized into global processing functions and expression functions.

- Global processing functions

  Global processing functions receive, process, and return events. Only global processing functions can be used to construct each step of a transformation rule.

- Expression functions

  Expression functions are general functions that receive specific parameters and return specific values. Expression functions can be combined and passed to global processing functions as parameters to define more flexible logic.

The following table compares global processing functions with expression functions.

Category	Construct a step	Receive an event	Return results	Modify an event	Combine functions
Global processing function	Supported.	Events are automatically received.	Zero to multiple events are returned.	Supported. Events can be modified in most cases.	Supported.
Expression function	Not supported.	Supported only by a few expression functions. Most expression functions do not directly process events.	Specific data structures are returned.	Not supported.	Supported.

Global processing functions

Global processing functions receive, process, and return events.

> **Note** Only a global processing function can be placed in the first line of each step.

The following syntax is used:

```
Global Processing Function 1(...Parameters...)
Global Processing Function 2(...Parameters...)
Global Processing Function 3(...Parameters...)
Global Processing Function 4(...Parameters...)
```

Global processing functions can be further categorized into flow control functions and event processing functions.

Category	Description	Example
Flow control function	This type of function controls processes, receives events, and calls event processing functions based on conditions.	`e_if` , `e_switch` , and `e_if_else`
Event processing function	This type of function processes events. Zero to multiple events are returned.	Examples: • `e_drop_fields` : discards fields of events. • `e_kv` : extracts and adds key-value pairs of events. • `e_dict_map` : enriches events.

Transformation logic:

- Basic processing

  The data transformation feature reads streaming data from a source Logstore and sends each log event in a dictionary structure to a transformation rule. Then, the feature runs the event processing functions defined in the transformation rule in sequence to process the events and writes the processing results to the destination Logstores.

  > **Note** All the fields and values of an event are transmitted as strings. For example, the log event `{"__time__": "1234567", "__topic__": "", "k1": "test"}` is processed by the function `e_set("f1", 200)` , which adds the field `f1` with a value of *200*. After processing, the event `{"__time__": "1234567", "__topic__": "", "k1": "test", "f1": "200"}` is returned. `f1` and *200* are both strings.

  Event processing functions defined in a transformation rule are called in sequence. Each function receives and processes an event and returns a processed event.

  For example, the function `e_set("type", "test")` adds the field `type` to each event and sets the value to *test*. The next function receives the processed event.

- Condition evaluation
  - e_if: You can call the e_if function to add conditional expressions to process events. If an event does not meet the specified condition, the operation is skipped. The e_if function implements the `if` logic.

    For example, the function `e_if(e_match("status", "200"), e_regex("data", "ret: \d +", "result"))` first determines whether the value of the `status` field is *200*. If the value is 200, the function extracts a new field `result` from the `data` field by using the regular expression. If the value is not 200, the extract operation is not performed.
  - `e_if_else` : This function works in a similar way to the `if_else` function.

- Processing termination

- Some steps of a transformation rule may return no events. This indicates that the event is deleted.

  For example, if the function `e_if(str_islower(v("result")), e_drop())` is used and the value of each `result` field in an event is a string consisting of only lowercase characters, the function deletes this event. Then, the subsequent steps are not performed on this event. The system automatically processes the next event.

- If an event is written to a destination Logstore, the processing is terminated. For example, the `e_output` function writes an event to a destination Logstore and deletes the event, and the subsequent steps are not performed on this event.

  > **Note** However, the `e_coutput` function writes an event to a destination Logstore but retains the original event, and the subsequent steps are performed on this event.

- Event splitting for parallel processing

  Some steps of a transformation rule may return multiple events. This indicates that the original event is split.

  For example, the function `e_split(data)` splits an event into two events based on the value of the `data` field. If the value of the `data` field in the original event is `"abc, xyz"`, the values of the `data` field in the events generated after splitting are *abc* and *xyz*.

  All the events generated after splitting are processed in the subsequent steps.

Expression functions

In addition to global processing functions, the DSL for Log Service provides 200 expression functions that receive specific parameters and return specific values. You can call a single expression function or a combination of expression functions in a global processing function. The following syntax is used:

```
Global Processing Function 1(Expression Function 1(...), ...)
Global Processing Function 2(..., Expression Function 2(...), Expression Function 3(...), ...)
```

Expression functions can be categorized into event check functions, resource functions, control functions, and other expression functions.

Category	Description	Example
Event check function	This type of function receives events, extracts specific information, and returns the information. Events are not modified.	`v`: returns the values of fields. `e_search` and `e_match`: check whether events meet the specified conditions.
Resource function	This type of function accesses on-premises or external resources, receives specific parameters, and returns specific values. The data types of return values include dictionary and table.	res_oss_file, res_rds_mysql, and res_log_logstore_pull.

Category	Description	Example
Control function	This type of function receives specific parameters and performs logical operations on expressions or condition-based control. This type of function also calls other expression functions to return results.	`op_and`, `op_or`, `op_not`, `op_if`, and `op_coalesce`.
Other expression function	This type of function receives specific parameters or the results of other functions and returns specific values.	String functions, date and time functions, and conversion functions.

12.3. Data structures

This topic describes the data structures in data transformation syntax.

Basic data structures

The following table describes the basic data structures.

Type	Description
Integer	You can use integers as field values. You can also use integers to pass the values of function parameters. For example, the `e_set("f1", 100)` function is used to set the value of the `f1` field to *100*.
Float	You can use floats as field values. You can also use floats to pass the values of function parameters. For example, the `e_set("f1", 1.5)` function is used to set the value of the `f1` field to *1.5*.

Type	Description
String	Strings can be used in multiple formats. For example: • `"abc"` is equivalent to `'abc'`. If a string contains a double quotation mark ("), you can write the string in the `'abc"xyz'` format. You can also use a backslash (\) to escape the double quotation mark in the following format: `"abc\"xyz"`. Backslashes (\) are used to escape special characters. For example, `"\\abc\\xyz"` refers to the `\abc\xyz` string. • Both `r"\\10.64.1.1\share\folder"` and `"\\\\10.64.1.1\\share\\folder"` refer to the `\\10.64.1.1\share\folder` string. • Multibyte character strings are encoded in Unicode. For example, the length of `a string consisting of two Chinese characters` is 2. • Regular expressions are represented as strings. **Note** The strings used in the e_search function must be enclosed by double quotation marks ("") instead of single quotation marks (''). For example, `e_search("domain:'/url/test.jsp'")` is invalid whereas `e_search('domain:"/url/test.jsp"')` is valid.
Bytes	Example: `b'abc'`. Bytes are encoded in memory by using a format that is different from the format of strings. Bytes are received and returned by special functions.
None	Both `None` and `null` indicate a NULL value. Some named parameters of functions use `None` as the default value to indicate a specific default behavior. **Note** None or null does not indicate an empty string.
List	An array, for example, [1, 2, 3, 4]. • Some functions receive lists as parameters, for example, `e_dict_map("dict data", ["f1", "f2", "f3"], ...)`. • Some functions return lists as a result. For example, if you call the `json_select` function to extract an array, a list is returned.
Tuple	Tuples are similar to lists, for example, (1,2,3,4).

Type	Description
Dictionary	A collection of key-value pairs in the format of `{"key": "value", "k2": "v2", ...}`. A key is usually a unique string and a value can be of any data type. The key-value pairs are stored in a hash table in an unordered manner. • An event is a special dictionary. • Some functions can receive dictionaries in specific formats, for example, `{"key": [1,2, 3], "ke": {"k3": "va3"} }`. • The dictionary structure is also used as the input data for mapping fields to a dictionary.
Boolean	For example, `True`, `False`, `true`, and `false`.
Table	Each table consists of multiple rows and columns. You can load data in the comma-separated values (CSV) file that consists of multiple rows from an external resource to construct a table. You can also load multiple columns of data from an ApsaraDB RDS database or a Logstore to construct a table. Tables are used in advanced operations such as data mapping and enrichment.
Datetime object	A memory object that indicates date and time. A datetime object can be converted to a UNIX timestamp or a formatted time string. A datetime object can be passed to `dt_` functions for further conversion.

Event structure and fields

Events have the following structures and fields:

- Basic structure

  An event is processed in the dictionary structure during the data transformation process, for example, `{"__topic__": "access_log", "content": "....."}`.

  The keys and values of the dictionary correspond to the fields and values in the log.

  > **Note** The keys and values of an event are strings, and the keys must be unique.

- Meta-fields

  The following meta-fields are supported:

  - `__time__`: the timestamp when the log is received by Log Service. The value is an integer string. The timestamp follows the UNIX time format. It is the number of seconds that have elapsed since 00:00:00 Thursday, January 1, 1970.
  - `__topic__`: the topic of the log. Topics are used to group logs in a Logstore. You can specify a topic for logs that are written to Log Service. You can also specify a topic when you query logs.
  - `__source__`: the source of the log. For example, the value of this field can be the IP address of the server that is used to generate the log.

- Time field modification

You can change the value of the __time__ field to modify the event time of a log. You can use date and time functions to perform more operations on the __time__ field.

> **Note** If the __time__ field is deleted, the system time when the log is processed is used as the event time when the log is written to the destination Logstore.

- Tags

  Tags are used to differentiate fields in logs. Tags are in the format of `__tag__:name`.

  - If the source Logstore enables the feature of recording the public IP address, logs contain the `__tag__:__receive_time__` tag.
  - Container logs contain many container-related tags, such as `__tag__:__container_name__`.
  - You can add and modify tags. For example, you can add a tag named *type* by calling the `e_set("__tag__:type", "access_log")` function.

- Automatic type conversion during assignment

  The keys and values of an event are represented as strings. Therefore, when you assign a value to a field, the value is automatically converted to a string. For example:

```
e_set("v1", 12.3)
e_set("v2", True)
```

  After you call the preceding functions, the `v1` field is set to the string *12.3*, and the `v2` field is set to the string *true*.

  The following table provides conversion examples of different data types.

Type	Value	Conversion type	Conversion value
Integer	`1`	String	`"1"`
Float	`1.2`	String	`"1.2"`
Boolean	`True`	String	`"true"`
Byte	`b"123"`	String encoded in UTF-8	`"123"`
Tuple	◦ Example 1: `(1, 2, 3)` ◦ Example 2: `("a", 1)`	String that represents a list	◦ Example 1: `"[1, 2, 3]"` ◦ Example 2: `"[\"a\", 1]"`
List	◦ Example 1: `[1,2,3]` ◦ Example 2: `["a", 1]`	String	◦ Example 1: `"[1, 2, 3]"` ◦ Example 2: `"[\"a\", 1]"`

Type	Value	Conversion type	Conversion value
Dictionary	`{"1":2, "3":4}`	String	`"{\"1\": 2, \"3\": 4}"`
Datetime	`datetime(2018, 10, 10, 10, 10, 10)`	String that represents time in the ISO format	`2018-10-10 10:10:10`

Identifiers

The data transformation feature provides identifiers to represent field names or values. You can use the identifiers to simplify your code or make your code easy to understand.

Identifier	Type	Equivalent
true	Bool	`True`
false	Bool	`False`
null	None	`None`
F_TAGS	String	Regular expression that represents `tag` fields: `"__tag__:. +"`
F_META	String	Regular expression that represents `tag`, `__topic__`, and `__source__` fields: `__tag__:.+|__topic__|__source__`
F_TIME	String	`__time__`
F_PACK_META	String	Regular expression that represents `pack meta` fields: `"__pack_meta__|__tag__:__pack_id__"`
F_RECEIVE_TIME	String	`"__tag__:__receive_time__"`

JSON objects

JSON objects are returned by JSON expression functions, such as `json_select` and `json_parse`. The objects are in basic data structures. For more information, see Basic data structures. The following table describes how strings are converted to JSON objects.

String	JSON object	Type
`1`	`1`	Integer

String	JSON object	Type
`1.2`	`1.2`	Float
`true`	`True`	Boolean
`false`	`False`	Boolean
`"abc"`	`"abc"`	String
`null`	`None`	None
`["v1", "v2", "v3"]`	`["v1", "v2", "v3"]`	List
`["v1", 3, 4.0]`	`["v1", 3, 4.0]`	List
`{"v1": 100, "v2": "good"}`	`{"v1": 100, "v2": "good"}`	Dictionary
`{"v1": {"v11": 100, "v2": 200}, "v3": "good"}`	`{"v1": {"v11": 100, "v2": 200}, "v3": "good"}`	Dictionary

12.4. Basic syntax

This topic describes the basic syntax that is used in the domain-specific language (DSL) of Log Service.

Comments

Start the comment of a step with a number sign (`#`). Examples:

```
# Set the default topic. This is a comment placed at the beginning of a step.
e_set("__topic__", "access_log")   # Set the default topic. This is a comment placed at the end of a step.
```

Line wrapping

If the parameter list or string of a function does not fit on a single line, you can break the parameter list or the string. If the parameter list contains a comma (`,`), you can break the parameter list after the comma (`,`). If you want to break a string, use a backslash (`\`) to indicate that the string continues in the next line. Examples:

```
e_set("__topic__", "v1",
        "type", "v2",       # Break the parameter list with a comma (,).
        "length", 100)
e_set("__topic__", "this is a very long long long ..........." \
                           "......long text")
```

Function invoking

- Invoke the basic functions

  For example:

```
e_set("abc", "xyz")
```

> **Note** The number of parameter values that you pass to the function must be the same as the number of parameters defined for the function.

- Pass the basic variable parameters

  For example:

```
str_replace(value, old [,new [,count] ])
```

> **Note** The parameters in the square brackets are optional, for example, the `new` and `count` parameters in the preceding function. You cannot pass these parameters in the same way as the named parameters.
>
> ```
> # The following functions are invalid.
> str_replace("a-b-c", "-", new='%')
> str_replace("a-b-c", "-", new='%', count=1)
> ```
>
> You must pass these parameters in sequence.
>
> ```
> str_replace("a-b-c", "-", '%')
> str_replace("a-b-c", "-", '%', 2)
> ```

- Pass the named parameters

  A parameter with a default value is called a named parameter. For example, the `mode` parameter in the `e_set("abc", "xyz", mode="fill")` function is a named parameter.

  - You must pass the values for the named parameters in some functions under specific conditions. For more information, see the parameter description of each function.
  - You can pass the value of a named parameter by setting a parameter in the format of `mode=...`.
  - Multiple named parameters can be passed in random order. For example, `e_csv("data", ["f1", "f2", "f3"], sep='#', quote="|")` is equivalent to `e_csv("data", ["f1", "f2", "f3"], quote="|", sep='#')`.

> **Note** The named parameters always come after the non-named parameters.

- Invoke a combination of functions

  You can pass the returned value of a function as the value of a parameter to another function. In this case, you must make sure that the returned value is of the same data type as the value of the parameter. Examples:

```
e_set("abc", v("xyz"))
e_set("abc", str_lower(v("xyz")))
```

- Variable parameters

You can pass variable parameters to some functions. The `v("f1", ....)` function indicates that multiple parameters can be passed, for example, `v("f1", "f2", "f3")`.

If you need to pass both variable parameters and named parameters, you must place the named parameters after the variable parameters, for example, `v("f1", "f2", "f3", "f4", mode="fill")`.

Operator

You cannot use operators in DSL of the standard mode. Instead, you can use functions to perform the operations.

Operation	Function	Example
Addition (`+`)	op_add	`op_add(v("age"), 2)`
Subtraction (`-`)	op_sub	`op_sub(v("age"), 2)`
Multiplication (`*`)	op_mul	`op_mul(v("size"), 2)`
Exponentiation (`**`)	op_pow	`op_pow(v("size"), 2)`
Floor division (`//`)	op_div_floor	`op_div_floor(v("bytes"), 1024)`
Modulus (`%`)	op_mod	`op_mod(v("age"), 10)`
Negation (`-`)	op_neg	`op_neg(v("profit"))`
Existence check (`in`)	op_in	`op_in(["pass", "ok"], v("result"))`
Nonexistence check (`not in`)	op_not_in	`op_not_in(["pass", "ok"], v("result"))`
Logical operator AND (`and`)	op_and	`op_and(op_gt(v("age"), 18), op_lt(v("age"), 31))`
Logical operator OR (`or`)	op_or	`op_or(op_le(v("age"), 18), op_gt(v("age"), 65))`
Logical operator NOT (`not`)	op_not	`op_not(op_gt(v("age"), 18))`
Equal to (`==`)	op_eq	`op_eq(v("name"), "xiaoming")`
Not equal to (`!=`)	op_ne	`op_ne(v("name"), "xiaoming")`
Greater than (`>`)	op_gt	`op_gt(ct_int(v("age")), )`

Operation	Function	Example
Greater than or equal to (>=)	op_ge	op_ge(ct_int(v("age")), 18)
Less than (<)	op_lt	op_lt(ct_int(v("age")), 18)
Less than or equal to (<=)	op_le	op_le(ct_int(v("age")), 18)
String slicing ([...])	op_slice	op_slice(v("message"), 0, 20)

Assume that the value of the a field is *3600 * 6*. The following examples show an invalid function and a valid function to set the value for the field.

```
# *
e_set("a", 3600 * 6)    # Invalid
e_set("a", op_mul(3600, 6))    # Valid
# /
e_set("bytes_kb", v("bytes") / 1024)    # Invalid
e_set("bytes_kb", op_div_floor(v("bytes"), 1024))    # Valid
```

True or false evaluation

Some functions determine the event processing logic by checking whether a condition is true or false. A condition can be a fixed value or a value returned by an expression function.

You can perform a true or false evaluation of all data types in DSL orchestration. The following table lists the rules for true or false evaluation of various data types.

Data type	True	False
Boolean	True, true	False, false
None	-	Always false
Numeric	Not 0 or 0.0	0 or 0.0
String	Not empty	Empty string
Bytes	Not empty	Empty bytes
Tuple	Not empty	Empty tuple
List	Not empty	Empty list
Dictionary	Not empty	Empty dictionary
Table	One or more tables exist	No table exists

Data type	True	False
Datetime	One or more datetime objects exist	No datetime object exists

The following functions can be used to discard log entries based on the conditions:

```
e_if(True, DROP)      # Discard log entries if the value of the first parameter is true.
e_if(1, DROP)       # Discard log entries if the value of the first parameter is 1.
e_if(v("abc"), DROP)      # Discard log entries if the abc field exists and is not empty.
e_if(str_isdigit(v("abc")), DROP) # Discard log entries if the abc field exists and contains only digits.
```

12.5. Function overview

This topic describes all functions that are provided by Log Service to transform data.

Global processing functions

Category	Function	Description
Flow control functions	e_if	Performs an operation if a condition is met. You can specify multiple condition-operation pairs. • If a condition is met, the function performs the operation that corresponds to the condition. If the condition is not met, the function does not perform the operation, but evaluates the next condition. • If the function deletes an event by performing an operation, the function no longer performs other operations on the event.
	e_if_else	Performs an operation based on the evaluation result of a condition.
	e_switch	Performs an operation if a condition is met. You can specify multiple condition-operation pairs. • If a condition is met, the function performs the operation that corresponds to the condition and returns a result. If the condition is not met, the function does not perform the operation, but evaluates the next condition. • If no specified conditions are met and the default parameter is configured, the function performs the operation that is specified by the default parameter and returns a result. • If the function deletes an event by performing an operation, the function no longer performs other operations on the event.

Category	Function	Description
	e_compose	Combines multiple operations. • The function is commonly used in the `e_if` , `e_switch` , or `e_if_else` function. • The function performs specified operations on an event in sequence and returns a result. • If the function deletes an event by performing an operation, the function no longer performs other operations on the event.
Event processing functions	e_drop	Deletes an event if a condition is met.
	e_keep	Retains an event if a condition is met.
	e_split	Splits an event into multiple events based on the value of a specified field. You can also use JMESPath to extract the value of the field, and then split the event.
	e_output	Writes an event to a specified destination Logstore. You can configure parameters such as topic, source, and tags. The event is deleted after the event is written to the destination Logstore.
	e_coutput	Writes an event to a specified destination Logstore. You can configure parameters such as topic, source, and tags. The event is retained after the event is written to the destination Logstore. The retained event continues to be transformed.
	e_to_metric	Converts log data to time series data that can be stored in a Metricstore.
Field processing functions	v	Extracts the value of a field from an event. If multiple field names are passed to the function, the value of the first field that exists in the event is returned.
	e_set	Adds a field or specifies a new value for an existing field.
	e_drop_fields	Deletes the log fields that meet a specified condition.
	e_keep_fields	Retains the log fields that meet a specified condition.
	e_pack_fields	Packs specified log fields and assigns the log fields as a value to a new field.
	e_rename	Renames the log fields that meet a specified condition.
	e_regex	Extracts the value of a field from an event based on a regular expression and assigns the value to other fields.
	e_json	Performs operations on JSON objects in a specified field in an event. You can configure the parameters to expand JSON data, extract JSON data by using JMESPath, and expand the extracted JSON data.

Category	Function	Description
Value extraction functions	e_kv	Extracts key-value pairs from multiple source fields and encloses the values of the pairs by using the character that is specified by the quote parameter.
	e_kv_delimit	Extracts key-value pairs from source fields and separates the key-value pairs by using a specified delimiter.
	e_csv	Extracts multiple fields from a specified source field by using a specified delimiter and predefined field names. The default delimiter is a comma (,).
	e_tsv	Extracts multiple fields from a specified source field by using a specified delimiter and predefined field names. The default delimiter is \t.
	e_psv	Extracts multiple fields from a specified source field by using a specified delimiter and predefined field names. The default delimiter is a vertical bar(|).
	e_syslogrfc	Calculates the values of the facility and severity fields and returns level information, which is indicated by the facilitylabel field. The function calculates the values based on the value of the priority field and the specified syslog protocol.
	e_anchor	Extracts strings by using the defined anchor_rules.
Mapping and enrichment functions	e_dict_map	Maps the value of a source field to a value in a specified dictionary and returns a new field.
	e_table_map	Maps the value of a source field to a row in a specified table and returns a new field.
	e_search_dict_map	Maps data based on a specified dictionary and returns new fields. The dictionary consists of keys that are query strings and values for the keys.
	e_search_table_map	Maps data based on a specified table and returns new fields. The table consists of a column whose values are query strings and the rows for the column.
Value-added content function	e_threat_intelligence	Obtains threat intelligence from a log field and assigns the threat intelligence as a value to a new field.

Expression functions

Category	Function	Description
Event check	e_has	Checks whether a field exists.
	e_not_has	Checks whether a field does not exist.
	e_search	Searches for events in a simplified manner.

functions Category	Function	Description
	e_match, e_match_all, and e_match_any	Checks whether the value of a field in an event matches a specified expression.
Operator functions	op_if	Returns an expression based on a specified condition.
	op_ifnull	Returns the value of the first expression whose value is not None.
	op_coalesce	Returns the value of the first expression whose value is not None.
	op_nullif	Returns the value None if the value of Expression 1 is equal to the value of Expression 2. If the values of Expression 1 and Expression 2 are different, the value of Expression 1 is returned.
	op_and	Invokes the AND operation.
	op_not	Invokes the NOT operation.
	op_or	Invokes the OR operation.
	op_eq	Returns the result that is calculated based on the `a==b` condition. The data types of a and b must be the same. For example, a and b are both strings, numbers, or lists.
	op_ge	Returns the result that is calculated based on the `a>=b` condition. The data types of a and b must be the same. For example, a and b are both strings, numbers, or lists.
	op_gt	Returns the result that is calculated based on the `a>b` condition. The data types of a and b must be the same. For example, a and b are both strings, numbers, or lists.
	op_le	Returns the result that is calculated based on the `a<=b` condition. The data types of a and b must be the same. For example, a and b are both strings, numbers, or lists.
	op_lt	Returns the result that is calculated based on the `a<b` condition. The data types of a and b must be the same. For example, a and b are both strings, numbers, or lists.
	op_ne	Returns the result that is calculated based on the `a!=b` condition. The data types of a and b must be the same. For example, a and b are both strings, numbers, or lists.
	op_len	Calculates the number of characters in a text string. You can use this function in strings and expressions that return tuples, lists, or dictionaries.
	op_in	Checks whether a string, tuple, list, or dictionary contains a specified element.

Category	Function	Description
	op_not_in	Checks whether a string, tuple, list, or dictionary does not contain a specified element.
	op_slice	Extracts strings from a specified string, array, or tuple.
	op_index	Returns the element that corresponds to the index of a specified string, array, or tuple.
	op_add	Calculates the sum of multiple values. The values can be strings or numbers.
	op_max	Returns the largest value among the values of multiple fields or expressions.
	op_min	Returns the smallest value among the values of multiple fields or expressions.
Conversion functions	ct_int	Converts the value of a field or an expression to an integer.
	ct_float	Converts the value of a field or an expression to a floating-point number.
	ct_str	Converts the value of a field or an expression to a string.
	ct_bool	Converts the value of a field or an expression to a Boolean value.
	ct_chr	Converts the ANSI or Unicode value of a field or an expression to a character.
	ct_ord	Converts the value of a field or an expression to an ANSI or Unicode value.
	ct_hex	Converts the value of a field or an expression to a hexadecimal number.
	ct_oct	Converts the value of a field or an expression to an octal number.
	ct_bin	Converts the value of a field or an expression to a binary number.
	bin2oct	Converts a binary byte string to an octal string.
	bin2hex	Converts a binary byte string to a hexadecimal string.
	op_abs	Returns the absolute value of an input value.
	op_div_floor	Returns the integer part of the quotient of two input values.
	op_div_true	Returns the quotient of two input values.
	op_pow	Raises an input value to a specified power.
	op_mul	Returns the product of two input values.

Category	Function	Description
Arithmetic functions	op_neg	Returns the opposite number of an input value.
	op_mod	Returns the remainder of an input value divided by the other input value.
	op_sub	Returns the difference of two input values.
	op_round	Rounds an input value.
	op_sum	Returns the sum of input values.
	mat_ceil	Rounds an input value up to the nearest integer.
	mat_exp	Raises e to a specified power.
	mat_fabs	Returns the absolute value of an input value.
	mat_floor	Rounds an input value down to the nearest integer.
	mat_log	Returns the logarithm of an input value with the base specified by the other input value.
	mat_log10	Returns the base-10 logarithm of an input value.
	mat_sqrt	Returns the square root of an input value.
	mat_degrees	Converts radians to degrees.
	mat_radians	Converts degrees to radians.
	mat_sin	Returns the sine of an input value in radians.
	mat_cos	Returns the cosine of an input value in radians.
	mat_tan	Returns the tangent of an input value in radians.
	mat_acos	Returns the arc cosine of an input value in radians.
	mat_asin	Returns the arc sine of an input value in radians.
	mat_atan	Returns the arc tangent of an input value in radians.
	mat_atan2	Returns the arc tangent of X and Y coordinates.
	mat_atanh	Returns the inverse hyperbolic tangent of an input value.
	mat_hypot	Returns the Euclidean norm of two input values.
	str_format	Formats strings.
	str_join	Concatenates the elements in a sequence with a specified connector to generate a new string.

Category	Function	Description
String functions	str_zip	Concurrently splits two values or strings that are returned by expressions and combines the results to a string.
	str_encode	Encodes a string.
	str_decode	Decodes a string.
	str_hex_escape_encode	Escapes special characters. The function can escape hexadecimal characters to Chinese characters.
	str_sort	Sorts the characters in a string.
	str_reverse	Reverses the characters in a string.
	str_replace	Replaces an old string with a new string based on a specified rule.
	str_logtash_config_normalize	Converts the data in the Logstash configuration language to the JSON format.
	str_translate	Replaces specified characters in a string with mapping characters.
	str_strip	Deletes specified characters from a string.
	str_lstrip	Deletes specified characters from the start of a string.
	str_rstrip	Deletes specified characters from the end of a string.
	str_lower	Converts all uppercase letters in a string to lowercase letters.
	str_upper	Converts all lowercase letters in a string to uppercase letters.
	str_title	Capitalizes the first letter of each word in a string and converts the other letters in the string to lowercase letters.
	str_capitalize	Capitalizes the first letter of a string and converts the other letters in the string to lowercase letters.
	str_swapcase	Interchanges the uppercase letters and lowercase letters in a string.
	str_count	Counts the number of occurrences of a character in a string.
	str_find	Checks whether a string contains a specified substring.
	str_rfind	Returns the position of the last occurrence of a specified character in a string.
	str_endswith	Checks whether a string ends with a specified suffix.
	str_startswith	Checks whether a string starts with a specified substring.
	str_split	Splits a string by using a specified delimiter.

Category	Function	Description
	str_splitlines	Splits a string by using a line feed.
	str_partition	Splits a string into three parts from left to right by using a specified delimiter.
	str_rpartition	Splits a string into three parts from right to left by using a specified delimiter.
	str_center	Pads a string to a specified length by using a specified character.
	str_ljust	Right pads a string to a specified length by using a specified character.
	str_rjust	Left pads a string to a specified length by using a specified character.
	str_zfill	Left pads a string to a specified length by using 0.
	str_expandtabs	Converts `\t` in a string to spaces.
	str_isalnum	Checks whether a string contains only letters and digits.
	str_isalpha	Checks whether a string contains only letters.
	str_isascii	Checks whether a string is in the ASCII table.
	str_isdecimal	Checks whether a string contains only decimal characters.
	str_isdigit	Checks whether a string contains only digits.
	str_isidentifier	Checks whether a string is a valid Python identifier or checks whether a variable name is valid.
	str_islower	Checks whether a string contains lowercase letters.
	str_isnumeric	Checks whether a string contains digits.
	str_isprintable	Checks whether all characters in a string are printable characters.
	str_isspace	Checks whether a string contains only spaces.
	str_istitle	Checks whether the first letter of each word in a string is in uppercase and whether the other letters in the string are in lowercase.
	str_isupper	Checks whether all letters in a string are in uppercase.
	str_uuid	Generates a random UUID.
	dt_parse	Converts a value or the value of a time expression to a datetime object.

Category	Function	Description
Date and time functions	dt_str	Converts a value or the value of a time expression to a string.
	dt_parsetimestamp	Converts a value or the value of a time expression to a UNIX timestamp.
	dt_prop	Returns the specified attribute of a value or the value of a time expression.
	dt_now	Returns the current date and time.
	dt_today	Returns only the current date.
	dt_utcnow	Returns the current date and time in the current time zone.
	dt_fromtimestamp	Converts a UNIX timestamp to a datetime object.
	dt_utcfromtimestamp	Converts a UNIX timestamp to a datetime object in the current time zone.
	dt_strptime	Parses a time string into a datetime object.
	dt_currentstamp	Returns a UNIX timestamp.
	dt_totimestamp	Converts a datetime object to a UNIX timestamp.
	dt_strftime	Converts a datetime object to a string in a specified format.
	dt_strftimestamp	Converts a UNIX timestamp to a string in a specified format.
	dt_truncate	Extracts a time value from a value or the value of a time expression based on a specified time granularity.
	dt_add	Changes a value or the value of a time expression based on a specified time granularity.
	dt_MO	Offsets a specified time to the date of the previous or following Nth Monday. The offset value N is passed to the `weekday` parameter of the `dt_add` function, where N can be a positive or negative integer. If you want to pass a negative integer, use `op_neg(Positive integer)`.
	dt_TU	Offsets a specified time to the date of the previous or following Nth Tuesday. The offset value N is passed to the `weekday` parameter of the `dt_add` function, where N can be a positive or negative integer. If you want to pass a negative integer, use `op_neg(Positive integer)`.

Category	Function	Description
	dt_WE	Offsets a specified time to the date of the previous or following Nth Wednesday. The offset value N is passed to the `weekday` parameter of the `dt_add` function, where N can be a positive or negative integer. If you want to pass a negative integer, use `op_neg(Positive integer)`.
	dt_TH	Offsets a specified time to the date of the previous or following Nth Thursday. The offset value N is passed to the `weekday` parameter of the `dt_add` function, where N can be a positive or negative integer. If you want to pass a negative integer, use `op_neg(Positive integer)`.
	dt_FR	Offsets a specified time to the date of the previous or following Nth Friday. The offset value N is passed to the `weekday` parameter of the `dt_add` function, where N can be a positive or negative integer. If you want to pass a negative integer, use `op_neg(Positive integer)`.
	dt_SA	Offsets a specified time to the date of the previous or following Nth Saturday. The offset value N is passed to the `weekday` parameter of the `dt_add` function, where N can be a positive or negative integer. If you want to pass a negative integer, use `op_neg(Positive integer)`.
	dt_SU	Offsets a specified time to the date of the previous or following Nth Sunday. The offset value N is passed to the `weekday` parameter of the `dt_add` function, where N can be a positive or negative integer. If you want to pass a negative integer, use `op_neg(Positive integer)`.
	dt_astimezone	Converts a value or the value of a time expression to a datetime object in a specified time zone.
	dt_diff	Returns the difference between two values or the values of two time expressions based on a specified time granularity.
Regular expression functions	regex_select	Extracts a specified value based on a regular expression.
	regex_findall	Extracts all values that match a regular expression.
	regex_match	Checks whether a value matches a regular expression.
	regex_replace	Replaces specified characters in a string based on a regular expression.
	regex_split	Splits a string into an array of strings.
Grok function	grok	Extracts a specified value based on a regular expression.
	json_select	Extracts or calculates specified values from a JSON expression by using JMESPath.

Category	Function	Description
Structured data functions	json_parse	Parses a specified value into a JSON object.
	xml_to_json	Converts XML data to JSON data and expands the data.
	gzip_compress	Compresses data, encodes the compressed data by using the Base64 algorithm, and then returns the result.
	gzip_decompress	Decodes data by using the Base64 algorithm, decompresses the decoded data, and then returns the result.
IP address parsing functions	geo_parse	Parses an IP address into the information about the city, province, and country of the IP address.
	ip_cidrmatch	Checks whether an IP address belongs to a CIDR block.
	ip_version	Checks whether the version of an IP address is IPv4 or IPv6.
	ip_type	Identifies the type of an IP address and checks whether the type of the IP address is private or public.
	ip_makenet	Converts an IP address to a CIDR block.
	ip_to_format	Converts a CIDR block to a format that specifies the netmask or prefix length of the CIDR block.
	ip_overlaps	Checks whether two CIDR blocks overlap.
Encoding and decoding functions	url_encoding	Encodes URL data.
	url_decoding	Decodes URL data.
	base64_encoding	Encodes data by using the Base64 algorithm.
	base64_decoding	Decodes data by using the Base64 algorithm.
	html_encoding	Encodes data in the HTML format.
	html_decoding	Decodes data that is encoded in the HTML format.
	md5_encoding	Encodes data by using the MD5 algorithm.
	sha1_encoding	Encodes data by using the SHA1 algorithm.
	ip2long	Converts an IP address to a value of the long type.
	long2ip	Converts a value of the long type to an IP address.
	aes_encrypt	Encrypts data by using the AES algorithm.
	aes_decrypt	Decrypts data by using the AES algorithm.

Category	Function	Description
Parsing functions	ua_parse_device	Parses User-Agent and returns the device information.
	ua_parse_os	Parses User-Agent and returns the operating system information.
	ua_parse_agent	Parses User-Agent and returns the browser information.
	ua_parse_all	Parses User-Agent and returns all information.
List functions	lst_make	Creates a list.
	lst_insert	Inserts elements to a specified position in a list.
	lst_append	Adds elements to the end of a list.
	lst_delete_at	Deletes the element at a specified position from a list.
	lst_reverse	Reverses the values in a list.
	lst_get	Returns an element of a list or a tuple.
Dictionary functions	dct_make	Constructs a dictionary.
	dct_update	Updates a dictionary.
	dct_delete	Deletes key-value pairs from a dictionary.
	dct_keys	Returns the keys of a dictionary.
	dct_values	Returns the values of a dictionary.
	dct_get	Returns the value that corresponds to a specified key in a dictionary.
Table functions	tab_parse_csv	Constructs a table from CSV text.
	tab_to_dict	Constructs a dictionary from a table.
Resource functions	res_local	Returns the value of an advanced parameter that is specified in a data transformation task.
	res_rds_mysql	Returns the data of a specified database table in ApsaraDB RDS for MySQL. The data can be refreshed at regular intervals.
	res_log_logstore_pull	Pulls data from another Logstore when data in a Logstore is being transformed. You can pull data and maintain table data in a continuous manner.
	res_oss_file	Returns the content of an object in a specified bucket of Object Storage Service (OSS). The content can be refreshed at regular intervals.

12.6. Expression functions

12.6.1. Function overview

The domain-specific language (DSL) for Log Service is provided to construct expression functions that return specific values. This helps you transform log data.

The following table describes the expression functions.

Category	Function	Description
Event check functions	e_has, e_not_has, e_search, e_match, e_match_any, and e_match_all	Checks whether a field exists or whether a field or the value of a field meets a specified condition.
Operator functions	Some op_* functions	Compares values, evaluates values based on a specified condition or container, or performs general-purpose multi-value operations.
Conversion functions	ct_* functions	Converts data types among numeric values, strings, and Boolean values, or converts numbers between different numeral systems.
Arithmetic functions	math_*, mat_*, and some op_* functions	Performs mathematical calculation or multi-value calculation.
String functions	str_* functions	Processes strings.
Date and time functions	dt_* functions	Converts time values among UNIX timestamps, datetime objects, and datetime strings, changes time zones, and returns the difference between two time values.
Regular expression functions	regex_* functions	Extracts, retrieves, replaces, or splits values based on regular expressions.
Grok function	Grok function	Extracts specific values based on regular expressions.
Structured data functions	json_*, xml_*, and gzip_* functions	Extracts or parses fields.
IP address parsing functions	geo_parse and ip_* functions	Parses IP addresses.
Encoding and decoding functions	url_*, html_*, md5_*, sha1_*, base64_*, ip2long, long2ip, aes_encrypt, and aes_decrypt functions	Encodes or decodes data.
Parsing functions	ua_* functions	Parses a User-Agent header.

Category	Function	Description
List functions	Some op_* and lst_* functions	Performs operations on a list, including obtaining or modifying a list.
Dictionary functions	Some op_* and dct_* functions	Performs operations on a dictionary, including obtaining or modifying a dictionary.
Table functions	tab_* functions	Constructs a table from text or constructs a dictionary from a table.
Resource functions	res_* functions	Pulls configuration data and data from an Object Storage Service (OSS) bucket, a Logstore, or a table in an ApsaraDB RDS for MySQL database.

12.6.2. Event check functions

This topic describes the syntax and parameters of event check functions. This topic also provides examples on how to use the functions.

Functions

Type	Function	Description
Basic functions	e_has	Checks whether a field exists.
	e_not_has	Checks whether a field does not exist.
Expression functions	e_search	Searches for an event by using Lucene-like query syntax.
	e_match*	The following functions are used to match data: `e_match`, `e_match_all`, and `e_match_any`. Checks whether the value of a field in an event meets the conditions specified in an expression.

The following table describes the expression functions that you can use together with event check functions.

Type	Function	Description
Logical functions	op_and	Invokes the AND operation.
	op_or	Invokes the OR operation.
	op_not	Invokes the NOT operation.
	op_nullif	Checks whether the values of two expressions are equal.

Type	Function	Description
	op_ifnull	Returns the value of the first expression whose value is not None.
	op_coalesce	Returns the value of the first expression whose value is not None.

e_has

- Syntax

```
e_has("field name")
```

- Parameters

Parameter	Type	Required	Description
field name	String	Yes	The name of a field in an event.

- Response

  If the specified field exists, True is returned. Otherwise, False is returned.

- Example

  Check whether an event contains the content field. If the event contains the content field, the event is retained. Otherwise, the event is dropped.

  - Raw log entry:

    ```
    content: 123
    ```

  - Transformation rule:

    ```
    e_keep(e_has("content"))
    ```

  - Result:

    ```
    content: 123
    ```

e_not_has

- Syntax

```
e_not_has("key")
```

- Parameters

Parameter	Type	Required	Description
key	String	Yes	The name of a field in an event.

- Response

  If the specified field does not exist, True is returned. Otherwise, False is returned.

- Example

Check whether an event contains the content field. If the event does not contain the content field, the event is retained. Otherwise, the event is dropped.

- Raw log entry:

```
content: 123
```

- Transformation rule:

```
e_if_else(e_not_has("content"),KEEP,DROP)
```

- Result:

  The event is dropped.

e_search

- Syntax

```
e_search(Query string)
```

- Parameters

Parameter	Type	Required	Description
Query string	String	Yes	The string that you want to use to filter log data and simplify data transformation. For more information, see Query string syntax.

- Response

  If the specified conditions are met, True is returned. Otherwise, False is returned.

- Example

```
# Full-text search
e_search("active error")   # Search for multiple substrings in full text. The substrings are associated with each other by using the logical operator OR.
e_search('"active error"')   # Search for a substring in full text.
# Field search
e_search("status: active")   # Search for a substring in a specified field.
e_search('author: "john smith"') # Search for a substring that contains a space character in a specified field.
e_search('field: active error')   # Search the specified field for the substring "active" or searches all logs for the substring "error". The query string in this example is equivalent to field:active OR "error".
# Exact match
e_search('author== "john smith"')
# Search for field values by using wildcard characters. You can use an asterisk (*) to match zero or more characters. You can use a question mark (?) to match one character.
e_search("status: active * test")   # active*test contains one asterisk (*). You do not need to enclose the value in double quotation marks ("").
e_search("status: active?good")    # active?good contains one question mark (?). You do not need to enclose the value in double quotation marks ("").
e_search("status== ac*tive?good")   # The query string is used for exact match.
# Escape special characters in a field value. Asterisks (*) or question marks (?) that are not used as wildcards must be escaped in a field value by using backslashes (\).
e_search('status: "\*\?()[]:="')  # \*\?()[]:= contains multiple special characters. You must enclose the value in double quotation marks (""). The asterisks (*), question marks (?), and backslashes (\) in the value are escaped.
e_search("status: active\* test")   # active\*test contains one asterisk (*). You do not need to enclose the value in double quotation marks ("").
e_search("status: active\?test")  # active\?test contains one question mark (?). You do not need to enclose the value in double quotation marks ("").
# Escape special characters in a field name
e_search("\*\(1+1\)\?: abc")   # You cannot enclose the field name in double quotation marks (""). You must escape special characters by using backslashes (\).
e_search("__tag__\:__container_name__: abc")   # You must escape special characters by using backslashes (\).
e_search("field name in Chinese: abc")                      # Enter the Chinese characters that comprise the field name.
# Search for strings by using regular expressions.
e_search('content~="regular expression"')   # Search for substrings that match the regular expression.
# Numeric value comparison
e_search('count: [100, 200]')   # >=100 and <=200
e_search('count: [*, 200]')     # <=200
e_search('count: [200, *]')     # >=200
e_search('age >= 18')           # >= 18
e_search('age > 18')            # > 18
# Relational operators
e_search("abc OR xyz")   # The relational operator is case-insensitive.
e_search("abc and (xyz or zzz)")
e_search("abc and not (xyz and not zzz)")
e_search("abc && xyz")    # and
e_search("abc || xyz")    # or
e_search("abc || !xyz")   # or not
```

e_match*

- Syntax

```
e_match(field name, regular expression, full=True)
e_match_all(field name 1, regular expression 1, field name 2, regular expression 2, ..., full=True)
e_match_any(field name 1, regular expression 1, field name 2, regular expression 2, ..., full=True)
```

> **Note**
> - The `field name` and `regular expression` parameters must be specified in pairs.
> - In most cases, the e_match function is used together with the `op_not`, `op_and`, or `op_or` function.

- Parameters

Parameter	Type	Required	Description
field name	String	Yes	The name of a field. If the specified field does not exist, the condition that is specified for the field is not met. For example, if the `f1` field does not exist, the `e_match("f1", ...)` function returns *False*.
regular expression	String	Yes	The regular expression that you want to use to match strings. If you want to match strings by using exact strings, you can use the `str_regex_escape` function to convert regular expressions.
full	Bool	No	Specifies whether to perform an exact match. By default, the parameter is set to *True*, which specifies an exact match. For more information, see Regular expressions.

- Response

  If the specified field matches the regular expression, True is returned. Otherwise, False is returned.

> **Note**
> - `e_match_any`: If one or more specified fields match the regular expression, True is returned. Otherwise, False is returned.
> - `e_match_all`: If all specified fields match the regular expression, True is returned. Otherwise, False is returned.

- Examples

- Example 1

  Perform an exact match by using the e_match function.

  - Raw log entry:

    ```
    k1: 123
    ```

  - Transformation rule:

    ```
    e_set("match",e_match("k1",r'\d+'))
    ```

  - Result:

    ```
    k1: 123
    match: True
    ```

- Example 2

  Perform an exact match by using the e_match_all function.

  - Raw log entry:

    ```
    k1: 123
    k2: abc
    k3: abc123
    ```

  - Transformation rule:

    ```
    e_set("match",e_match_all('k1', r'\d+', 'k4', '.+'))
    ```

  - Result:

    ```
    k1: 123
    k2: abc
    k3: abc123
    match: False
    ```

  > **Note** You can use the e_match_any and e_match_all functions in a similar manner.

12.6.3. Operator functions

This topic describes the syntax and parameters of operator functions. This topic also provides examples on how to use the functions.

> **Note** If you want to obtain a negative value, use the op_neg (positive value) function. For example, use op_neg(1) to return `-1`.

Functions

Type	Function	Description
	op_if	Returns an expression based on a specified condition.

Type	Function	Description
Conditional functions and logical functions		
	op_ifnull and op_coalesce	Returns the value of the first expression whose value is not None.
	op_nullif	Returns the value None if the value of Expression 1 is equal to the value of Expression 2. Otherwise, the value of Expression 1 is returned.
	op_and	Returns a Boolean value after the AND operator computes the logical AND of two or more specified fields. Each field can be of an arbitrary data type.
	op_not	Returns a Boolean value after the NOT operator computes logical negation of a specified field. The field can be of an arbitrary data type.
	op_or	Returns a Boolean value after the OR operator computes the logical OR of two or more specified fields. Each field can be of an arbitrary data type.
Comparison functions	op_eq	Returns the result that is calculated based on the `a==b` condition. The data types of a and b must be the same. For example, a and b are both strings, numbers, or lists.
	op_ge	Returns the result that is calculated based on the `a>=b` condition. The data types of a and b must be the same. For example, a and b are both strings, numbers, or lists.
	op_gt	Returns the result that is calculated based on the `a>b` condition. The data types of a and b must be the same. For example, a and b are both strings, numbers, or lists.
	op_le	Returns the result that is calculated based on the `a<=b` condition. The data types of a and b must be the same. For example, a and b are both strings, numbers, or lists.
	op_lt	Returns the result that is calculated based on the `a<b` condition. The data types of a and b must be the same. For example, a and b are both strings, numbers, or lists.

Type	Function	Description
	op_ne	Returns the result that is calculated based on the `a!=b` condition. The data types of a and b must be the same. For example, a and b are both strings, numbers, or lists.
Container functions	op_len	Calculates the number of characters in a text string. You can use this function in expressions that return strings, tuples, lists, or dictionaries.
	op_in	Checks whether a string, tuple, list, or dictionary contains a specified element.
	op_not_in	Checks whether a string, tuple, list, or dictionary does not contain a specified element.
	op_slice	Truncates a specified string, array, or tuple.
	op_index	Returns the element that corresponds to the index of a specified string, array, or tuple.
General-purpose multivalued functions	op_add	Calculates the sum of multiple values. The values can be strings or numbers.
	op_max	Returns the maximum value among the values of multiple fields or expressions.
	op_min	Returns the minimum value among the values of multiple fields or expressions.

op_if

- Syntax

```
op_if(Condition,Value 1,Value 2)
```

- Parameters

Parameter	Type	Required	Description
Condition	Arbitrary	Yes	A condition. If the value of the condition is not a Boolean value, the system evaluates whether the condition is true or false. For more information, see True or false evaluation.

Parameter	Type	Required	Description
Value 1	Arbitrary	Yes	Expression 1 that is returned if the specified condition is evaluated to *True*.
Value 2	Arbitrary	Yes	Expression 2 that is returned if the specified condition is evaluated to *False*.

- Response

  The expression that corresponds to the evaluation result of the specified condition is returned.

- Examples
  - Example 1: If the value of the `content` field is *True*, the value of Expression 1 is aspecified for the `test_if` field.

    Raw log entry:

    ```
    content: hello
    ```

    Transformation rule:

    ```
    e_set("test_if", op_if(v("content"),"still origion content","replace this"))
    ```

    Result:

    ```
    content: hello
    test_if:  still origion content
    ```

  - Example 2: If the value of the `content` field is false, the value of Expression 2 is assigned to the `test_if` field.

    Raw log entry:

    ```
    content: 0
    ```

    Transformation rule:

    ```
    e_set("test_if", op_if(ct_int(v("content", default=0)),"still origion content","replace this"))
    ```

    Result:

    ```
    content: 0
    test_if:  replace this
    ```

op_ifnull

- Syntax

  ```
  op_ifnull(Value 1,Value 2)
  ```

- Parameters

Parameter	Type	Required	Description
Value 1	Arbitrary	Yes	The value of the expression that you want to return.
Value 2	Arbitrary	Yes	The value of the expression that you want to return.

- Response

  The value of the first expression whose value is not None is returned.

- Examples
  - Example 1

    Raw log entry:

    ```
    test_if: hello
    escape_name: Etl
    ```

    Transformation rule:

    ```
    e_set("test_ifnull", op_ifnull(v("escape_name"),v("test_if")))
    ```

    Result:

    ```
    test_if: hello
    escape_name: Etl
    test_ifnull:  Etl
    ```

  - Example 2

    Raw log entry:

    ```
    test_if: hello
    escape_name: Etl
    ```

    Transformation rule:

    ```
    e_set("test_ifnull", op_ifnull(v("test_if"),v("escape_name")))
    ```

    Result:

    ```
    test_if: hello
    escape_name: Etl
    test_ifnull:  hello
    ```

op_coalesce

The parameters and examples of the op_coalesce function are similar to the parameters and examples of the `op_ifnull` function. For more information, see op_ifnull.

op_nullif

- Syntax

```
op_nullif(Value 1,Value 2)
```

- Parameters

Parameter	Type	Required	Description
Value 1	Arbitrary	Yes	The value of a field.
Value 2	Arbitrary	Yes	The value of a field.

- Response

  If Value 1 is equal to Value 2, the value None is returned. Otherwise, Value 1 is returned.

- Examples
  - Example 1

    Raw log entry:

    ```
    content: hello
    escape_name: Etl
    ```

    Transformation rule:

    ```
    e_set("test_ifnull", op_nullif(v("content"),v("escape_name")))
    ```

    Result:

    ```
    content: hello
    escape_name: Etl
    test_nullif:  hello
    ```

  - Example 2

    Raw log entry:

    ```
    content: hello
    escape_name: hello
    ```

    Transformation rule:

    ```
    e_set("test_ifnull", op_nullif(v("content"),v("escape_name")))
    ```

    Result:

    ```
    content: hello
    escape_name: hello
    # In this example, the value of the content field is the same as the value of the escape_name field. Therefore, the value None is returned. This indicates that no content is assigned to the test_isnull field.
    ```

op_and

- Syntax

```
op_and(Value 1,Value 2, ...)
```

- Parameters

Parameter	Type	Required	Description
Value 1	Arbitrary	Yes	The field value on which you want to perform the AND operation.
Value 2	Arbitrary	Yes	The field value on which you want to perform the AND operation.

- Response
  - If all specified fields are evaluated to true, the value True is returned.
  - The system evaluates all types of fields to true or false. For more information, see True or false evaluation.
- Examples
  - Example 1

    Raw log entry:

```
number1: 123
number2: 234
```

    Transformation rule:

```
e_set("op_and", op_and(v("number1"),v("number2")))
```

    Result:

```
number1: 123
number2: 234
op_and:  True
```

- Example 2

  Raw log entry:

  ```
  number1: 0
  number2: 234
  ```

  Transformation rule:

  ```
  e_set("op_and", op_and(v("number1"),v("number2")))
  ```

  Result:

  ```
  number1: 0
  number2: 234
  op_and: True
  ```

- Example 3

  Raw log entry:

  ```
  ctx1: False
  ctx2: 234
  ```

  Transformation rule:

  ```
  e_set("op_and", op_and(v("ctx1"),v("ctx2")))
  ```

  Result:

  ```
  ctx1: False
  ctx2: 234
  op_and: False
  ```

- Example 4

  Raw log entry:

  ```
  ctx1: True
  ctx2: 234
  ```

  Transformation rule:

  ```
  e_set("op_and", op_and(v("ctx1"),v("ctx2")))
  ```

  Result:

  ```
  ctx1: True
  ctx2: 234
  op_and: True
  ```

op_not

- Syntax

  ```
  op_not(Value)
  ```

- Parameters

Parameter	Type	Required	Description
Value	Arbitrary	Yes	The field value on which you want to perform the NOT operation.

- Response
  - The value True or False is returned.
  - The system evaluates all types of fields to true or false. For more information, see True or false evaluation.
- Examples
  - Example 1

    Raw log entry:

    ```
    ctx1: True
    ```

    Transformation rule:

    ```
    e_set("op_not", op_not(v("ctx1")))
    ```

    Result:

    ```
    ctx1: True
    op_not:  False
    ```

  - Example 2

    Raw log entry:

    ```
    ctx1: 345
    ```

    Transformation rule:

    ```
    e_set("op_not", op_not(v("ctx1")))
    ```

    Result:

    ```
    ctx1: 345
    op_not:  False
    ```

- Example 3

  Raw log entry:

  ```
  ctx1: 0
  ```

  Transformation rule:

  ```
  e_set("op_not", op_not(ct_int(v("ctx1"))))
  ```

  Result:

  ```
  ctx1: 0
  op_not:  True
  ```

- Example 4

  Raw log entry:

  ```
  ctx1: ETL
  ```

  Transformation rule:

  ```
  e_set("op_not", op_not(v("ctx1")))
  ```

  Result:

  ```
  ctx1: ETL
  op_not:  False
  ```

- Example 5

  Raw log entry:

  ```
  ctx1: None
  ```

  Transformation rule:

  ```
  e_set("op_not", op_not(v("ctx1")))
  ```

  Result:

  ```
  ctx1: None
  op_not:  True
  ```

op_or

- Syntax

  ```
  op_or(Value 1, Value 2, ...Variable...)
  ```

- Parameters

Parameter	Type	Required	Description

Parameter	Type	Required	Description
Value 1	Arbitrary	Yes	The field value on which you want to perform the OR operation.
Value 2	Arbitrary	Yes	The field value on which you want to perform the OR operation.

- Response
  - If one or more of the specified fields are evaluated to true, the value True is returned. Otherwise, the value False is returned.
  - The system evaluates all types of fields to true or false. For more information, see True or false evaluation.
- Examples
  - Example 1

    Raw log entry:

    ```
    ctx1: 123
    ctx2: 234
    ```

    Transformation rule:

    ```
    e_set("op_or", op_or(v("ctx1"),v("ctx2")))
    ```

    Result:

    ```
    ctx1: 123
    ctx2: 234
    op_or:  True
    ```

  - Example 2

    Raw log entry:

    ```
    ctx1: 0
    ctx2: 234
    ```

    Transformation rule:

    ```
    e_set("op_or", op_or(v("ctx1"),v("ctx2")))
    ```

    Result:

    ```
    ctx1: 0
    ctx2: 234
    op_or:  True
    ```

- Example 3

  Raw log entry:

  ```
  ctx1: ETL
  ctx2: ALIYUN
  ```

  Transformation rule:

  ```
  e_set("op_or", op_or(v("ctx1"),v("ctx2")))
  ```

  Result:

  ```
  ctx1: ETL
  ctx2: ALIYUN
  op_or:  True
  ```

- Example 4

  Raw log entry:

  ```
  ctx1: True
  ctx2: False
  ```

  Transformation rule:

  ```
  e_set("op_or", op_or(v("ctx1"),v("ctx2")))
  ```

  Result:

  ```
  ctx1: True
  ctx2: False
  op_or:  True
  ```

- Example 5

  Raw log entry:

  ```
  ctx1: 0
  ctx2: False
  ```

  Transformation rule:

  ```
  e_set("op_or", op_or(ct_int(v("ctx1")),v("ctx2")))
  ```

  Result:

  ```
  ctx1: 0
  ctx2: False
  op_or:  False
  ```

- Example 6

  Raw log entry:

  ```
  ctx1: 124
  ctx2: True
  ```

  Transformation rule:

  ```
  e_set("op_or", op_or(v("ctx1"),v("ctx2")))
  ```

  Result:

  ```
  ctx1: 124
  ctx2: True
  op_or:  True
  ```

op_eq

- Syntax

  ```
  op_eq(Value 1,Value 2)
  ```

- Parameters

Parameter	Type	Required	Description
Value 1	Arbitrary	Yes	The value of a field.
Value 2	Same as the data type of Value 1	Yes	The value of a field.

- Response

  If Value 1 is equal to Value 2, the value True is returned. Otherwise, the value False is returned.

- Examples
  - Example 1

    Raw log entry:

    ```
    content: hello
    ctx: hello
    ```

    Transformation rule:

    ```
    e_set("test_eq", op_eq(v("content"),v("ctx")))
    ```

    Result:

    ```
    content: hello
    ctx: hello
    test_eq: True
    ```

- Example 2

  Raw log entry:

  ```
  content: hello
  ctx: ctx
  ```

  Transformation rule:

  ```
  e_set("test_eq", op_eq(v("content"),v("ctx")))
  ```

  Result:

  ```
  content: hello
  ctx: ctx
  test_eq: False
  ```

op_ge

- Syntax

  ```
  op_ge(Value 1, Value 2)
  ```

- Parameters

Parameter	Type	Required	Description
Value 1	Arbitrary	Yes	The value of a field.
Value 2	Same as the data type of Value 1	Yes	The value of a field.

- Response

  If Value 1 is greater than or equal to Value 2, the value True is returned. Otherwise, the value False is returned.

- Examples
  - Example 1: If the value of the apple_price field is greater than or equal to the value of the orange_price field, the value True is returned.

    Raw log entry:

    ```
    apple_price: 16
    orange_price: 14
    ```

    Transformation rule:

    ```
    e_set("test_ge", op_ge(ct_int(v("apple_price")),ct_int(v("orange_price"))))
    ```

    Result:

    ```
    apple_price: 16
    orange_price: 14
    test_ge: true
    ```

- Example 2: If the value of the apple_price field is less than the value of the orange_price field, the value False is returned.

  Raw log entry:

  ```
  apple_price: 12
  orange_price: 14
  ```

  Transformation rule:

  ```
  e_set("test_ge", op_ge(ct_int(v("apple_price")),ct_int(v("orange_price"))))
  ```

  Result:

  ```
  apple_price: 12
  orange_price: 14
  test_ge: false
  ```

op_gt

- Syntax

  ```
  op_gt(Value 1, Value 2)
  ```

- Parameters

Parameter	Type	Required	Description
Value 1	Arbitrary	Yes	The value of a field.
Value 2	Same as the data type of Value 1	Yes	The value of a field.

- Response

  If Value 1 is greater than Value 2, the value True is returned. Otherwise, the value False is returned.

- Examples
  - Example 1: If the value of the old_number field is greater than the value of the young_number field, the value True is returned. Otherwise, the value False is returned.

    Raw log entry:

    ```
    old_number: 16
    young_number: 14
    ```

    Transformation rule:

    ```
    e_set("op_gt",op_gt(ct_int(v("old_number")),ct_int(v("young_number"))))
    ```

    Result:

    ```
    old_number: 16
    young_number: 14
    test_ge: True
    ```

- Example 2: If the value of the priority field is greater than the value of the price field, the value True is returned. Otherwise, the value False is returned.

  Raw log entry:

  ```
  priority: 14
  price: 16
  ```

  Transformation rule:

  ```
  e_set("op_gt",op_gt(ct_int(v("priority")),ct_int(v("price"))))
  ```

  Result:

  ```
  priority: 14
  price: 16
  test_ge: False
  ```

op_le

- Syntax

  ```
  op_le(Value 1, Value 2)
  ```

- Parameters

Parameter	Type	Required	Description
Value 1	Arbitrary	Yes	The value of a field.
Value 2	Same as the data type of Value 1	Yes	The value of a field.

- Response

  If Value 1 is less than or equal to Value 2, the value True is returned. Otherwise, the value False is returned.

- Examples
  - Example 1: If the value of the priority field is less than or equal to the value of the price field, the value True is returned. Otherwise, the value False is returned.

    Raw log entry:

    ```
    priority: 16
    price: 14
    ```

    Transformation rule:

    ```
    e_set("op_le",op_le(ct_int(v("priority")),ct_int(v("price"))))
    ```

    Result:

    ```
    priority: 16
    price: 14
    test_ge: False
    ```

- Example 2: If the value of the priority field is less than or equal to the value of the price field, the value True is returned. Otherwise, the value False is returned.

  Raw log entry:

  ```
  priority: 14
  price: 16
  ```

  Transformation rule:

  ```
  e_set("op_le",op_le(ct_int(v("priority")),ct_int(v("price"))))
  ```

  Result:

  ```
  priority: 14
  price: 16
  test_ge: True
  ```

op_lt

- Syntax

  ```
  op_lt(Value 1, Value 2)
  ```

- Parameters

Parameter	Type	Required	Description
Value 1	Arbitrary	Yes	The value of a field.
Value 2	Same as the data type of Value 1	Yes	The value of a field.

- Response

  If Value 1 is less than Value 2, the value True is returned. Otherwise, the value False is returned.

- Examples
  - Example 1: If the value of the priority field is less than the value of the price field, the value True is returned. Otherwise, the value False is returned.

    Raw log entry:

    ```
    priority: 16
    price: 14
    ```

    Transformation rule:

    ```
    e_set("op_lt",op_lt(ct_int(v("priority")),ct_int(v("price"))))
    ```

    Result:

    ```
    priority: 16
    price: 14
    op_lt: False
    ```

- Example 2: If the value of the priority field is less than the value of the price field, the value True is returned. Otherwise, the value False is returned.

  Raw log entry:

  ```
  priority: 14
  price: 15
  ```

  Transformation rule:

  ```
  e_set("op_lt",op_lt(ct_int(v("priority")),ct_int(v("price"))))
  ```

  Result:

  ```
  priority: 14
  price: 15
  op_lt: True
  ```

op_ne

- Syntax

  ```
  op_ne(Value 1, Value 2)
  ```

- Parameters

Parameter	Type	Required	Description
Value 1	Arbitrary	Yes	The value of a field.
Value 2	Same as the data type of Value 1	Yes	The value of a field.

- Response

  If Value 1 is not equal to Value 2, the value True is returned. Otherwise, the value False is returned.

- Examples
  - Example 1

    Raw log entry:

    ```
    priority: 16
    price: 14
    ```

    Transformation rule:

    ```
    e_set("op_ne",op_ne(ct_int(v("priority")),ct_int(v("price"))))
    ```

    Result:

    ```
    priority: 16
    price: 14
    op_ne: True
    ```

- Example 2

  Raw log entry:

  ```
  priority: 14
  price: 14
  ```

  Transformation rule:

  ```
  e_set("op_ne",op_ne(ct_int(v("priority")),ct_int(v("price"))))
  ```

  Result:

  ```
  priority: 14
  price: 14
  op_ne: False
  ```

op_len

- Syntax

  ```
  op_len(Value)
  ```

- Parameters

Parameter	Type	Required	Description
Value	String, tuple, list, or dictionary	Yes	The field value whose length you want to calculate.

- Response

  The length of the specified field value is returned.

- Example

  Raw log entry:

  ```
  content: I,love,this,world
  ```

  Transformation rule:

  ```
  e_set("op_len",op_len(v("content")))
  ```

  Result:

  ```
  content: I,love,this,world
  op_len: 17
  ```

op_in

- Syntax

  ```
  op_in(a, b)
  ```

- Parameters

Parameter	Type	Required	Description
a	String, tuple, list, or dictionary	Yes	The name of a container.
b	Arbitrary	Yes	The name of an element.

> **Note** In this function, the a parameter that specifies a container precedes the b parameter that specifies an element.

- Response

  If the container a contains the element b, the value True is returned. Otherwise, the value False is returned.

- Example

  Raw log entry:

```
list:  [1, 3, 2, 7, 4, 6]
num2:  2
```

  Transformation rule:

```
e_set("op_in",op_in(v("list"),v("num2")))
```

  Result:

```
list:  [1, 3, 2, 7, 4, 6]
num2:  2
op_in: True
```

op_not_in

- Syntax

```
op_not_in(Container, Element)
```

- Parameters

Parameter	Type	Required	Description
Container	String, tuple, list, or dictionary	Yes	The name of a container. The value of this parameter can be a string, tuple, list, or dictionary.
Element	Arbitrary	Yes	The name of an element.

> **Note** In this function, the a parameter that specifies a container precedes the b parameter that specifies an element

- Response

  If the specified container does not contain the specified element, the value True is returned. Otherwise, the value False is returned.

- Example

  Raw log entry:

```
list:  [1, 3, 2, 7, 4, 6]
num2:  12
```

  Transformation rule:

```
e_set("op_not_in",op_not_in(v("list"),v("num2")))
```

  Result:

```
list:  [1, 3, 2, 7, 4, 6]
num2:  12
op_in: True
```

op_slice

- Syntax

```
op_slice(Value, start=None, end=None, step)
```

- Parameters

Parameter	Type	Required	Description
Value	String	Yes	The field value that you want to truncate.
start	Num	No	The position from which the value of the specified field is truncated. Default value: 0.
end	Num	No	The position to which the value of the specified field is truncated. The character in this position is not truncated. The position ends at the end of the specified string by default.

Parameter	Type	Required	Description
step	Num	No	The number of characters that you want to truncate each time.

- Response

  The substrings that are truncated from the value of the specified field are returned.

- Examples
  - Example 1: The value of the word field is truncated from the start to the end at a step of 2.

    Raw log entry:

    ```
    word:  I,love,this,world
    ```

    Transformation rule:

    ```
    e_set("op_slice",op_slice(v("word"),2))
    ```

    Result:

    ```
    word:  I,love,this,world
    op_slice: I,
    ```

  - Example 2: The value of the word field is truncated from position 2 to position 9 at a step of 1.

    Raw log entry:

    ```
    word:  I,love,this,world
    ```

    Transformation rule:

    ```
    e_set("op_slice",op_slice(v("word"),2,9,1))
    ```

    Result:

    ```
    word:  I,love,this,world
    op_slice: love,th
    ```

op_index

- Syntax

  ```
  op_index (Value, index)
  ```

- Parameters

Parameter	Type	Required	Description
Value	String	Yes	The field value that you want to truncate.

Parameter	Type	Required	Description
index	Num	No	The index of the specified string, array, or tuple.

- Response

  The element that corresponds to the index is returned.

- Examples
  - Example 1: The element whose index is 0 in the value of the word field is returned.

    Raw log entry:

    ```
    word:  I,love,this,world
    ```

    Transformation rule:

    ```
    e_set("op_index",op_index(v("word"),0))
    ```

    Result:

    ```
    word:  I,love,this,world
    op_slice: I,
    ```

  - Example 2: The element whose index is 3 in the value of the word field is returned.

    Raw log entry:

    ```
    word:  I,love,this,world
    ```

    Transformation rule:

    ```
    e_set("op_index",op_index(v("word"),3))
    ```

    Result:

    ```
    word:  I,love,this,world
    op_index: o
    ```

op_add

- Syntax

  ```
  op_add(Value 1, Value 2, ...)
  ```

- Parameters

Parameter	Type	Required	Description
Value 1	String, tuple, list, or dictionary	Yes	The field value on which you want to perform the ADD operation.

Parameter	Type	Required	Description
Value 2	Same as the data type of Value 1	Yes	The field value on which you want to perform the ADD operation.

- Response

  The sum of the specified values is returned.

- Examples
  - Example 1: Calculate the sum of the values of the price_orange and price_apple fields to obtain the total price.

    Raw log entry:

    ```
    price_orange: 2
    price_apple: 13
    ```

    Transformation rule:

    ```
    e_set("account",op_add(ct_int(v("price_orange")),ct_int(v("price_apple"))))
    ```

    Result:

    ```
    price_orange: 2
    price_apple: 13
    account:  15
    ```

  - Example 2: Calculate the sum of the values of the bytes_in and bytes_out fields to obtain the total number of bytes.

    Raw log entry:

    ```
    bytes_in: 214
    bytes_out: 123
    ```

    Transformation rule:

    ```
    e_set("total_bytes", op_add(ct_int(v("bytes_in"), ct_int(v("bytes_out")))))
    ```

    Result:

    ```
    bytes_in: 214
    bytes_out: 123
    total_bytes:  337
    ```

- Example 3: Add the https:// prefix to a website URL.

  Raw log entry:

  ```
  host: aliyun.com
  ```

  Transformation rule:

  ```
  e_set("website", op_add("https://", v("host")))
  ```

  Result:

  ```
  host: aliyun.com
  website: https://aliyun.com
  ```

op_max

- Syntax

  ```
  op_max(Value 1, Value 2, ...)
  ```

- Parameters

Parameter	Type	Required	Description
Value 1	Arbitrary	Yes	The value of a field.
Value 2	Same as the data type of Value 1	Yes	The value of a field.

- Response

  The maximum value among the specified values is returned.

- Example

  Raw log entry:

  ```
  price_orange:  2
  priority_apple:  13
  ```

  Transformation rule:

  ```
  e_set("max_price", op_max(ct_int(v("price_orange")),ct_int(v("priority_apple"))))
  ```

  Result:

  ```
  price_orange:  2
  priority_apple:  13
  max_price:  13
  ```

op_min

- Syntax

  ```
  op_min(Value 1, Value 2, ...)
  ```

- Parameters

Parameter	Type	Required	Description
Value 1	Arbitrary	Yes	The value of a field.
Value 2	Same as the data type of Value 1	Yes	The value of a field.

- Response

  The minimum value among the specified values is returned.

- Example

  Raw log entry:

```
price_orange:  2
price_apple:  13
```

  Transformation rule:

```
e_set("op_min", op_min(ct_int(v("price_orange")),ct_int(v("price_apple"))))
```

  Result:

```
price_orange:  2
price_apple:  13
op_min:  2
```

12.6.4. Conversion functions

This topic describes the syntax and parameters of conversion functions. This topic also provides examples on how to use the functions.

Functions

Type	Function	Description
Basic type conversion	ct_int	Converts the value of a field or an expression to an integer.
	ct_float	Converts the value of a field or an expression to a floating-point number.
	ct_str	Converts the value of a field or an expression to a string.
	ct_bool	Converts the value of a field or an expression to a Boolean value.
	ct_chr	Converts the ANSI or Unicode value of a field or an expression to a character.
	ct_ord	Converts the value of a field or an expression to an ANSI or a Unicode value.

Type	Function	Description
Number conversion	ct_hex	Converts the value of a field or an expression to a hexadecimal number.
	ct_oct	Converts the value of a field or an expression to an octal number.
	ct_bin	Converts the value of a field or an expression to a binary number.
Numeral system conversion	bin2oct	Converts a binary string to an octal string.
	bin2hex	Converts a binary string to a hexadecimal string.

ct_int

The ct_int function is used to convert the value of a field or an expression to an integer.

- Syntax

```
ct_int(value, base=10)
```

- Parameters

Parameter	Type	Required	Description
value	Number or numeric string	Yes	The value that you want to convert.
base	Number	No	The numeral system. Default value: 10. This value indicates the decimal numeral system. If you set the base parameter to 8, this function converts an octal value to a decimal value.

- Response

  An integer is returned.

- Examples
  - Example 1: Convert a string to an integer.
    - Raw log entry:

```
number: 2
```

    - Transformation rule:

```
e_set("int_number", ct_int(v("number")))
```

    - Result:

```
number: 2
int_number:  2
```

- Example 2: Convert a hexadecimal value to a decimal value.
  - Raw log entry:

    ```
    number: AB
    ```

  - Transformation rule:

    ```
    e_set("int_number", ct_int(v("number"),base=16))
    ```

  - Result:

    ```
    number: AB
    int_number:  171
    ```

ct_float

The ct_float function is used to convert the value of a field or an expression to a floating-point number.

- Syntax

  ```
  ct_float(value)
  ```

- Parameters

Parameter	Type	Required	Description
value	Number or numeric string	Yes	The value that you want to convert.

- Response

  A floating-point number is returned.

- Examples
  - Raw log entry:

    ```
    price: 2
    ```

  - Transformation rule:

    ```
    e_set("price_float", ct_float(v("price")))
    ```

  - Result:

    ```
    price: 2
    price_float:  2.0
    ```

ct_str

The ct_str function is used to convert the value of a field or an expression to a string.

- Syntax

  ```
  ct_str(value)
  ```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary	Yes	The value that you want to convert.

- Response

  A string is returned.

- Example
  - Transformation rule:

    ```
    e_set("ct_str", ct_str(b'test byte'))
    ```

  - Result:

    ```
    ct_str: test byte
    ```

ct_bool

The ct_bool function is used to convert the value of a field or an expression to a Boolean value. For information about the true or false evaluation of different data types, see True or false evaluation.

- Syntax

  ```
  ct_bool(value)
  ```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary	Yes	The value that you want to convert.

- Response

  A Boolean value is returned.

- Example
  - Raw log entry:

    ```
    num: 2
    ```

  - Transformation rule:

    ```
    e_set("ct_bool", ct_bool(v("num")))
    ```

  - Result:

    ```
    num: 2
    ct_bool: true
    ```

ct_chr

The ct_chr function is used to convert the ANSI or Unicode value of a field or an expression to a character.

- Syntax

```
ct_chr(value)
```

- Parameters

Parameter	Type	Required	Description
value	Number or numeric string	Yes	The value that you want to convert.

- Response

  A character is returned.

- Example
  - Raw log entry:

```
number: 78
```

  - Transformation rule:

```
e_set("ct_chr", ct_chr(v("number")))
```

  - Result:

```
number: 78
ct_chr: N
```

ct_ord

The ct_ord function is used to convert the value of a field or an expression to an ANSI or a Unicode value.

- Syntax

```
ct_ord(value)
```

- Parameters

Parameter	Type	Required	Description
value	String	Yes	The value that you want to convert. The value contains only one character.

- Response

  An ANSI or a Unicode value is returned.

- Example
  - Raw log entry:

```
world: a
```

- Transformation rule:

```
e_set("ct_ord", ct_ord(v("world")))
```

- Result:

```
world: a
ct_ord: 97
```

ct_hex

The ct_hex function is used to convert the value of a field or an expression to a hexadecimal number.

- Syntax

```
ct_hex(value)
```

- Parameters

Parameter	Type	Required	Description
value	Number or numeric string	Yes	The value that you want to convert.

- Response

  A hexadecimal number is returned.

- Example
  - Raw log entry:

```
number: 123
```

  - Transformation rule:

```
e_set("ct_hex", ct_hex(v("number")))
```

  - Result:

```
number: 123
ct_hex: 0x7b
```

ct_oct

The ct_oct function is used to convert the value of a field or an expression to an octal number.

- Syntax

```
ct_oct(value)
```

- Parameters

Parameter	Type	Required	Description
value	Number or numeric string	Yes	The value that you want to convert.

- Response

  An octal number is returned.

- Example
  - Raw log entry:

    ```
    number: 123
    ```

  - Transformation rule:

    ```
    e_set("ct_oct", ct_oct(v("number")))
    ```

  - Result:

    ```
    number: 123
    ct_oct: 0o173
    ```

ct_bin

The ct_bin function is used to convert the value of a field or an expression to a binary number.

- Syntax

  ```
  ct_bin(value)
  ```

- Parameters

Parameter	Type	Required	Description
value	Number or numeric string	Yes	The value that you want to convert.

- Response

  A binary number is returned.

- Example
  - Raw log entry:

    ```
    number: 123
    ```

  - Transformation rule:

    ```
    e_set("ct_bin", ct_bin(v("number")))
    ```

  - Result:

    ```
    number: 123
    ct_bin: 0b1111011
    ```

bin2oct

The bin2oct function is used to convert a binary string to an octal string.

- Syntax

  ```
  bin2oct(binary)
  ```

- Parameters

Parameter	Type	Required	Description
binary	Binary	Yes	The binary string that you want to convert.

- Response

  An octal string is returned.

- Example
  - Raw log entry:

    ```
    test : test
    ```

  - Transformation rule:

    ```
    e_set("new",bin2oct(base64_decoding("ARi8WnFiLAAACHcAGgkADV37Xs8BXftezgAdqwF9")))
    ```

  - Result:

    ```
    test : test
    new : 21427426470542130000000207340006404400032567732754740127375536634000355260057 5
    ```

bin2hex

The bin2hex function is used to convert a binary string to a hexadecimal string.

- Syntax

  ```
  bin2hex(binary)
  ```

- Parameters

Parameter	Type	Required	Description
binary	Binary	Yes	The binary string that you want to convert.

- Response

  A hexadecimal string is returned.

- Example
  - Raw log entry:

    ```
    test : test
    ```

  - Transformation rule:

    ```
    e_set("new",bin2hex(base64_decoding("ARi8WnFiLAAACHcAGgkADV37Xs8BXftezgAdqwF9")))
    ```

  - Result:

    ```
    test : test
    new :0118bc5a71622c00000877001a09000d5dfb5ecf015dfb5ece001dab017d
    ```

12.6.5. Arithmetic functions

This topic describes the syntax and parameters of arithmetic functions. This topic also provides examples on how to use the functions.

> **Note** If you want to obtain a negative value, use the op_neg (positive value) function. For example, use op_neg(1) to return `-1`.

Functions

Type	Function	Description
Sum calculation	op_sum	Returns the sum of passed values.
Basic calculation	op_abs	Returns the absolute value of a passed value.
	op_div_floor	Returns the integer part of the quotient of two passed values.
	op_div_true	Returns the quotient of two passed values.
	op_pow	Returns a value raised to a specified power.
	op_mul	Returns the product of two passed values.
	op_neg	Returns the opposite number of a passed value.
	op_mod	Returns the remainder of a passed value divided by the other passed value.
	op_sub	Returns the difference between two passed values.
	op_round	Returns a passed value rounded.
	mat_ceil	Returns a passed value rounded up to the nearest integer.
	mat_exp	Returns Euler's number raised to the power of a passed value.
	mat_fabs	Returns the absolute value of a passed value.
	mat_floor	Returns a passed value rounded down to the nearest integer.
	mat_log	Returns the logarithm of a passed value with the other passed value as the base.
	mat_log10	Returns the base-10 logarithm of a passed value.
	mat_sqrt	Returns the square root of a passed value.

Type	Function	Description
Mathematical calculation	mat_degrees	Converts radians to degrees.
	mat_radians	Converts degrees to radians.
	mat_sin	Returns the sine of a passed value (in radians).
	mat_cos	Returns the cosine of a passed value (in radians).
	mat_tan	Returns the tangent of a passed value (in radians).
	mat_acos	Returns the arc cosine (in radians) of a passed value.
	mat_asin	Returns the arc sine (in radians) of a passed value.
	mat_atan	Returns the arc tangent (in radians) of a passed value.
	mat_atan2	Returns the arc tangent of the X-coordinate and the Y-coordinate.
	mat_atanh	Returns the inverse hyperbolic tangent of a passed value.
	mat_hypot	Returns the Euclidean norm of two passed values.
	MATH_PI	Obtains the constant pi.
	MATCH_E	Obtains the constant e.

op_sum

- Syntax

```
op_sum(Value 1, Value 2, ...)
```

- Parameters

Parameter	Type	Required	Description
Value 1	Number or numeric string	Yes	The value that you want to use for the calculation.
Value 2	Number or numeric string	Yes	The value that you want to use for the calculation.

- Response

  The sum of all passed values is returned.

- Example: Calculate the sum of the values of the course_price and goods_price fields.

  Raw log entry:

```
course_price: 12
goods_price: 2
```

Transformation rule:

```
e_set("account", op_sum(v("course_price"),v("goods_price")))
```

Result:

```
course_price: 12
goods_price: 2
account: 14
```

op_abs

- Syntax

```
op_abs(Value)
```

- Parameters

Parameter	Type	Required	Description
Value	Number or numeric string	Yes	The value that you want to use for the calculation.

- Response

  The absolute value of the passed value is returned.

- Example: Calculate the absolute value of the value of the course_price field.

  Raw log entry:

```
course_price: -4
```

  Transformation rule:

```
e_set("op_abs", op_abs(v("course_price")))
```

  Result:

```
course_price: -4
op_abs: 4
```

op_div_floor

- Syntax

```
op_div_floor(Value 1, Value 2)
```

- Parameters

Parameter	Type	Required	Description
Value 1	Number or numeric string	Yes	The value that you want to use for the calculation.
Value 2	Number or numeric string	Yes	The value that you want to use for the calculation.

- Response

  The integer part of the quotient that is obtained after Value 1 is divided by Value 2 is returned.

- Example: Calculate the unit price based on the value of the course_price field.

  Raw log entry:

```
course_price: 4
count: 2
```

  Transformation rule:

```
e_set("op_div_floor", op_div_floor(v("course_price"),v("count")))
```

  Result:

```
course_price: 4
count: 2
op_div_floor: 2
```

op_div_true

The op_div_true function is used to calculate the quotient of two passed values.

> **Note** This function automatically converts the data types of passed values. The value that you pass can be a string or an integer.

- Syntax

```
op_div_true(Value 1, Value 2)
```

- Parameters

Parameter	Type	Required	Description
Value 1	Number or numeric string	Yes	The value that you want to use for the calculation.
Value 2	Number or numeric string	Yes	The value that you want to use for the calculation.

- Response

The quotient that is obtained after Value 1 is divided by Value 2 is returned.

- Examples
  - Example 1: Calculate the unit price based on the value of the fruit_price field.

    ```
    e_set("op_div_true", op_div_true(v("fruit_price"),v("count")))
    ```

    Raw log entry:

    ```
    fruit_price: 9
    count: 2
    ```

    Result:

    ```
    fruit_price: 9
    count: 2
    op_div_true: 4.5
    ```

  - Example 2: Calculate the acceleration based on the values of the one_speed and two_speed fields. The returned value is rounded. Formula: a = (one_speed - two_speed)/time.

    ```
    e_set("a", op_round(op_div_true(op_sub(v("one_speed"),v("two_speed")),v("time")),2))
    ```

    Raw log entry:

    ```
    one_speed: 9
    two_speed: 2
    time: 3
    ```

    Result:

    ```
    a:2.33
    one_speed:9
    time:3
    two_speed:2
    ```

op_pow

- Syntax

  ```
  op_pow(Value 1,Value 2)
  ```

- Parameters

Parameter	Type	Required	Description
Value 1	Number or numeric string	Yes	The value that you want to use for the calculation.
Value 2	Number or numeric string	Yes	The value that you want to use for the calculation.

- Response

Value 1 raised to the power of Value 2 is returned.

- Example: Calculate the value of the course field raised to the power of the value of the pow field.

  Raw log entry:

  ```
  course: 100
  pow: 2
  ```

  Transformation rule:

  ```
  e_set("pow_course", op_pow(v("course"),v("pow")))
  ```

  Result:

  ```
  course: 100
  pow: 2
  pow_course: 10000
  ```

op_mul

- Syntax

  ```
  op_mul(Value 1,Value 2)
  ```

- Parameters

Parameter	Type	Required	Description
Value 1	Number, string, tuple, or list	Yes	The value that you want to use for the calculation.
Value 2	Number	Yes	The value that you want to use for the calculation.

- Response

  The product of Value 1 and Value 2 is returned.

  - If Value 1 is a number, the product of Value 1 and Value 2 is returned.
  - If Value 1 is a string, tuple, or list, the result is the original value repeated for the specified times.

- Examples

- Example 1

  Raw log entry:

  ```
  course: 10
  price: 23
  ```

  Transformation rule:

  ```
  e_set("account", op_mul(ct_int(v("course")),ct_int(v("price"))))
  ```

  Result:

  ```
  course: 10
  price: 23
  account: 230
  ```

- Example 2

  Raw log entry:

  ```
  course: "abc"
  ```

  Transformation rule:

  ```
  e_set("course", op_mul(v("course"), 3))
  ```

  Result:

  ```
  course: "abcabcabc"
  ```

op_neg

- Syntax

  ```
  op_neg(Value)
  ```

- Parameters

Parameter	Type	Required	Description
Value	Number or numeric string	Yes	The value that you want to use for the calculation.

- Response

  The opposite number of the passed value is returned.

- Example

  Raw log entry:

  ```
  course: -100
  ```

  Transformation rule:

  ```
  e_set("account", op_neg(v("course_price")))
  ```

Result:

```
course: -100
account: 100
```

op_mod

- Syntax

```
op_mod(Value 1,Value 2)
```

- Parameters

Parameter	Type	Required	Description
Value 1	Number or numeric string	Yes	The value that you want to use for the calculation.
Value 2	Number or numeric string	Yes	The value that you want to use for the calculation.

- Response

  The remainder that is obtained after Value 1 is divided by Value 2 is returned.

- Example

  Raw log entry:

```
course: 4
count: 3
```

  Transformation rule:

```
e_set("op_mod", op_mod(v("course"),v("count")))
```

  Result:

```
course: 4
count: 3
op_mod: 1
```

op_sub

- Syntax

```
op_sub(Value 1,Value 2)
```

- Parameters

Parameter	Type	Required	Description

Parameter	Type	Required	Description
Value 1	Number or numeric string	Yes	The value that you want to use for the calculation.
Value 2	Number or numeric string	Yes	The value that you want to use for the calculation.

- Response

  The difference between Value 1 and Value 2 is returned.

- Example: Calculate the difference between the value of the count field and the value of the count_apple field.

  Raw log entry:

```
count: 6
count_apple: 3
```

  Transformation rule:

```
e_set("sub_number", op_sub(v("count"),v("count_apple")))
```

  Result:

```
count: 6
count_apple: 3
sub_number:  3
```

op_round

- Syntax

```
op_round(Value, Number of decimal places)
```

- Parameters

Parameter	Type	Required	Description
Value	Number or numeric string	Yes	The value that you want to use for the calculation.
Decimal place	Number	Yes	The number of decimal places to which the value is rounded. Default value: 0.

- Response

  The passed value rounded is returned.

- Example: Round the value of the price field to one decimal place.

Raw log entry:

```
price: 4.56
```

Transformation rule:

```
e_set("round_price", op_round(v("price"),1))
```

Result:

```
price: 4.56
round_price: 4.6
```

mat_ceil

- Syntax

```
mat_ceil(Value)
```

- Parameters

Parameter	Type	Required	Description
Value	Number or numeric string	Yes	The value that you want to use for the calculation.

- Response

  The smallest integer that is not less than the passed value is returned.

- Example

  Raw log entry:

```
price: 4.1
```

  Transformation rule:

```
e_set("mat_ceil", mat_ceil(v("price")))
```

  Result:

```
price: 4.1
mat_ceil: 5
```

mat_exp

- Description

  The mat_exp function is used to calculate Euler's number raised to the power of a passed value.

- Syntax

```
mat_exp(Value)
```

- Parameters

Parameter	Type	Required	Description
Value	Number or numeric string	Yes	The value that you want to use for the calculation.

- Response

  Euler's number raised to the power of the passed value is returned.

- Example

  Raw log entry:

  ```
  e: 1
  ```

  Transformation rule:

  ```
  e_set("e_x", mat_exp(v("e")))
  ```

  Result:

  ```
  e: 1
  e_x: 2.718281828459045
  ```

mat_fabs

- Description

  The mat_fabs function is used to calculate the absolute value of a passed value.

- Syntax

  ```
  mat_fabs(Value)
  ```

- Parameters

Parameter	Type	Required	Description
Value	Number or numeric string	Yes	The value that you want to use for the calculation.

- Response

  The absolute value of the passed value is returned.

- Example

  Raw log entry:

  ```
  course_price: -10
  ```

  Transformation rule:

  ```
  e_set("mat_fabs", mat_fabs(v("course_price")))
  ```

  Result:

```
course_price: -10
mat_fabs: 10.0
```

mat_floor

- Description

  The mat_floor function is used to calculate a passed value rounded down to the nearest integer.

- Syntax

```
mat_floor(Value)
```

- Parameters

Parameter	Type	Required	Description
Value	Number or numeric string	Yes	The value that you want to use for the calculation.

- Response

  The largest integer that is not greater than the passed value is returned.

- Example

  Raw log entry:

```
course_price: 4.9
```

  Transformation rule:

```
e_set("mat_floor", mat_floor(v("course_price")))
```

  Result:

```
course_price: 4.9
mat_floor: 4
```

mat_log

- Description

  The mat_log function is used to calculate the logarithm of a passed value with the other passed value as the base.

- Syntax

```
mat_log(Value 1,Value 2)
```

- Parameters

Parameter	Type	Required	Description
Value 1	Number or numeric string	Yes	The value that you want to use for the calculation.

Parameter	Type	Required	Description
Value 2	Number or numeric string	Yes	The value that you want to use for the calculation.

- Response

  The logarithm of Value 1 with base Value 2 is returned.

- Example

  Raw log entry:

```
number1: 100
number2: 10
```

  Transformation rule:

```
e_set("mat_log", mat_log(v("number1"),v("number2")))
```

  Result:

```
number1: 100
number2: 10
mat_log: 2.0
```

mat_log10

- Description

  The mat_log10 function is used to calculate the base-10 logarithm of a passed value.

- Syntax

```
mat_log10(Value)
```

- Parameters

Parameter	Type	Required	Description
Value	Number or numeric string	Yes	The value that you want to use for the calculation.

- Response

  The base-10 logarithm of the passed value is returned.

- Example

  Raw log entry:

```
number: 100
```

  Transformation rule:

```
e_set("number2", mat_log10(v("number")))
```

Result:

```
number: 100
numbe2: 2.0
```

mat_sqrt

- Description

  The mat_sqrt function is used to calculate the square root of a passed value.

- Syntax

```
mat_sqrt(Value)
```

- Parameters

Parameter	Type	Required	Description
Value	Number or numeric string	Yes	The value that you want to use for the calculation.

- Response

  The square root of the passed value is returned.

- Example

  Raw log entry:

```
number1: 100
```

  Transformation rule:

```
e_set("sqrt_account", mat_sqrt(v("number1")))
```

  Result:

```
number1: 100
sqrt_account: 10.0
```

mat_degrees

- Description

  The mat_degrees function is used to convert radians to degrees.

- Syntax

```
mat_degrees(Value)
```

- Parameters

Parameter	Type	Required	Description

Parameter	Type	Required	Description
Value	Number or numeric string	Yes	The value that you want to use for the calculation.

- Response

  The degrees are returned.

- Example

  Raw log entry:

  ```
  num: 1
  ```

  Transformation rule:

  ```
  e_set("mat_degrees", mat_degrees(v("num")))
  ```

  Result:

  ```
  num: 1
  mat_degrees: 57.29577951308232
  ```

mat_radians

- Description

  The mat_radians function is used to convert degrees to radians.

- Syntax

  ```
  mat_radians(Value)
  ```

- Parameters

Parameter	Type	Required	Description
Value	Number or numeric string	Yes	The value that you want to use for the calculation.

- Response

  The radians are returned.

- Example

  Raw log entry:

  ```
  rad: 30
  ```

  Transformation rule:

  ```
  e_set("mat_radians", mat_radians(v("rad")))
  ```

  Result:

```
rad: 30
mat_radians: 0.5235987755982988
```

mat_sin

- Description

  The mat_sin function is used to calculate the sine of a passed value (in radians).

- Syntax

```
mat_sin(Value)
```

- Parameters

Parameter	Type	Required	Description
Value	Number or numeric string	Yes	The value that you want to use for the calculation.

- Response

  The sine of the passed value (in radians) is returned.

- Example

  Raw log entry:

```
sin: 90
```

  Transformation rule:

```
e_set("mat_sin", mat_sin(v("sin")))
```

  Result:

```
sin: 90
mat_sin: 0.8939966636005579
```

mat_cos

- Description

  The mat_cos function is used to calculate the cosine of a passed value (in radians).

- Syntax

```
mat_cos(Value)
```

- Parameters

Parameter	Type	Required	Description
Value	Number or numeric string	Yes	The value that you want to use for the calculation.

- Response

  The cosine of the passed value (in radians) is returned.

- Example

  Raw log entry:

  ```
  cos: 30
  ```

  Transformation rule:

  ```
  e_set("mat_cos", mat_cos(v("cos")))
  ```

  Result:

  ```
  cos: 30
  mat_cos: 0.15425144988758405
  ```

mat_tan

- Description

  The mat_tan function is used to calculate the tangent of a passed value (in radians).

- Syntax

  ```
  mat_tan(Value)
  ```

- Parameters

Parameter	Type	Required	Description
Value	Number or numeric string	Yes	The value that you want to use for the calculation.

- Response

  The tangent of the passed value (in radians) is returned.

- Example

  Raw log entry:

  ```
  tan: 30
  ```

  Transformation rule:

  ```
  e_set("mat_tan", mat_tan(v("tan")))
  ```

  Result:

  ```
  tan: 30
  mat_tan: 1.6197751905438615
  ```

mat_acos

- Description

The mat_acos function is used to calculate the arc cosine (in radians) of a passed value.

- Syntax

```
mat_acos(Value)
```

- Parameters

Parameter	Type	Required	Description
Value	Number or numeric string	Yes	The value that you want to use for the calculation.

- Response

  The arc cosine (in radians) of the passed value is returned.

- Example

  Raw log entry:

```
acos: 1
```

  Transformation rule:

```
e_set("mat_acos", mat_acos(v("acos")))
```

  Result:

```
acos: 1
mat_acos: 0.0
```

mat_asin

- Description

  The mat_asin function is used to calculate the arc sine (in radians) of a passed value.

- Syntax

```
mat_asin(Value)
```

- Parameters

Parameter	Type	Required	Description
Value	Number or numeric string	Yes	The value that you want to use for the calculation.

- Response

  The arc sine (in radians) of the passed value is returned.

- Example

  Raw log entry:

```
asin: 1
```

Transformation rule:

```
e_set("mat_asin", mat_asin(v("asin")))
```

Result:

```
asin: 1
mat_asin: 1.5707963267948966
```

mat_atan

- Description

  The mat_atan function is used to calculate the arc tangent (in radians) of a passed value.

- Syntax

```
mat_atan(Value)
```

- Parameters

Parameter	Type	Required	Description
Value	Number or numeric string	Yes	The value that you want to use for the calculation.

- Response

  The arc tangent (in radians) of the passed value is returned.

- Example

  Raw log entry:

```
atan: 1
```

  Transformation rule:

```
e_set("mat_atan", mat_atan(v("atan")))
```

  Result:

```
atan: 1
mat_atan: 0.7853981633974483
```

mat_atan2

- Description

  The mat_atan2 function is used to calculate the arc tangent of the X-coordinate and the Y-coordinate.

- Syntax

```
mat_atan2(x,y)
```

- Parameters

Parameter	Type	Required	Description
x	Number or numeric string	Yes	The X-coordinate.
y	Number or numeric string	Yes	The Y-coordinate.

- Response

  The arc tangent of the X-coordinate and the Y-coordinate is returned.

- Example

  Raw log entry:

```
atan1: 1
atan2: 2
```

  Transformation rule:

```
e_set("mat_atan2", mat_atan2(v("atan1"),v("atan2")))
```

  Result:

```
atan1: 1
atan2: 2
mat_atan2: 0.4636476090008061
```

mat_atanh

- Description

  The mat_atanh function is used to calculate the inverse hyperbolic tangent of a passed value.

- Syntax

```
mat_atanh(Value)
```

- Parameters

Parameter	Type	Required	Description
Value	Number or numeric string	Yes	The X-coordinate.

- Response

  The inverse hyperbolic tangent of the passed value is returned.

- Example

  Raw log entry:

```
atanh: 0.5
```

Transformation rule:

```
e_set("mat_atanh", mat_atanh(v("atanh")))
```

Result:

```
atanh:0.5
mat_atanh:0.5493061443340548
```

mat_hypot

- Description

  The mat_hypot function is used to calculate the Euclidean norm of two passed values.

- Syntax

```
mat_hypot(Value 1,Value 2)
```

- Parameters

Parameter	Type	Required	Description
Value 1	Number or numeric string	Yes	The X-coordinate.
Value 2	Number or numeric string	Yes	The Y-coordinate.

- Response

  The Euclidean norm of the passed values is returned.

- Example

  Raw log entry:

```
hypot1: 1
hypot2: 2
```

  Transformation rule:

```
e_set("mat_hypot", mat_hypot(v("hypot1"),v("hypot2")))
```

  Result:

```
hypot1:1
hypot2:2
mat_hypot:2.23606797749979
```

12.6.6. String functions

This topic describes the syntax and parameters of string functions. This topic also provides examples on how to use the functions.

Functions

Category	Function	Description
Multi-string operation	str_format	Formats strings.
	str_join	Concatenates input values by using a specified connector to generate a new string.
	str_zip	Concurrently splits two values or strings that are returned by expressions and combines the results to a string.
Encoding and decoding	str_encode	Encodes a string by using a specified encoding format.
	str_decode	Decodes an input value by using a specified encoding format.
	str_hex_escape_encode	Escapes special characters. The function can escape hexadecimal characters to Chinese characters.
	str_uuid	Generates a random UUID.
Sorting, reversing, and replacement	str_sort	Sorts a specified object.
	str_reverse	Reverses a string.
	str_replace	Replaces an old string with a new string based on a specified rule.
	str_logstash_config_normalize	Converts data in the Logstash configuration language to the JSON format.
	str_translate	Replaces specified characters in a string with mapping characters.
Regular munging	str_strip	Deletes specified characters from a string.
	str_lstrip	Deletes specified characters from the start of a string.
	str_rstrip	Deletes specified characters from the end of a string.
	str_lower	Converts all uppercase letters in a string to lowercase letters.
	str_upper	Converts all lowercase letters in a string to uppercase letters.
	str_title	Capitalizes the first letter of each word in a string and converts the other letters in the string to lowercase letters.

Category	Function	Description
	str_capitalize	Capitalizes the first letter of a string and converts the other letters in the string to lowercase letters.
	str_swapcase	Interchanges the uppercase letters and lowercase letters in a string.
Search and check	str_count	Counts the number of occurrences of a character in a string.
	str_find	Checks whether a string contains a specified substring.
	str_rfind	Returns the position of the last occurrence of a specified character in a string.
	str_endswith	Checks whether a string ends with a specified suffix.
	str_startswith	Checks whether a string starts with a specified string.
Splitting	str_split	Splits a string by using a specified delimiter.
	str_splitlines	Splits a string by using a line feed.
	str_partition	Splits a string into three parts from left to right by using a specified delimiter.
	str_rpartition	Splits a string into three parts from right to left by using a specified delimiter.
Formatting	str_center	Pads a string to a specified length by using a specified character.
	str_ljust	Pads a string to a specified length by using a specified character from the end of the string.
	str_rjust	Pads a string to a specified length by using a specified character from the start of the string.
	str_zfill	Pads a string to a specified length by using 0 from the start of the string.
	str_expandtabs	Converts `\t` in a string to spaces.
	str_isalnum	Checks whether a string contains only letters and digits.
	str_isalpha	Checks whether a string contains only letters.
	str_isascii	Checks whether a string is in the ASCII table.

Category	Function	Description
Character set check	str_isdecimal	Checks whether a string contains only decimal characters.
	str_isdigit	Checks whether a string contains only digits.
	str_isidentifier	Checks whether a string is a valid Python identifier or checks whether a variable name is valid.
	str_islower	Checks whether a string contains lowercase letters.
	str_isnumeric	Checks whether a string contains digits.
	str_isprintable	Checks whether all characters in a string are printable characters.
	str_isspace	Checks whether a string contains only spaces.
	str_istitle	Checks whether the first letter of each word in a string is in uppercase and the other letters in the string are in lowercase.
	str_isupper	Checks whether all letters in a string are in uppercase.

The following table describes the functions that can be used together with string functions.

Category	Function	Description
Multi-string operation	op_add	Returns the sum value among multiple numeric values or strings.
	op_max	Returns the maximum value among multiple numeric values or strings.
	op_min	Returns the minimum value among multiple numeric values or strings.
String truncation	op_slice	Truncates a string.
Length calculation	op_len	Returns the length of a string.

str_format

- Syntax

```
str_format(formatting string, value 1, value 2, ...)
```

- Parameters

Parameter	Type	Required	Description

Parameter	Type	Required	Description
formatting string	Arbitrary (automatically converted to the string type)	Yes	The format of the output string. Example: `{}={}`.
value 1	Arbitrary	Yes	The value that you want to format.
value 2	Arbitrary	Yes	The value that you want to format.

- Response

  A formatted string is returned.

- Examples

  Raw log:

```
class: Format
escape_name: Traditional
```

  Transformation rule:

```
e_set("str_format", str_format("{}={}",v("class"),v("escape_name")))
```

  Result:

```
class: Format
escape_name: Traditional
str_format: Format=Traditional
```

str_join

- Syntax

```
str_join(connector, value 1, value 2, ...)
```

- Parameters

Parameter	Type	Required	Description
connector	Arbitrary (automatically converted to the string type)	Yes	The connector. Supported connectors include the exclamation point (!), at sign (@), number sign (#), dollar sign ($), and percent sign (%).
value 1	Arbitrary (automatically converted to the string type)	Yes	The value that you want to concatenate.
value 2	Arbitrary (automatically converted to the string type)	Yes	The value that you want to concatenate.

- Response

  A concatenated string is returned.

- Examples

  Raw log:

```
name: ETL
company: aliyun.com
```

  Transformation rule:

```
e_set("email", str_join("@",v("name"),v("company")))
```

  Result:

```
name: ETL
company: aliyun.com
email:ETL@aliyun.com
```

str_encode

- Syntax

```
str_encode(value, "utf8", errors="ignore"))
```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The value that you want to encode.
encoding	String	No	The encoding format. Default value: utf8. ASCII is supported.
errors	String	No	The method that is used to process characters if the characters cannot be recognized based on the encoding format. Valid values: ◦ ignore: The characters are not encoded. This is the default value. ◦ strict: An error is reported, and the data of the log is discarded. ◦ replace: The characters are replaced with question marks (?). ◦ xmlcharrefreplace: The characters are replaced with the related XML characters.

- Response

  An encoded string is returned.

- Examples
  - Example 1

    Raw log:

    ```
    test: asewds
    ```

    Transformation rule:

    ```
    e_set("f1", str_decode(str_encode("hello", "utf8"), "utf8"))
    ```

    Result:

    ```
    test: asewds
    f1: hello
    ```

  - Example 2

    Raw log:

    ```
    f2: test test data
    ```

    Transformation rule:

    ```
    e_set("f1", str_encode(v("f2"), "ascii", errors="ignore"))
    ```

    Result:

    ```
    f1:test
    f2:test test data
    ```

  - Example 3

    Raw log:

    ```
    f2: test test data
    ```

    Transformation rule:

    ```
    e_set("f1", str_encode(v("f2"), "ascii", errors="strict"))
    ```

    Result:

    ```
    An error is reported during execution.
    ```

- Example 4

  Raw log:

  ```
  f2: test test data
  ```

  Transformation rule:

  ```
  e_set("f1", str_encode(v("f2"), "ascii", errors="replace"))
  ```

  Result:

  ```
  f1:test ????
  f2:test test data
  ```

- Example 5

  Raw log:

  ```
  f2: test test data
  ```

  Transformation rule:

  ```
  e_set("f1", str_encode(v("f2"), "ascii", errors="xmlcharrefreplace"))
  ```

  Result:

  ```
  f1:test &#27979;&#35797;&#25968;&#25454;
  f2:test test data
  ```

str_decode

- Syntax

  ```
  str_decode(value, "utf8", errors="ignore"))
  ```

  > **Note** The str_decode function can process only the data of the byte data type.

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The value that you want to decode.
encoding	Arbitrary (automatically converted to the string type)	No	The encoding format. Default value: utf8. ASCII is supported.

Parameter	Type	Required	Description
errors	Arbitrary (automatically converted to the string type)	No	The method that is used to process characters if the characters cannot be recognized based on the encoding format. Valid values: ◦ ignore: The characters are not encoded. This is the default value. ◦ strict: An error is reported, and the data of the log is discarded. ◦ replace: The characters are replaced with question marks (?). ◦ xmlcharrefreplace: The characters are replaced with the related XML characters.

- Response

  A decoded value is returned.

- Examples

  Raw log:

  ```
  test: asewds
  ```

  Transformation rule:

  ```
  e_set("encoding", str_decode(b'\xe4\xbd\xa0\xe5\xa5\xbd', "utf8",'strict'))
  ```

  Result:

  ```
  test: asewds
  encoding: hello
  ```

str_replace

- Syntax

  ```
  str_replace(value, old, new, count)
  ```

> **Note** You can call this function by passing basic variable parameters. For more information, see Function invoking.

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The value in which you want to replace a string.
old	Arbitrary (automatically converted to the string type)	Yes	The string that you want to replace.
new	Arbitrary (automatically converted to the string type)	Yes	The string that you want to use to replace the specified string.
count	Number	No	The number of replacements. If you do not configure this parameter, the specified string in all occurrences in the value is replaced.

- Response

  A new string is returned.

- Examples: Convert a dictionary to the JSON format.

  Raw log:

```
content:  {'referer': '-', 'request': 'GET /phpMyAdmin', 'status': 404, 'data-1': {'aaa': 'Mozilla', 'bbb': 'asde'}, 'data-2': {'up_adde': '-', 'up_host': '-'}}
```

  Transformation rule:

```
e_set("content_json", str_replace(ct_str(v("content")),"'",'"'))
```

  Result:

```
content:  {'referer': '-', 'request': 'GET /phpMyAdmin', 'status': 404, 'data-1': {'aaa': 'Mozilla', 'bbb': 'asde'}, 'data-2': {'up_adde': '-', 'up_host': '-'}}
content_json:  {"referer": "-", "request": "GET /phpMyAdmin", "status": 404, "data-1": {"aaa": "Mozilla", "bbb": "asde"}, "data-2": {"up_adde": "-", "up_host": "-"}}
```

str_sort

- Syntax

```
str_sort(value, reverse)
```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The string that you want to sort.
reverse	Boolean	No	Default value: False. This value indicates that the string is sorted in ascending order.

- Response

  A sorted string is returned.

- Examples
  - Example 1: Sort the str field value in alphabetical order.

    Raw log:

    ```
    str: twish
    ```

    Transformation rule:

    ```
    e_set("str_sort", str_sort(v("str")))
    ```

    Result:

    ```
    str: twish
    str_sort: histw
    ```

  - Example 2: Sort the str field value in reverse alphabetical order at a granularity of two-letter pairs.

    Raw log:

    ```
    str: twish
    ```

    Transformation rule:

    ```
    e_set("str_sort", str_sort(v("str"),True))
    ```

    Result:

    ```
    str: twish
    str_sort: wtsih
    ```

str_reverse

- Syntax

  ```
  str_reverse(value)
  ```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The value that you want to reverse.

- Response

  A reversed string is returned.

- Examples: Reverse the data field value.

  Raw log:

  ```
  data:twish
  ```

  Transformation rule:

  ```
  e_set("reverse_data", str_reverse(v("data")))
  ```

  Result:

  ```
  data:twish
  reverse_data:hsiwt
  ```

str_logstash_config_normalize

- Syntax

  ```
  str_logstash_config_normalize(value)
  ```

  > **Note** For more information about the Logstash configuration language, see Logstash.

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The value that you want to convert.

- Response

  A converted string is returned.

- Examples: Convert the logstash field value.

  Raw log:

  ```
  logstash: {"name"=>"tw5"}
  ```

  Transformation rule:

  ```
  e_set("normalize_data", str_logstash_config_normalize(v("logstash")))
  ```

  Result:

```
logstash: {"name"=>"tw5"}
normalize_data:{"name":"tw5"}
```

str_hex_escape_encode

- Syntax

```
str_hex_escape_encode(value)
```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The value that you want to escape.

- Response

  An escaped string is returned.

str_strip

- Syntax

```
str_strip(value, chars)
```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The original string that you want to modify.
chars	Arbitrary (automatically converted to the string type)	No	The character set that you want to delete from the start and end of the specified string. Default value: `\t\r\n`.

- Response

  A modified string is returned.

- Examples

- Example 1: Delete the asterisks (*) from the start of the strip field value.

  Raw log

  ```
  strip: ***I love Etl
  ```

  Transformation rule

  ```
  e_set("str_strip", str_strip(v("strip"),"*"))
  ```

  Result

  ```
  strip: ***I love Etl
  str_strip:I love Etl
  ```

- Example 2: Delete the spaces from the start of the strip field value.

  Raw log

  ```
  strip:    I love Etl
  ```

  Transformation rule

  ```
  e_set("str_strip", str_strip(v("strip")))
  ```

  Result

  ```
  strip:    I love Etl
  str_strip: I love Etl
  ```

- Example 3: Delete the character set `xy`.

  Raw log

  ```
  strip:xy123yx
  ```

  Transformation rule

  ```
  e_set("str_strip", str_strip(v("strip"),"xy"))
  ```

  Result

  ```
  strip:xy123yx
  str_strip:123
  ```

str_lower

- Syntax

  ```
  str_lower(value)
  ```

- Parameters

Parameter	Type	Required	Description

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The string that you want to convert.

- Response

  A converted string is returned.

- Examples: Convert the name field value to lowercase letters.

  Raw log:

  ```
  name: Etl
  ```

  Transformation rule:

  ```
  e_set("str_lower", str_lower(v("name")))
  ```

  Result:

  ```
  name: Etl
  str_lower: etl
  ```

str_upper

- Syntax

  ```
  str_upper(value)
  ```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The string that you want to convert.

- Response

  A converted string is returned.

- Examples: Convert the name field value to uppercase letters.

  Raw log:

  ```
  name: etl
  ```

  Transformation rule:

  ```
  e_set("str_upper", str_upper(v("name")))
  ```

  Result:

  ```
  name: etl
  str_upper: ETL
  ```

str_title

- Syntax

```
str_title(value)
```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The string that you want to convert.

- Response

  A converted string is returned.

- Examples: Capitalize the first letter of each word in the word field value.

  Raw log:

```
word: this is etl
```

  Transformation rule:

```
e_set("str_title", str_title(v("word")))
```

  Result:

```
word: this is etl
str_title: This Is Etl
```

str_capitalize

- Syntax

```
str_capitalize(value)
```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The string that you want to convert.

- Response

  A converted string is returned.

- Examples: Capitalize the first letter of the word field value and convert the other letters in the value to lowercase letters.

  Raw log:

```
word: this Is MY EAL
```

Transformation rule:

```
e_set("str_capitalize", str_capitalize(v("word")))
```

Result:

```
word: this Is MY EAL
str_capitalize: This is my eal
```

str_lstrip

- Syntax

```
str_lstrip(value, chars)
```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The original string that you want to modify.
chars	Arbitrary (automatically converted to the string type)	No	The character set that you want to delete from the start of the string. Default value: space.

- Response

  A modified string is returned.

- Examples
  - Example 1: Delete the asterisks (*) from the start of the word field value.

    Raw log:

    ```
    word: ***this is string
    ```

    Transformation rule:

    ```
    e_set("str_lstrip", str_lstrip(v("word"),"*"))
    ```

    Result:

    ```
    word: ***this is string
    str_lstrip: this is string
    ```

- Example 2: Delete the spaces from the start of the word field value.

  Raw log:

  ```
  word:     this is string
  ```

  Transformation rule:

  ```
  e_set("str_lstrip", str_lstrip(v("word")))
  ```

  Result:

  ```
  word:     this is string
  str_lstrip: this is string
  ```

- Example 3: Delete the character set `xy`.

  Raw log:

  ```
  lstrip:xy123yx
  ```

  Transformation rule:

  ```
  e_set("str_lstrip", str_lstrip(v("lstrip"),"xy"))
  ```

  Result:

  ```
  lstrip:xy123yx
  str_lstrip:123yx
  ```

str_rstrip

- Syntax

  ```
  str_rstrip(value, chars)
  ```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The original string that you want to modify.
chars	Arbitrary (automatically converted to the string type)	No	The character set that you want to delete from the end of the string. Default value: space.

- Response

  A modified string is returned.

- Examples

- Example 1: Delete the asterisks (*) from the end of the word field value.

  Raw log:

  ```
  word: this is string*****
  ```

  Transformation rule:

  ```
  e_set("str_rstrip", str_rstrip(v("word"),"*"))
  ```

  Result:

  ```
  word: this is string*****
  str_rstrip: this is string
  ```

- Example 2: Delete the character set `xy`.

  Raw log:

  ```
  word:xy123yx
  ```

  Transformation rule:

  ```
  e_set("str_rstrip", str_rstrip(v("word"),"xy"))
  ```

  Result:

  ```
  word:xy123yx
  str_rstrip:xy123
  ```

str_swapcase

- Syntax

  ```
  str_swapcase(value)
  ```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The string that you want to convert.

- Response

  A converted string is returned.

- Examples

  Raw log:

  ```
  name: this is string
  ```

  Transformation rule:

  ```
  e_set("str_swapcase", str_swapcase(v("name")))
  ```

Result:

```
name: this is string
str_swapcase: THIS IS STRING
```

str_translate

- Syntax

```
str_translate(value, original character set, mapping character set)
```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The original string in which you want to replace characters.
original character set	Arbitrary (automatically converted to the string type)	Yes	The original character set.
mapping character set	Arbitrary (automatically converted to the string type)	Yes	The character set that is used to replace characters in the original character set.

- Response

  A processed string is returned.

- Examples

  Raw log:

```
name: I love ETL!!!
```

  Transformation rule:

```
e_set("str_translate", str_translate(v("name"),"aeiou","12345"))
```

  Result:

```
name: I love ETL!!!
str_translate: I l4v2 ETL!!!
```

str_endswith

- Syntax

```
str_endswith(value, suffix, start, end)
```

> **Note** You can call this function by passing basic variable parameters. For more information, see Function invoking.

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The original string that you want to check.
suffix	Arbitrary (automatically converted to the string type)	Yes	The suffix. The value of this parameter can be a string or an element.
start	Number	No	The position from which the string suffix check starts. The value 0 indicates the first character. The value -1 indicates the last character.
end	Number	No	The position at which the string suffix check ends. The value 0 indicates the first character. The value -1 indicates the last character.

- Response

  If the string ends with the specified suffix, the value True is returned. Otherwise, the value False is returned.

- Examples

  Raw log:

```
name: this is endswith!!!
```

  Transformation rule:

```
e_set("str_endswith",str_endswith(v("name"),"!"))
```

  Result:

```
name: this is endswith!!!
str_endswith: True
```

str_startswith

- Syntax

```
str_startswith(value, prefix, start, end)
```

> **Note** You can call this function by passing basic variable parameters. For more information, see Function invoking.

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The original string that you want to check.
prefix	Arbitrary (automatically converted to the string type)	Yes	The prefix. The value of this parameter can be a string or an element.
start	Number	No	The position from which the string prefix check starts. The value 0 indicates the first character. The value -1 indicates the last character.
end	Number	No	The position at which the string prefix check ends. The value 0 indicates the first character. The value -1 indicates the last character.

- Response

  If the string starts with the specified prefix, the value True is returned. Otherwise, the value False is returned.

- Examples

  Raw log:

```
name: !! this is startwith
```

  Transformation rule:

```
e_set("str_startswith",str_startswith(v("name"),"!!"))
```

  Result:

```
name: !! this is startwith
str_startswith: True
```

str_find

- Syntax

```
str_find(value, str, begin, end)
```

> **Note** You can call this function by passing basic variable parameters. For more information, see Function invoking.

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The original string in which you want to search for a substring.
str	Arbitrary (automatically converted to the string type)	Yes	The substring that you want to search for.
begin	Number	No	The position from which the substring search starts. Default value: 0. The value 0 indicates the first character. The value -1 indicates the last character.
end	Number	No	The position at which the substring search ends. The default value is the length of the string. The value 0 indicates the first character. The value -1 indicates the last character.

- Response

  The position of the specified substring in the original string is returned. If the specified substring appears multiple times in the original string, only the position of the first occurrence of the substring is returned.

- Examples

  Raw log:

```
name: hello world
```

  Transformation rule:

```
e_set("str_find",str_find(v("name"),"h"))
```

Result:

```
name: hello world
str_find: 0
```

str_count

- Syntax

```
str_count(value, sub, start, end)
```

> **Note** You can call this function by passing basic variable parameters. For more information, see Function invoking.

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The original string in which you want to count the number of occurrences of a character.
sub	Arbitrary (automatically converted to the string type)	Yes	The character whose number of occurrences you want to count.
start	Number	No	The position from which the search for the specified character starts in the string. Valid values: ◦ 0: the first character. This is the default value. ◦ -1: the last character.
end	Number	No	The position at which the search for the specified character ends in the string. Valid values: ◦ 0: the first character. ◦ -1: the last character. This is the default value.

- Response

  The number of occurrences of the specified character is returned.

- Examples

  Raw log:

  ```
  name: this is really a string
  ```

  Transformation rule:

  ```
  e_set("str_count",str_count(v("name"),"i"))
  ```

  Result:

  ```
  name: this is really a string
  str_count: 3
  ```

str_rfind

- Syntax

  ```
  str_rfind(value, substr, beg, end)
  ```

  > **Note** You can call this function by passing basic variable parameters. For more information, see Function invoking.

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The original string in which you want to search for a character.
substr	Arbitrary (automatically converted to the string type)	Yes	The character that you want to search for.
beg	Number	No	The position from which the search starts. Default value: 0.
end	Number	No	The position at which the search ends. The default value is the length of the string.

- Response

  The position of the last occurrence of the specified character is returned.

- Examples

  Raw log:

  ```
  name: this is really a string
  ```

  Transformation rule:

```
e_set("str_rfind",str_rfind(v("name"),"i"))
```

Result:

```
name: this is really a string
str_rfind: 20
```

str_split

- Syntax

```
str_split(value, sep=None, maxsplit=-1)
```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The original string that you want to split.
sep	Number	No	The delimiter. The value None indicates a space.
maxsplit	Number	No	The maximum number of strings into which you can split the original string. The value -1 indicates no limit.

- Response

  A processed string is returned.

- Examples: Split the content field value by using the space delimiter.

  Raw log:

```
content: hello world
```

  Transformation rule:

```
e_set("str_split",str_split(v("content")," "))
```

  Result:

```
content: hello world
str_split: ["hello", "world"]
```

str_splitlines

- Syntax

```
str_splitlines(value, keepends)
```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The original string that you want to split.
keepends	Boolean	No	Specifies whether to delete line feeds from the output result. The line feeds include `\r`, `\r\n`, and `\n`. Default value: *False*. This value indicates that line feeds are deleted. The value *True* indicates that line feeds are retained.

- Response

  Processed strings are returned.

- Examples
  - Example 1

    Raw log:

    ```
    content: ab c\n\nde fg\rkl\r\n
    ```

    Transformation rule:

    ```
    e_set("str_splitlines",str_splitlines(v("content"),False))
    ```

    Result:

    ```
    content: ab c\n\nde fg\rkl\r\n
    str_splitlines: ['ab c', '', 'de fg', 'kl']
    ```

  - Example 2

    Raw log:

    ```
    content: ab c\n\nde fg\rkl\r\n
    ```

    Transformation rule:

    ```
    e_set("str_splitlines",str_splitlines(v("content"),True))
    ```

    Result:

    ```
    content: ab c\n\nde fg\rkl\r\n
    str_splitlines: ['ab c\n', '\n', 'de fg\r', 'kl\r\n']
    ```

str_partition

- Syntax

```
str_partition(value, substr)
```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The string that you want to split.
substr	Arbitrary (automatically converted to the string type)	No	The delimiter.

- Response

  Split strings are returned.

- Examples: Split the website field value into three parts from left to right by using the `.` delimiter.

  Raw log:

```
website: www.aliyun.com
```

  Transformation rule:

```
e_set("str_partition",str_partition(v("website"),"."))
```

  Result:

```
website: www.aliyun.com
str_partition:  ["www", ".", "aliyun.com"]
```

str_rpartition

- Syntax

```
str_rpartition(value, substr)
```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The original string that you want to split.
substr	Arbitrary (automatically converted to the string type)	No	The delimiter.

- Response

  Processed strings are returned.

- Examples: Split the website field value into three parts from right to left by using the `.` delimiter.

  Raw log:

  ```
  website: www.aliyun.com
  ```

  Transformation rule:

  ```
  e_set("str_partition",str_rpartition(v("website"),"."))
  ```

  Result:

  ```
  website: www.aliyun.com
  str_partition: ["www.aliyun", ".", "com"]
  ```

str_center

- Syntax

  ```
  str_center(value, width, fillchar)
  ```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The original string that you want to modify.
width	Number	Yes	The length of the string after padding.
fillchar	Arbitrary (automatically converted to the string type)	No	The character that is used for padding. Default value: space.

- Response

  A processed string is returned.

  > **Note** If the length of the string after padding is less than the length of the original string, the original string is returned.

- Examples

- Example 1: Pad a string by using asterisks (*).

  Raw log:

  ```
  center: this is center
  ```

  Transformation rule:

  ```
  e_set("str_center", str_center(v("center"), 40, "*"))
  ```

  Result:

  ```
  center: this is center
  str_center: *************this is center*************
  ```

- Example 2: Pad a string by using spaces.

  Raw log:

  ```
  center: this is center
  ```

  Transformation rule:

  ```
  e_set("str_center",str_center(v("center"),40))
  ```

  Result:

  ```
  center: this is center
  str_center:              this is center
  ```

str_zfill

- Syntax

  ```
  str_zfill(value, width)
  ```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The original string that you want to modify.
width	Number	Yes	The length of the string after padding.

- Response

  A processed string is returned.

- Examples

  Raw log:

  ```
  center: this is zfill
  ```

  Transformation rule:

```
e_set("str_zfill",str_zfill(v("center"),40))
```

Result:

```
center:this is zfill
str_zfill:000000000000000000000000000this is zfill
```

str_expandtabs

- Syntax

```
str_expandtabs(value, tabsize)
```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The original string that you want to modify.
tabsize	Number	Yes	The number of spaces after conversion.

- Response

  A processed string is returned.

- Examples
  - Example 1: Convert `\t` in the logstash field value to a space.

    Raw log:

    ```
    logstash: this is\tstring
    ```

    Transformation rule:

    ```
    e_set("str_expandtabs",str_expandtabs(v("logstash")))
    ```

    Result:

    ```
    logstash: this is\tstring
    str_expandtabs: this is string
    ```

- Example 2: Convert `\t` in the center field value to spaces.

  Raw log:

  ```
  {"center": "this is\tstring"}
  ```

  Transformation rule:

  ```
  e_set("str_expandtabs",str_expandtabs(v("center"),16))
  ```

  Result:

  ```
  center: this is\tstring
  str_expandtabs: this is         string
  ```

str_ljust

- Syntax

  ```
  str_ljust(value, width, fillchar)
  ```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The original string that you want to modify.
width	Number	Yes	The length of the string after padding.
fillchar	Arbitrary (automatically converted to the string type)	No	The character that is used for padding. Default value: space.

- Response

  A processed string is returned.

  > **Note** If the length of the string after padding is less than the length of the original string, the original string is returned.

- Examples

- Example 1

  Raw log:

  ```
  content: this is ljust
  ```

  Transformation rule:

  ```
  e_set("str_ljust",str_ljust(v("content"),20,"*"))
  ```

  Result:

  ```
  content: this is ljust
  str_ljust: this is ljust*******
  ```

- Example 2

  Raw log:

  ```
  center: this is ljust
  ```

  Transformation rule:

  ```
  e_set("str_ljust",str_ljust(v("center"),20,))
  ```

  Result:

  ```
  center: this is ljust
  str_ljust: this is ljust
  ```

- Example 3: The value of width is less than the length of the original string. The original string is returned.

  Raw log:

  ```
  center: this is ljust
  ```

  Transformation rule:

  ```
  e_set("str_ljust",str_ljust(v("center"),10,"*"))
  ```

  Result:

  ```
  center: this is ljust
  str_ljust: this is ljust
  ```

str_rjust

- Syntax

  ```
  str_rjust(value, width, fillchar)
  ```

- Parameters

Parameter	Type	Required	Description

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The original string that you want to modify.
width	Number	Yes	The length of the string after padding.
fillchar	Arbitrary (automatically converted to the string type)	No	The character that is used for padding. Default value: space.

- Response

  A processed string is returned.

  > **Note** If the length of the string after padding is less than the length of the original string, the original string is returned.

- Examples

  Raw log:

  ```
  center: this is rjust
  ```

  Transformation rule:

  ```
  e_set("str_rjust",str_rjust(v("center"),20,"*"))
  ```

  Result:

  ```
  center: this is rjust
  str_rjust: *******this is rjust
  ```

str_zip

- Syntax

  ```
  str_zip(value 1, value 2, combine_sep=None, sep=None, quote=None, lparse=None, rparse=None)
  ```

- Parameters

Parameter	Type	Required	Description
value 1	Arbitrary (automatically converted to the string type)	Yes	The value that you want to combine.
value 2	Arbitrary (automatically converted to the string type)	Yes	The value that you want to combine.

Parameter	Type	Required	Description
combine_sep	Arbitrary (automatically converted to the string type)	No	The identifier that is used when elements are combined. Default value: `#`.
sep	Arbitrary (automatically converted to the string type)	No	The delimiter that is used between the elements after combining. The value must be a single character. Default value: `,`.
quote	Arbitrary (automatically converted to the string type)	No	The character is used to enclose the elements after combining. This parameter is required if the values contain delimiters. Default value: `"`.
lparse	Arbitrary (automatically converted to the string type)	No	The delimiter and quote that are used among the elements of `value 1`. Default delimiter: `,`. Default quote: `"`. Format: `lparse=(',', '"')`. **Note** The quote has a higher priority than the delimiter.
rparse	Arbitrary (automatically converted to the string type)	No	The delimiter and quote that are used among the elements of `value 2`. Default delimiter `,`. Default quote: `"`. Format: `rparse=(',', '"')`. **Note** The quote has a higher priority than the delimiter.

- Response

  A combined string is returned.

- Examples

- Example 1

  Raw log:

  ```
  website: wwww.aliyun.com
  escape_name: o
  ```

  Transformation rule:

  ```
  e_set("combine",str_zip(v("website"),v("escape_name"), combine_sep="@"))
  ```

  Result:

  ```
  website: wwww.aliyun.com
  escape_name: o
  combine: wwww.aliyun.com@o
  ```

- Example 2

  Raw log:

  ```
  website: wwww.aliyun.com
  escape_name: o
  ```

  Transformation rule:

  ```
  e_set("combine",str_zip(v("website"),v("escape_name")))
  ```

  Result:

  ```
  website: wwww.aliyun.com
  escape_name: o
  wwww.aliyun.com#o
  ```

- Example 3: In this example, the sep parameter is used.

  Raw log:

  ```
  f1: a,b,c
  f2: x,y,z
  ```

  Transformation rule:

  ```
  e_set("combine",str_zip(v("f1"), v("f2"), sep="|"))
  ```

  Result:

  ```
  f1: a,b,c
  f2: x,y,z
  combine: a#x|b#y|c#z
  ```

- Example 4: In this example, the quote parameter is used.

  Raw log:

  ```
  f1: "a,a", b, "c,c"
  f2: x, "y,y", z
  ```

  Transformation rule:

  ```
  e_set("combine",str_zip(v("f1"), v("f2"), quote='|'))
  ```

  Result:

  ```
  f1: "a,a", b, "c,c"
  f2: x, "y,y", z
  combine: |a,a#x|,|b#y,y|,|c,c#z|
  ```

- Example 5: In this example, field values with different lengths are used.

  Raw log:

  ```
  f1: a,b
  f2: x,y,z
  ```

  Transformation rule:

  ```
  e_set("combine",str_zip(v("f1"), v("f2")))
  ```

  Result:

  ```
  f1: a,b
  f2: x,y,z
  combine: a#x,b#y
  ```

- Example 6: In this example, the lparse and rparse parameters are used.

  Raw log:

  ```
  f1: a#b#c
  f2: x|y|z
  ```

  Transformation rule:

  ```
  e_set("combine",str_zip(v("f1"), v("f2"), lparse=("#", '"'), rparse=("|", '"')))
  ```

  Result:

  ```
  f1: a#b#c
  f2: x|y|z
  combine: a#x,b#y,c#z
  ```

- Example 7: In this example, the lparse and rparse parameters are used.

  Raw log:

  ```
  f1: |a,a|, b, |c,c|
  f2: x, #y,y#, z
  ```

  Transformation rule:

  ```
  e_set("combine",str_zip(v("f1"), v("f2"), lparse=(",", '|'), rparse=(",", '#')))
  ```

  Result:

  ```
  f1: |a,a|, b, |c,c|
  f2: x, #y,y#, z
  combine: "a,a#x","b#y,y","c,c#z"
  ```

str_isalnum

- Syntax

  ```
  str_isalnum(value)
  ```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The string that you want to check.

- Response

  The value True or False is returned.

- Examples

  Raw log:

  ```
  content: 13
  ```

  Transformation rule:

  ```
  e_set("str_isalnum",str_isalnum(v("content")))
  ```

  Result:

  ```
  content: 13
  str_isalnum: True
  ```

str_isalpha

- Syntax

  ```
  str_isalpha(value)
  ```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The string that you want to check.

- Response

  The value True or False is returned.

- Examples

  Raw log:

  ```
  content: 13
  ```

  Transformation rule:

  ```
  e_set("str_isalpha",str_isalpha(v("content")))
  ```

  Result:

  ```
  content: 13
  str_isalpha: False
  ```

str_isascii

- Syntax

  ```
  str_isascii(value)
  ```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The string that you want to check.

- Response

  The value True or False is returned.

- Examples

  Raw log:

  ```
  content: asw123
  ```

  Transformation rule:

  ```
  e_set("str_isascii",str_isascii(v("content")))
  ```

  Result:

  ```
  content: asw123
  str_isascii: True
  ```

str_isdecimal

- Syntax

```
str_isdecimal(value)
```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The string that you want to check.

- Response

  The value True or False is returned.

- Examples
  - Example 1

    Raw log:

    ```
    welcome: hello
    ```

    Transformation rule:

    ```
    e_set("str_isdecimal",str_isdecimal(v("welcome")))
    ```

    Result:

    ```
    welcome: hello
    str_isdecimal: False
    ```

  - Example 2

    Raw log:

    ```
    num: 123
    ```

    Transformation rule:

    ```
    e_set("str_isdecimal",str_isdecimal(v("num")))
    ```

    Result:

    ```
    num: 123
    str_isdecimal: True
    ```

str_isdigit

- Syntax

```
str_isdigit(value)
```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The string that you want to check.

- Response

  The value True or False is returned.

- Examples

  Raw log:

  ```
  content: 13
  ```

  Transformation rule:

  ```
  e_set("str_isdigit",str_isdigit(v("content")))
  ```

  Result:

  ```
  content: 13
  str_isdigit: True
  ```

str_isidentifier

- Syntax

  ```
  str_isidentifier(value)
  ```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The string that you want to check.

- Response

  The value True or False is returned.

- Examples

  Raw log:

  ```
  class: class
  ```

  Transformation rule:

  ```
  e_set("str_isidentifier",str_isidentifier(v("class")))
  ```

  Result:

  ```
  class: class
  str_isidentifier: True
  ```

str_islower

- Syntax

```
str_islower(value)
```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The string that you want to check.

- Response

  The value True or False is returned.

- Examples

  Raw log:

```
lower: asds
```

  Transformation rule:

```
e_set("str_islower",str_islower(v("lower")))
```

  Result:

```
lower: asds
str_islower: True
```

str_isnumeric

- Syntax

```
str_isnumeric(value)
```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The string that you want to check.

- Response

  The value True or False is returned.

- Examples

  Raw log:

```
num: 123
```

  Transformation rule:

```
e_set("str_isnumeric",str_isnumeric(v("num")))
```

Result:

```
num: 123
str_isnumeric: True
```

str_isprintable

- Syntax

```
str_isprintable(value)
```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The string that you want to check.

- Response

  The value True or False is returned.

- Examples

  Raw log:

```
content: vs;,.as
```

  Transformation rule:

```
e_set("str_isprintable",str_isprintable(v("content")))
```

  Result:

```
content: vs;,.as
str_isprintable: True
```

str_isspace

- Syntax

```
str_isspace(value)
```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The string that you want to check.

- Response

  The value True or False is returned.

- Examples

  Raw log:

  ```
  space:  as afsd
  ```

  Transformation rule:

  ```
  e_set("str_isspace",str_isspace(v("space")))
  ```

  Result:

  ```
  space:  as afsd
  str_isspace: False
  ```

str_istitle

- Syntax

  ```
  str_istitle(value)
  ```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The string that you want to check.

- Response

  The value True or False is returned.

- Examples

  Raw log:

  ```
  title: Alex
  ```

  Transformation rule:

  ```
  e_set("str_istitle",str_istitle(v("title")))
  ```

  Result:

  ```
  title: Alex
  str_istitle: True
  ```

str_isupper

- Syntax

  ```
  str_isupper(value)
  ```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The string that you want to check.

- Response

  The value True or False is returned.

- Examples

  Raw log:

  ```
  content: ASSD
  ```

  Transformation rule:

  ```
  e_set("str_isupper",str_isupper(v("content")))
  ```

  Result:

  ```
  content: ASSD
  str_isupper: True
  ```

str_uuid

- Syntax

  ```
  str_uuid(lower=True)
  ```

- Parameters

Parameter	Type	Required	Description
lower	Boolean	No	Specifies whether the letters in the output UUID are in lowercase. Default value: True. This value indicates that the letters are in lowercase.

- Response

  A UUID is returned.

- Examples

- Example 1

  Raw log:

  ```
  content: I am Iron man
  ```

  Transformation rule:

  ```
  e_set("UUID",str_uuid())
  ```

  Result:

  ```
  content: I am Iron man
  UUID: dcc54d7e-c95f-11e9-9791-ffd6754a4f32
  ```

- Example 2

  Raw log:

  ```
  content: I am Iron man
  ```

  Transformation rule:

  ```
  e_set("UUID",str_uuid(lower=False))
  ```

  Result:

  ```
  content: I am Iron man
  UUID: DCC72D74-C95F-11E9-9921-F395B8647FFA
  ```

12.6.7. Date and time functions

This topic describes the syntax and parameters of date and time functions. This topic also provides examples on how to use data and time functions.

Date and time functions support the following data types: datetime object, string, and UNIX timestamp.

> **Note** A UNIX timestamp is a string.

All values in log events are stored as strings based on the transformation logic of Log Service domain-specific language (DSL). You must convert data types based on your business requirements.

Among the date and time functions that are described in this topic, only the `dt_parse`, `dt_str`, and `dt_parsetimestamp` functions support parameter values of all the three types. For each of other date and time functions, you must make sure that the parameter values are of the same type.

Functions

Scenario	Function	Description
	dt_parse	Converts the value of a time expression to a datetime object.
	dt_str	Converts the value of a time expression to a string.

Scenario	Function	Description
Convert common datetime	dt_parsetimestamp	Converts the value of a time expression to a UNIX timestamp.
	dt_prop	Obtains a specific attribute from the value of a time expression.
Obtains datetime	dt_now	Obtains the current datetime.
	dt_today	Obtains the current date without time.
	dt_utcnow	Obtains the current datetime in the current time zone.
	dt_fromtimestamp	Converts a UNIX timestamp to a datetime object.
	dt_utcfromtimestamp	Converts a UNIX timestamp to a datetime object in the current time zone.
	dt_strptime	Parses a time string into a datetime object.
Obtains a UNIX timestamp	dt_currentstamp	Obtains a UNIX timestamp.
	dt_totimestamp	Converts a datetime object to a UNIX timestamp.
Obtains a datetime string	dt_strftime	Converts a datetime object to a string in the specified format.
	dt_strftimestamp	Converts a UNIX timestamp to a string in the specified format.
	dt_truncate	Truncates the value of a time expression based on the specified precision.
	dt_add	Modifies the value of a time expression based on the specified precision.
	dt_MO	Offsets the specified time to the date of the previous or next Nth Monday. The offset value N is passed to the `weekday` parameter of the `dt_add` function. N can be a positive integer or a negative integer. To pass in a negative integer, use `op_neg(positive integer)`.
	dt_TU	Offsets the specified time to the date of the previous or next Nth Tuesday. The offset value N is passed to the `weekday` parameter of the `dt_add` function. N can be a positive integer or a negative integer. To pass in a negative integer, use `op_neg(positive integer)`.

Scenario	Function	Description
Modify datetime	dt_WE	Offsets the specified time to the date of the previous or next Nth Wednesday. The offset value N is passed to the `weekday` parameter of the `dt_add` function. N can be a positive integer or a negative integer. To pass in a negative integer, use `op_neg(positive integer)`.
	dt_TH	Offsets the specified time to the date of the previous or next Nth Thursday. The offset value N is passed to the `weekday` parameter of the `dt_add` function. N can be a positive integer or a negative integer. To pass in a negative integer, use `op_neg(positive integer)`.
	dt_FR	Offsets the specified time to the date of the previous or next Nth Friday. The offset value N is passed to the `weekday` parameter of the `dt_add` function. N can be a positive integer or a negative integer. To pass in a negative integer, use `op_neg(positive integer)`.
	dt_SA	Offsets the specified time to the date of the previous or next Nth Saturday. The offset value N is passed to the `weekday` parameter of the `dt_add` function. N can be a positive integer or a negative integer. To pass in a negative integer, use `op_neg(positive integer)`.
	dt_SU	Offsets the specified time to the date of the previous or next Nth Sunday. The offset value N is passed to the `weekday` parameter of the `dt_add` function. N can be a positive integer or a negative integer. To pass in a negative integer, use `op_neg(positive integer)`.
Modify a time zone	dt_astimezone	Converts the value of a time expression to a datetime object in the specified time zone.
Obtains the time difference	dt_diff	Obtains the difference between values or between the values of time expressions based on the specified precision.

dt_parse

- Syntax

```
dt_parse(value, tz=None)
```

- Parameters

Parameter	Type	Required	Description
value	String, UNIX timestamp, or datetime object	Yes	The value that you want to convert.
tz	String	No	The time zone. Default value: None. For more information, see Time zone list.

- Returned result

  The converted datetime object.

- Examples
  - Example 1: Convert a timestamp to a datetime object.

    Raw log entry:

    ```
    time: 1559500886
    ```

    Transformation rule:

    ```
    e_set("test_time", dt_parse(v("time")))
    ```

    Result:

    ```
    time: 1559500886
    test_time: 2019-06-03 18:41:26
    ```

  - Example 2: Convert a timestamp to a datetime object in the time zone of Shanghai.

    Raw log entry:

    ```
    time: 2019-06-01 10:10:10
    tz: Asia/Shanghai
    ```

    Transformation rule:

    ```
    e_set("test_time", dt_parse(v("time"),tz=v("tz")))
    ```

    Result:

    ```
    time: 2019-06-01 10:10:10
    tz: Asia/Shanghai
    test_time: 2019-06-01 10:10:10+08:00
    ```

dt_str

- Syntax

  ```
  dt_str(value, fmt="format", tz=None)
  ```

- Parameters

Parameter	Type	Required	Description
value	String, UNIX timestamp, or datetime object	Yes	The time expression.
fmt	String	No	The format. For more information, see Date and time formatting directives. If you do not set this parameter, the default format is used.
tz	String	No	The time zone. Default value: None. For more information, see Time zone list.

- Returned result

  The converted time string.

- Examples
  - Example 1: Convert a point in time to a time string in the specified format in the time zone of Tokyo.

    Raw log entry:

    ```
    time: 2019-06-03 02:41:26
    fmt: %Y/%m/%d %H-%M-%S
    tz: Asia/Tokyo
    ```

    Transformation rule:

    ```
    e_set("dt_str", dt_str(v("time"),fmt=v("fmt"),tz=v("tz")))
    ```

    Result:

    ```
    time: 2019-06-03 02:41:26
    fmt: %Y/%m/%d %H-%M-%S
    tz: Asia/Tokyo
    dt_str: 2019/06/03 02-41-26
    ```

  - Example 2: Convert a timestamp to a time string in the specified format.

    Raw log entry:

    ```
    time: 1559500886
    fmt: %Y/%m/%d %H-%M-%S
    ```

    Transformation rule:

    ```
    e_set("dt_str", dt_str(v("time"),fmt=v("fmt")))
    ```

    Result:

    ```
    time: 1559500886
    fmt: %Y/%m/%d %H-%M-%S
    dt_str: 2019/06/03 02-41-26
    ```

- Example 3: Convert a point in time to a time string in the default format.

  Raw log entry:

  ```
  time: 2019-06-03 02:41:26
  ```

  Transformation rule:

  ```
  e_set("dt_str", dt_str(v("time")))
  ```

  Result:

  ```
  time: 2019-06-03 02:41:26
  dt_str: 2019-06-03 02:41:26
  ```

dt_parsetimestamp

- Syntax

  ```
  dt_parsetimestamp(value, tz=None)
  ```

- Parameters

Parameter	Type	Required	Description
value	String, UNIX timestamp, or datetime object	Yes	The time expression.
tz	String	No	The time zone. Default value: None. For more information, see Time zone list.

- Returned result

  The converted UNIX timestamp.

- Examples
  - Example 1: Convert a point in time to a timestamp in the time zone of Tokyo.

    Raw log entry:

    ```
    time: 2019-06-03 2:41:26
    tz: Asia/Tokyo
    ```

    Transformation rule:

    ```
    e_set("dt_parsetimestamp", dt_parsetimestamp(v("time"),v("tz")))
    ```

    Result:

    ```
    time: 2019-06-03 2:41:26
    tz: Asia/Tokyo
    dt_parsetimestamp: 1559500886
    ```

- Example 2: Convert a point in time to a timestamp.

  Raw log entry:

  ```
  time: 2019-06-03 2:41:26
  ```

  Transformation rule:

  ```
  e_set("dt_parsetimestamp",dt_parsetimestamp(v("time")))
  ```

  Result:

  ```
  time: 2019-06-03 2:41:26
  dt_parsetimestamp: 1559529686
  ```

- Example 3: Convert a point in time in UTC+8 to a timestamp.

  Raw log entry:

  ```
  time: 2019-06-03 02:41:26+8:00
  ```

  Transformation rule:

  ```
  e_set("dt_parsetimestamp",dt_parsetimestamp(v("time")))
  ```

  Result:

  ```
  time: 2019-06-03 02:41:26+8:00
  dt_parsetimestamp: 1559500886
  ```

dt_prop

- Syntax

  ```
  dt_prop(value, attribute)
  ```

- Parameters

Parameter	Type	Required	Description
value	String, UNIX timestamp, or datetime object	Yes	The time expression.
attribute	String	Yes	The attribute whose value you want to obtain. For example, if the attribute parameter is set to year, only the year is returned. Valid values: `day`, `year`, `month`, `hour`, `second`, `minute`, `microsecond`, `weekday`, `weekdayname`, `weekdayshortname`, `monthname`, `monthshortname`, `dayofyear`, `dayofweek`, `weekofyear`, `weekofyear_mo`, `tzname`, and `weekofmonth`.

- Returned result

  The extracted attribute value.

- Examples
  - Example 1: Extract the value of the day attribute from a time string.

    Raw log entry:

    ```
    time: 2018-10-2 09:11:40
    day: day
    ```

    Transformation rule:

    ```
    e_set("dt_parsetimestamp",dt_prop(dt_parse(v("time")),v("day")))
    ```

    Result:

    ```
    time: 2018-10-2 09:11:40
    day: day
    dt_prop: 2
    ```

  - Example 2: Extract the value of the year attribute from a time string.

    Raw log entry:

    ```
    time: 2018-10-2 09:11:40
    year: year
    ```

    Transformation rule:

    ```
    e_set("dt_parsetimestamp",dt_prop(dt_parse(v("time")),v("year")))
    ```

    Result:

    ```
    time: 2018-10-2 09:11:40
    year: year
    dt_prop: 2018
    ```

  - Example 3: Extract the value of the weekdayname attribute from a time string.

    Raw log entry:

    ```
    time: 2018-10-2 09:11:40
    weekdayname: weekdayname
    ```

    Transformation rule:

    ```
    e_set("dt_prop",dt_prop(dt_parse(v("time")),v("weekdayname")))
    ```

    Result:

    ```
    time: 2018-10-2 09:11:40
    weekdayname: weekdayname
    dt_prop: Monday
    ```

- Example 4: Extract the value of the weekofyear attribute from a time string.

  Raw log entry:

  ```
  time: 2018-10-2 09:11:40
  weekofyear: weekofyear
  ```

  Transformation rule:

  ```
  e_set("dt_prop",dt_prop(dt_parse(v("time")),v("weekofyear")))
  ```

  Result:

  ```
  time: 2018-10-2 09:11:40
  weekofyear: weekofyear
  dt_prop: 22
  ```

dt_now

- Syntax

  ```
  dt_now(tz=None)
  ```

- Parameters

Parameter	Type	Required	Description
tz	String	No	The time zone. Default value: None. For more information, see Time zone list.

- Returned result

  The datetime object in the specified time zone.

- Example: Obtain the current time in the time zone of Shanghai.

  Raw log entry:

  ```
  tz: Asia/Shanghai
  ```

  Transformation rule:

  ```
  e_set("dt_now",dt_now(tz=v("tz")))
  ```

  Result:

  ```
  tz: Asia/Shanghai
  dt_now: 2019-06-10 11:24:00.655290
  ```

dt_today

- Syntax

  ```
  dt_today(tz=None)
  ```

- Parameters

Parameter	Type	Required	Description
tz	String	No	The time zone. Default value: None. For more information, see Time zone list.

- Returned result

  The date object in the specified time zone.

- Example: Obtain the current date.

  Transformation rule:

  ```
  e_set("dt_today",dt_today())
  ```

  Result:

  ```
  dt_today: 2019-06-10
  ```

dt_utcnow

- Feature

  This function is used to obtain the current datetime in the current time zone.

- Syntax

  ```
  dt_utcnow()
  ```

- Parameters

  None.

- Returned result

  The current datetime in the current time zone.

- Example: Obtain the current datetime in the current time zone.

  Transformation rule:

  ```
  e_set("dt_utcnow",dt_utcnow())
  ```

  Result:

  ```
  dt_utcnow: 2019-06-10 03:32:51.510494
  ```

dt_fromtimestamp

- Syntax

  ```
  dt_fromtimestamp(value, tz=None)
  ```

- Parameters

Parameter	Type	Required	Description
value	String	Yes	The value or expression of the UNIX timestamp that you want to convert.
tz	String	No	The time zone. Default value: None. For more information, see Time zone list.

- Returned result

  The converted datetime.

- Examples
  - Example 1: Convert a timestamp to a datetime.

    Raw log entry:

    ```
    time: 1559500886
    ```

    Transformation rule:

    ```
    e_set("dt_fromtimestamp",dt_fromtimestamp(v("time")))
    ```

    Result:

    ```
    time: 1559500886
    dt_fromtimestamp: 2019-06-03 02:41:26
    ```

  - Example 2: Convert a timestamp to a datetime in the time zone of Shanghai.

    Raw log entry:

    ```
    time: 1559500886
    tz: Asia/Shanghai
    ```

    Transformation rule:

    ```
    e_set("dt_fromtimestamp",dt_fromtimestamp(v("time"),tz=v("tz")))
    ```

    Result:

    ```
    time: 1559500886
    tz: Asia/Shanghai
    dt_fromtimestamp: 2019-06-03 02:41:26+08:00
    ```

dt_utcfromtimestamp

- Feature

  This function is used to convert a UNIX timestamp to datetime in the current time zone.

- Syntax

```
dt_utcfromtimestamp(value)
```

- Parameters

Parameter	Type	Required	Description
value	String	Yes	The value or expression of the UNIX timestamp that you want to convert.

- Returned result

  The converted datetime object.

- Example

  Raw log entry:

  ```
  time: 1559500886
  ```

  Transformation rule:

  ```
  e_set("dt_utcfromtimestamp",dt_utcfromtimestamp(v("time")))
  ```

  Result:

  ```
  time: 1559500886
  dt_utcfromtimestamp: 2019-06-02 18:41:26
  ```

dt_strptime

- Syntax

  ```
  dt_strptime(value, "format")
  ```

- Parameters

Parameter	Type	Required	Description
value	String	Yes	The string that you want to parse.
fmt	String	No	The format. For more information, see Date and time formatting directives.

- Returned result

  The datetime object that is parsed.

- Example

  Raw log entry:

```
time: 2019/06/03 02-41-26
fmt: %Y/%m/%d %H-%M-%S
```

Transformation rule:

```
e_set("dt_strptime",dt_strptime(v("time"),v("fmt")))
```

Result:

```
time: 2019/06/03 02-41-26
fmt: %Y/%m/%d %H-%M-%S
dt_strptime: 2019-06-03 02:41:26
```

dt_currentstamp

- Feature

  This function is used to obtain the current UNIX timestamp.

- Syntax

```
dt_currentstamp(value, normalize='floor')
```

- Parameters

Parameter	Type	Required	Description
value	String	Yes	The string that you want to parse.
normalize	String	No	The numeric format in which you want the function to return the result. Valid values: ○ `floor`: rounds a number down to the nearest integer. Default value: floor. ○ `int`: removes the decimal part of a number. ○ `round`: rounds a number to the nearest integer. ○ `ceil`: rounds a number up to the nearest integer.

- Returned result

  The current UNIX timestamp.

- Example

  Transformation rule:

```
e_set("dt_currentstamp",dt_currentstamp())
```

  Result:

```
dt_currentstamp: 1559500886
```

dt_totimestamp

- Feature

  This function is used to convert a datetime object to a UNIX timestamp.

- Syntax

```
dt_totimestamp(datetime expression)
```

- Parameters

Parameter	Type	Required	Description
Datetime expression	Datetime object	Yes	The datetime object that you want to convert.

- Returned result

  The converted UNIX timestamp.

- Example

  Raw log entry:

```
time: 2019-06-03 2:41:26
```

  Transformation rule:

```
e_set("dt_totimestamp",dt_totimestamp(dt_parse(v("time"))))
```

  Result:

```
time: 2019-06-03 2:41:26
dt_totimestamp: 1559500886
```

dt_strftime

- Syntax

```
dt_strftime(datetime expression, "format")
```

- Parameters

Parameter	Type	Required	Description
Datetime expression	Datetime object	Yes	The datetime object that you want to convert.
format	String	Yes	The format. For more information, see Date and time formatting directives.

- Returned result

The formatted string.

- Example: Convert a time to a string in the specified format.

  Raw log entry:

```
time: 2019-06-03 2:41:26
fmt: %Y/%m/%d %H-%M-%S
```

  Transformation rule:

```
e_set("dt_strftime",dt_strftime(dt_parse(v("time")),v("fmt")))
```

  Result:

```
time: 2019-06-03 2:41:26
fmt: %Y/%m/%d %H-%M-%S
dt_strftime: 2019/06/03 02-41-26
```

dt_strftimestamp

- Syntax

```
dt_strftimestamp(value, fmt="format", tz=None)
```

- Parameters

Parameter	Type	Required	Description
value	String	Yes	The UNIX timestamp that you want to convert.
fmt	String	Yes	The format. For more information, see Date and time formatting directives.
tz	String	No	The time zone. Default value: None. For more information, see Time zone list.

- Returned result

  The formatted string.

- Examples

- Example 1

  Raw log entry:

  ```
  time: 1559500886
  fmt: %Y/%m/%d %H-%M-%S
  ```

  Transformation rule:

  ```
  e_set("dt_strftimestamp",dt_strftimestamp(v("time"),v("fmt")))
  ```

  Result:

  ```
  time: 1559500886
  fmt: %Y/%m/%d %H-%M-%S
  dt_strftimestamp: 2019/06/03 02-41-26
  ```

- Example 2

  Raw log entry:

  ```
  time: 1559500886
  fmt: %Y/%m/%d %H-%M-%S
  tz: Asia/Tokyo
  ```

  Transformation rule:

  ```
  e_set("dt_strftimestamp",dt_strftimestamp(v("time"),v("fmt"),v("tz")))
  ```

  Result:

  ```
  time: 1559500886
  fmt: %Y/%m/%d %H-%M-%S
  tz: Asia/Tokyo
  dt_strftimestamp: 2019/06/03 03-41-26 +0900
  ```

dt_truncate

- Syntax

  ```
  dt_truncate(value, unit='day')
  ```

- Parameters

Parameter	Type	Required	Description
value	String, UNIX timestamp, or datetime object	Yes	The time expression.

Parameter	Type	Required	Description
unit	String	Yes	The precision of the time attribute that you want to obtain. Default value: day. Valid values: `second`, `minute`, `<num>_minute` (for example, 5_minute, 19_minute, and 2_minute), `hour`, `day`, `week`, `month`, `quarter`, `half_year`, and `year`.

- Returned result

  The time that is in the specified precision.

- Examples
  - Example 1

    Raw log entry:

    ```
    time: 2019-06-03 2:41:26
    unit: year
    ```

    Transformation rule:

    ```
    e_set("dt_truncate",dt_truncate(v("time"),v("unit")))
    ```

    Result:

    ```
    time: 2019-06-03 2:41:26
    unit: year
    dt_truncate: 2019-01-01 00:00:00
    ```

  - Example 2

    Raw log entry:

    ```
    time: 2019-06-03 2:41:26
    unit: hour
    ```

    Transformation rule:

    ```
    e_set("dt_truncate",dt_truncate(v("time"),v("unit")))
    ```

    Result:

    ```
    time: 2019-06-03 2:41:26
    unit: hour
    dt_truncate: 2019-06-03 02:00:00
    ```

dt_add

- Syntax

```
dt_add(value, dt1=None, dt2=None, year(s)=None, month(s)=None, day(s)=None, hour(s)=None,
minute(s)=None, second(s)=None, microsecond(s)=None, week(s)=None, weekday=None)
```

- Parameters

Parameter	Type	Required	Description
value	String, UNIX timestamp, or datetime object	Yes	The datetime expression.
dt1	String, UNIX timestamp, or datetime object	No	The datetime expression. Default value: None.
dt2	String, UNIX timestamp, or datetime object	No	The datetime expression. Default value: None.
year/years	Number	No	◦ year: the year that is used to replace the year in the specified time. For example, you can set `year=2020`. Default value: None. ◦ years: the number of years by which you want to offset the specified time. For example, if you set `years=1`, the function increases the year by 1.
day/days	Number	No	◦ day: the day that is used to replace the day in the specified time. For example, you can set `day=1`. Default value: None. ◦ days: the number of days by which you want to offset the specified time. For example, if you set `days=1`, the function increases the day by 1.
hour/hours	Number	No	◦ hour: the hour that is used to replace the hour in the specified time. For example, you can set `hour=1`. Default value: None. ◦ hours: the number of hours by which you want to offset the specified time. For example, if you set `hours=1`, the function increases the hour by 1.

Parameter	Type	Required	Description
minute/minutes	Number	No	◦ minute: the minute that is used to replace the minute in the specified time. For example, you can set `minute=1`. Default value: None. ◦ minutes: the number of minutes by which you want to offset the specified time. For example, if you set `minutes=1`, the function increases the minute by 1.
second/seconds	Number	No	◦ second: the second that is used to replace the second in the specified time. For example, you can set `second=1`. Default value: None. ◦ seconds: the number of seconds by which you want to offset the specified time. For example, if you set `seconds=1`, the function increases the second by 1.
microsecond/microseconds	Number	No	◦ microsecond: the microsecond that is used to replace the microsecond in the specified time. For example, you can set `microsecond=1`. Default value: None. ◦ microseconds: the number of microseconds by which you want to offset the specified time. For example, if you set `microseconds=1`, the function increases the microsecond by 1.
week/weeks	Number	No	◦ week: the week that is used to replace the week in the specified time. For example, you can set `week=1`. Default value: None. ◦ weeks: the number of weeks by which you want to offset the specified time. For example, if you set `weeks=1`, the function increases the week by 1.

Parameter	Type	Required	Description
weekday	Number	No	The weekday that is used to replace the weekday in the specified time. For example, you can set `weekday=dt_MO(1)`. Default value: None.

- Returned result

  The time expression that is modified.

- Examples
  - Example 1

    Raw log entry:

    ```
    dt: 2018-10-10 1:2:3
    dt1: 2018-11-3 11:12:13
    dt2: 2018-10-1 10:10:10
    ```

    Transformation rule:

    ```
    e_set("dt_add",dt_add(dt_parse(v("dt")), dt1=dt_parse(v("dt1")), dt2=dt_parse(v("dt2"))
    ))
    ```

    Result:

    ```
    dt: 2018-10-10 1:2:3
    dt1: 2018-11-3 11:12:13
    dt2: 2018-10-1 10:10:10
    dt_add: 2018-11-12 02:04:06
    ```

  - Example 2

    Raw log entry:

    ```
    dt: 2018-10-11 02:03:04
    year: 2019
    ```

    Transformation rule:

    ```
    e_set("dt_add",dt_add(dt_parse(v("dt")),v("year")))
    ```

    Result:

    ```
    dt: 2018-10-11 02:03:04
    year: 2019
    dt_add: 2019-10-11 02:03:04
    ```

dt_MO

- Syntax

  ```
  dt_MO(A positive integer or a negative integer)
  ```

- Parameters

Parameter	Type	Required	Description
A positive integer or a negative integer	Number	Yes	The offset. To pass in a negative integer, use `op_neg(positive integer)`.

- Returned result

  The time that is offset.

- Example

  Raw log entry:

  ```
  time: 2019-08-13 02:03:04
  ```

  Transformation rule:

  ```
  e_set("dt_MO",dt_add(v("time"),weekday=dt_MO(1)))
  ```

  Result:

  ```
  time: 2019-08-13 02:03:04
  dt_MO: 2019-08-19 02:03:04
  ```

dt_TU

- Syntax

  ```
  dt_TU(A positive integer or a negative integer)
  ```

- Parameters

Parameter	Type	Required	Description
A positive integer or a negative integer	Number	Yes	The offset. To pass in a negative integer, use `op_neg(positive integer)`.

- Returned result

  The time that is offset.

- Examples

  For more information about examples, see dt_MO.

dt_WE

- Syntax

  ```
  dt_WE(A positive integer or a negative integer)
  ```

- Parameters

Parameter	Type	Required	Description
A positive integer or a negative integer	Number	Yes	The offset. To pass in a negative integer, use `op_neg(positive integer)`.

- Returned result

  The time that is offset.

- Examples

  For more information about examples, see dt_MO.

dt_TH

- Syntax

```
dt_TH(A positive integer or a negative integer)
```

- Parameters

Parameter	Type	Required	Description
A positive integer or a negative integer	Number	Yes	The offset. To pass in a negative integer, use `op_neg(positive integer)`.

- Returned result

  The time that is offset.

- Examples

  For more information about examples, see dt_MO.

dt_FR

- Syntax

```
dt_FR(A positive integer or a negative integer)
```

- Parameters

Parameter	Type	Required	Description
A positive integer or a negative integer	Number	Yes	The offset. To pass a negative integer, use `op_neg(positive integer)`.

- Returned result

  The time that is offset.

- Examples

  For more information about examples, see dt_MO.

dt_SA

- Syntax

```
dt_SA(A positive integer or a negative integer)
```

- Parameters

Parameter	Type	Required	Description
A positive integer or a negative integer	Number	Yes	The offset. To pass a negative integer, use `op_neg(positive integer)`.

- Returned result

  The time that is offset.

- Examples

  For more information about examples, see dt_MO.

dt_SU

- Syntax

```
dt_SU(A positive integer or a negative integer)
```

- Parameters

Parameter	Type	Required	Description
A positive integer or a negative integer	Number	Yes	The offset. To pass a negative integer, use `op_neg(positive integer)`.

- Returned result

  The time that is offset.

- Examples

  For more information about examples, see dt_MO.

dt_astimezone

- Syntax

```
dt_astimezone(value, tz=None, reset=False)
```

- Parameters

Parameter	Type	Required	Description
value	String, UNIX timestamp, or datetime object	Yes	The time expression that you want to modify.

Parameter	Type	Required	Description
tz	String	No	The time zone. Default value: None. For more information, see Time zone list.
reset	Bool	No	Specifies whether to reset the time zone. Default value: False. The value False indicates that the function does not reset the time zone. The value True indicates that the function resets the time zone and returns the time in the specified time zone.

- Returned result

  The datetime object in the specified time zone.

- Examples
  - Example 1

    Raw log entry:

    ```
    time: 2019-06-03 2:41:26
    tz: UTC
    ```

    Transformation rule:

    ```
    e_set("dt_astimezone",dt_astimezone(dt_parse(v("time")), v("tz")))
    ```

    Result:

    ```
    time: 2019-06-03 2:41:26
    tz: UTC
    dt_astimezone: 2019-06-03 02:41:26+00:00
    ```

  - Example 2

    Raw log entry:

    ```
    time: 2019-06-01 10:10:10+10:00
    tz: Asia/Tokyo
    ```

    Transformation rule:

    ```
    e_set("dt_astimezone",dt_astimezone(v("time"), v("tz"),reset=True))
    ```

    Result:

    ```
    time: 2019-06-01 10:10:10+10:00
    tz: Asia/Tokyo
    dt_astimezone: 2019-06-01 10:10:10+09:00
    ```

- Example 3

  Raw log entry:

```
time: 2019-06-01 10:10:10+08:00
tz: Asia/Tokyo
```

  Transformation rule:

```
e_set("dt_astimezone",dt_astimezone(v("time"), v("tz"),reset=False))
e_set("dt_astimezone_true",dt_astimezone(v("time"), v("tz"),reset=True))
```

  Result:

```
time: 2019-06-01 10:10:10+08:00
tz: Asia/Tokyo
dt_astimezone: 2019-06-01 11:10:10+09:00
dt_astimezone_true: 2019-06-01 10:10:10+09:00
```

dt_diff

- Syntax

```
dt_diff(value 1, value 2, unit='second', normalize='floor')
```

- Parameters

Parameter	Type	Required	Description
value 1	String, UNIX timestamp, or datetime object	Yes	Time expression 1 that you want to compare.
value 2	String, UNIX timestamp, or datetime object	Yes	Time expression 2 that you want to compare.
unit	String	No	The precision in which you want the function to return the time difference. Default value: `second`. Valid values: second, `microsecond`, `millisecond`, `minutes`, `hours`, and `day`.

Parameter	Type	Required	Description
normalize	String	No	The numeric format in which you want the function to return the result. Valid values: ○ `floor`: rounds a number down to the nearest integer. Default value: floor. ○ `int`: removes the decimal part of a number. ○ `round`: retains N decimal places. ○ `ceil`: rounds a number up to the nearest integer.

- Returned result

  The difference between two values, which is in the specified precision.

- Examples
  - Example 1

    Raw log entry:

    ```
    time1: 2018-10-1 10:10:10
    time2: 2018-10-1 10:10:10
    ```

    Transformation rule:

    ```
    e_set("diff",dt_diff(v("time1"), v("time2")))
    ```

    Result:

    ```
    time1: 2018-10-1 10:10:10
    time2: 2018-10-1 10:10:10
    diff: 0
    ```

  - Example 2

    Raw log entry:

    ```
    time1: 2018-10-1 11:10:10
    time2: 2018-10-1 10:10:10
    ```

    Transformation rule:

    ```
    e_set("diff",dt_diff(v("time1"), v("time2")))
    ```

    Result:

    ```
    time1: 2018-10-1 11:10:10
    time2: 2018-10-1 10:10:10
    diff: 3600
    ```

- Example 3

  Raw log entry:

  ```
  time1: 2018-10-1 11:10:11
  time2: 2018-10-1 10:10:10
  unit: microsecond
  ```

  Transformation rule:

  ```
  e_set("diff",dt_diff(v("time1"), v("time2"),v("unit")))
  ```

  Result:

  ```
  time1: 2018-10-1 11:10:11
  time2: 2018-10-1 10:10:10
  unit: microsecond
  diff: 1000000
  ```

- Example 4

  Raw log entry:

  ```
  time1: 2018-10-1 11:11:59
  time2: 2018-10-1 10:10:00
  unit: minute
  normalize: floor
  ```

  Transformation rule:

  ```
  dt_diff(v("time1"), v("time2"),v("unit"),v("normalize"))
  ```

  Result:

  ```
  time1: 2018-10-1 11:11:59
  time2: 2018-10-1 10:10:00
  unit: minute
  normalize: floor
  diff: 61
  ```

- Example 5: Calculate the difference between two time values without the dates involved.

  Raw log entry:

  ```
  time1: 10:00:00
  time2: 11:00:00
  unit: second
  normalize: floor
  ```

  Transformation rule:

  ```
  dt_diff(v("time1"), v("time2"),v("unit"),v("normalize"))
  ```

  Result:

  ```
  time1: 10:00:00
  time2: 11:00:00
  unit: second
  normalize: floor
  diff: 3600
  ```

12.6.8. Regular expression functions

This topic describes the syntax and parameters of regular expression functions. This topic also provides examples on how to use the functions.

Functions

Category	Function	Description
Value extraction	regex_select	Extracts a value that matches a regular expression.
	regex_findall	Extracts all values that match a regular expression.
Evaluation	regex_match	Checks whether a value matches a regular expression.
Replacement	regex_replace	Replaces the characters that match a regular expression in a string.
Splitting	regex_split	Splits a string into an array of strings.

regex_select

- Syntax

  ```
  regex_select(value, r"regular expression", mi=None, gi=None)
  ```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary	Yes	The input value.

Parameter	Type	Required	Description
regular expression	String	Yes	The regular expression.
mi	int	No	The sequence number of the value that is matched in the input value and will be returned. Default values: None and 0. The default values indicate that the first value that is matched will be returned.
gi	int	No	The sequence number of the group that is used for matching in the regular expression. Default values: None and 0. The default values indicate that the first group is used.

- Response

  The extracted value is returned.

- Examples
  - Example 1: Extract the first value that matches the regular expression from the str field.

    Raw log:

    ```
    str: iZbp1a65x3r1vhpe94fi2qZ
    ```

    Transformation rule:

    ```
    e_set("regex", regex_select(v("str"), r"\d+"))
    e_set("regex2", regex_select(v("str"), r"\d+", mi=None))
    e_set("regex3", regex_select(v("str"), r"\d+", mi=0))
    ```

    Result:

    ```
    regex:1
    regex2:1
    regex3:1
    str:iZbp1a65x3r1vhpe94fi2qZ
    ```

- Example 2: Extract the first and second values that match the regular expression from the str field.

  Raw log:

  ```
  str: abc123 xyz456
  ```

  Transformation rule:

  ```
  # Extract the first value that matches the regular expression from the str field.
  e_set("regex", regex_select(v("str"), r"\d+"))
  # Extract the second value that matches the regular expression from the str field.
  e_set("regex2", regex_select(v("str"), r"\d+", mi=1))
  ```

  Result:

  ```
  regex:   123
  regex2:  456
  str: abc123 xyz456
  ```

- Example 3

  Raw log:

  ```
  str: abc123 xyz456
  ```

  Transformation rule:

  ```
  # Extract the first value that matches the first group in the regular expression from t
  he str field.
  e_set("regex", regex_select(v("str"),r"[a-z]+(\d+)",gi=0))
  # Extract the second value that matches the first group in the regular expression from
  the str field.
  e_set("regex2", regex_select(v("str"),r"[a-z]+(\d+)",mi=1,gi=0))
  # Extract the first value that matches the first group in the regular expression from t
  he str field.
  e_set("regex3", regex_select(v("str"),r"([a-z]+)(\d+)",gi=0))
  # Extract the first value that matches the second regular expression from the str field
  .
  e_set("regex4", regex_select(v("str"),r"([a-z]+)(\d+)",gi=1))
  ```

  Result:

  ```
  str: abc123 xyz456
  regex:   123
  regex2:  456
  regex3:  abc
  regex4:  123
  ```

regex_findall

- Syntax

  ```
  regex_findall(value, r"regular expression")
  ```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary	Yes	The input value.
regular expression	String	Yes	The regular expression.

- Response

  The values that match the regular expression are returned.

- Example: Extract all numbers from the str field.

  Raw log:

  ```
  str: iZbp1a65x3r1vhpe94fi2qZ
  ```

  Transformation rule:

  ```
  e_set("regex_findall", regex_findall(v("str"),r"\d+"))
  ```

  Result:

  ```
  str: iZbp1a65x3r1vhpe94fi2qZ
  regex_findall:  ["1", "65", "3", "1", "94", "2"]
  ```

regex_match

- Syntax

  ```
  regex_match(value, r"regular expression", full=False)
  ```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary	Yes	The input value.
regular expression	String	Yes	The regular expression.
full	Bool	No	Specifies whether to perform exact match. Default value: False.

- Response

  The value True or the value False is returned.

- Example: Check whether the str field contains numbers.

  Raw log:

  ```
  str: iZbp1a65x3r1vhpe94fi2qZ
  ```

  Transformation rule:

```
# Check whether the str field contains numbers.
e_set("regex_match", regex_match(v("str"),r"\d+"))
# Check whether the str field contains only numbers.
e_set("regex_match2", regex_match(v("str"),r"\d+",full=True))
```

Result:

```
str: iZbp1a65x3r1vhpe94fi2qZ
regex_match:  True
regex_match2:  False
```

regex_replace

- Syntax

```
regex_replace(value, r"regular expression", replace="", count=0)
```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary	Yes	The input value whose characters you want to replace.
regular expression	String	Yes	The regular expression.
replace	String	No	The characters that you want to use to replace matched characters in the input value. Default value: null. The default value indicates that matched characters are deleted. You can specify a regular expression. Example: `r"\1****\2"`. This value indicates that the output value must match the regular expression. ○ `\1` indicates the first group. ○ `\2` indicates the second group.

Parameter	Type	Required	Description
count	Number	No	The number of times that you want to replace matched characters. Default value: 0. The default value indicates that all matched characters are replaced.

- Response

  The value after replacement is returned.

- Examples
  - Example 1: Replace all numbers in the str field with 13.

    Raw log:

    ```
    str: iZbp1a65x3r1vhpe94fi2qZ
    replace: 13
    ```

    Transformation rule:

    ```
    e_set("regex_replace", regex_replace(v("str"),r"\d+",v("replace")))
    ```

    Result:

    ```
    str: iZbp1a65x3r1vhpe94fi2qZ
    replace: 13
    regex_replace:  iZbp13a13x13r13vhpe13fi13qZ
    ```

- Example 2: Mask four digits in the middle of a mobile phone number.

  Raw log:

  ```
  iphone: 13900001234
  ```

  Transformation rule:

  ```
  e_set(
      "sec_iphone",
      regex_replace(v("iphone"), r"(\d{0,3})\d{4}(\d{4})", replace=r"\1****\2"),
  )
  ```

  > **Note**
  > - `replace=r"\1****\2"` indicates that the value after replacement must match the regular expression `r"\1****\2"`.
  > - `\1` indicates the first group. In this example, `(\d{0,3})` is the first group.
  > - `\2` indicates the second group. In this example, `(\d{4})` is the second group.

  Result:

  ```
  iphone: 13900001234
  sec_iphone: 139****1234
  ```

regex_split

- Syntax

  ```
  regex_split(value, r"regular expression", maxsplit=0)
  ```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary	Yes	The input value that you want to split.
regular expression	String	Yes	The regular expression.
maxsplit	int	No	The maximum number of times that the input value can be split. Default value: 0. The default value indicates that the input value is split based on all matched characters. The value 1 indicates that the input value is split based on only the first matched character.

- Response

  An array that contains the values after splitting is returned.

- Example: Split the str field by number.

  Raw log:

  ```
  str: iZbp1a65x3r1vhpe94fi2qZ
  ```

  Transformation rule:

  ```
  e_set("regex_split", regex_split(v("str"),r"\d+"))
  ```

  Result:

  ```
  str: iZbp1a65x3r1vhpe94fi2qZ
  regex_split:  ["iZbp", "a", "x", "r", "vhpe", "fi", "qZ"]
  ```

12.6.9. Grok function

This topic describes the syntax and parameters of the Grok function. This topic also provides several examples of the Grok function.

We recommend that you use the Grok function instead of Regular expression functions. You can also integrate the Grok function and regular expression functions. Examples:

```
e_match("content", grok(r "\w +: (%{IP})")) # The Grok pattern matches the abc: 192.168.0.0 or xyz: 192.168.1.1 pattern of log data.
e_match("content", grok(r "\w +: (%{IP})",escape=True)) # The Grok pattern does not match the abc: 192.168.0.0 pattern of log data. The Grok pattern matches the \w +: 192.168.0.0 pattern of log data.
```

Description

The Grok function extracts specified values based on a regular expression.

- Function format

  ```
  grok(pattern, escape=False, extend=None)
  ```

- Grok syntax

  ```
  %{SYNTAX}
  %{SYNTAX:NAME}
  ```

  In the Grok syntax, SYNTAX indicates a predefined regular expression, and NAME indicates a naming group. Examples:

  ```
  "%{IP}"               # Equivalent to r"(?:\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})"
  "%{IP:source_id}"     # Equivalent to r"(? P<source_id>\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})"
  ("%{IP}")             # Equivalent to r"(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})"
  ```

Some Grok patterns are captured based on the predefined naming group. Therefore, you can use only the %{SYNTAX} syntax. These Grok patterns are commonly used in statement parsing. For more information, see the Log formats section in Grok patterns. Examples:

```
"%{SYSLOGBASE}"
"%{COMMONAPACHELOG}"
"%{COMBINEDAPACHELOG}"
"%{HTTPD20_ERRORLOG}"
"%{HTTPD24_ERRORLOG}"
"%{HTTPD_ERRORLOG}"
...
```

Some Grok patterns contain non-capturing groups. Examples:

```
"%{INT}"
"%{YEAR}"
"%{HOUR}"
...
```

- Parameters

Parameter	Type	Required	Description
pattern	String	Yes	The Grok syntax. For more information, see Grok patterns.
escape	Boolean	No	Specifies whether to escape special characters that are related to regular expressions in other non-Grok patterns. Default value: False.
extend	Dict	No	The custom Grok expression.

Examples

- Example 1: Extract the date and reference content.
  - Raw log:

    ```
    content: 2019 June 24 "I am iron man"
    ```

  - Transformation rule:

    ```
    e_regex('content',grok('%{YEAR:year} %{MONTH:month} %{MONTHDAY:day} %{QUOTEDSTRING:motto}'))
    ```

- Transformation result:

```
content: 2019 June 24 "I am iron man"
year: 2019
month: June
day: 24
motto: "I am iron man"
```

- Example 2: Extract an HTTP request log.
  - Raw log:

```
content: 55.3.244.1 GET /index.html 15824 0.043
```

  - Transformation rule:

```
e_regex('content',grok('%{IP:client} %{WORD:method} %{URIPATHPARAM:request} %{NUMBER:bytes} %{NUMBER:duration}'))
```

  - Transformation result:

```
content: 55.3.244.1 GET /index.html 15824 0.043
client: 10.0.0.0
method: GET
request: /index.html
bytes: 15824
duration: 0.043
```

- Example 3: Extract an Apache log.
  - Raw log:

```
content: 127.0.0.1 - - [13/Apr/2015:17:22:03 +0800] "GET /router.php HTTP/1.1" 404 285 "-" "curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.15.3 zlib/1.2.3 libidn/1.18 libssh2/1.4.2"
```

  - Transformation rule:

```
e_regex('content',grok('%{COMBINEDAPACHELOG}'))
```

  - Transformation result:

```
content: 127.0.0.1 - - [13/Apr/2015:17:22:03 +0800] "GET /router.php HTTP/1.1" 404 285 "-" "curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.15.3 zlib/1.2.3 libidn/1.18 libssh2/1.4.2"
clientip: 127.0.0.1
ident: -
auth: -
timestamp: 13/Apr/2015:17:22:03 +0800
verb: GET
request: /router.php
httpversion: 1.1
response: 404
bytes: 285
referrer: "-"
agent: "curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.15.3 zlib/1.2.3 libidn/1.18 libssh2/1.4.2"
```

- Example 4: Set a log to the default Syslog format.
  - Raw log:

    ```
    content: May 29 16:37:11 sadness logger: hello world
    ```

  - Transformation rule:

    ```
    e_regex('content',grok('%{SYSLOGBASE} %{DATA:message}'))
    ```

  - Transformation result:

    ```
    content: May 29 16:37:11 sadness logger: hello world
    timestamp: May 29 16:37:11
    logsource: sadness
    program: logger
    message: hello world
    ```

- Example 5: Escape special characters.
  - Raw log:

    ```
    content: Nov  1 21:14:23 scorn kernel: pid 84558 (expect), uid 30206: exited on signal
    3
    ```

  - Transformation rule:

    ```
    e_regex('content',grok(r'%{SYSLOGBASE} pid %{NUMBER:pid} \(%{WORD:program}\), uid %{NUM
    BER:uid}: exited on signal %{NUMBER:signal}'))
    ```

    The transformation rules contain parentheses (). If you do not use escape characters, add the escape=True parameter, as shown in the following example:

    ```
    e_regex('content',grok('%{SYSLOGBASE} pid %{NUMBER:pid} (%{WORD:program}), uid %{NUMBER
    :uid}: exited on signal %{NUMBER:signal}', escape=True))
    ```

  - Transformation result:

    ```
    content: Nov  1 21:14:23 scorn kernel: pid 84558 (expect), uid 30206: exited on signal
    3
    timestamp: Nov  1 21:14:23
    logsource: scorn
    program: expect
    pid: 84558
    uid: 30206
    signal: 3
    ```

- Example 6: Customize a Grok expression.
  - Raw log:

    ```
    content: Beijing-1104,gary 25 "never quit"
    ```

  - Transformation rule:

    ```
    e_regex('content',grok('%{ID:user_id},%{WORD:name} %{INT:age} %{QUOTEDSTRING:motto}',ex
    tend={'ID': '%{WORD}-%{INT}'}))
    ```

- Transformation result:

```
content: Beijing-1104,gary 25 "never quit"
user_id: Beijing-1104
name: gary
age: 25
motto: "never quit"
```

- Example 7: Match JSON data.
  - Raw log:

```
content: 2019-10-29 16:41:39,218 - INFO: owt.AudioFrameConstructor - McsStats: {"event"
:"mediaStats","connectionId":"331578616547393100","durationMs":"5000","rtpPackets":"250
","rtpBytes":"36945","nackPackets":"0","nackBytes":"0","rtpIntervalAvg":"20","rtpInterv
alMax":"104","rtpIntervalVar":"4","rtcpRecvPackets":"0","rtcpRecvBytes":"0","rtcpSendPa
ckets":"1","rtcpSendBytes":"32","frame":"250","frameBytes":"36945","timeStampOutOfOrder
":"0","frameIntervalAvg":"20","frameIntervalMax":"104","frameIntervalVar":"4","timeStam
pIntervalAvg":"960","timeStampIntervalMax":"960","timeStampIntervalVar":"0"}
```

  - Transformation rule:

```
e_regex('content',grok('%{EXTRACTJSON}'))
```

  - Transformation result:

```
content: 2019-10-29 16:41:39,218 - INFO: owt.AudioFrameConstructor - McsStats: {"event"
:"mediaStats","connectionId":"331578616547393100","durationMs":"5000","rtpPackets":"250
","rtpBytes":"36945","nackPackets":"0","nackBytes":"0","rtpIntervalAvg":"20","rtpInterv
alMax":"104","rtpIntervalVar":"4","rtcpRecvPackets":"0","rtcpRecvBytes":"0","rtcpSendPa
ckets":"1","rtcpSendBytes":"32","frame":"250","frameBytes":"36945","timeStampOutOfOrder
":"0","frameIntervalAvg":"20","frameIntervalMax":"104","frameIntervalVar":"4","timeStam
pIntervalAvg":"960","timeStampIntervalMax":"960","timeStampIntervalVar":"0"}
json:{"event":"mediaStats","connectionId":"331578616547393100","durationMs":"5000","rtp
Packets":"250","rtpBytes":"36945","nackPackets":"0","nackBytes":"0","rtpIntervalAvg":"2
0","rtpIntervalMax":"104","rtpIntervalVar":"4","rtcpRecvPackets":"0","rtcpRecvBytes":"0
","rtcpSendPackets":"1","rtcpSendBytes":"32","frame":"250","frameBytes":"36945","timeSt
ampOutOfOrder":"0","frameIntervalAvg":"20","frameIntervalMax":"104","frameIntervalVar":
"4","timeStampIntervalAvg":"960","timeStampIntervalMax":"960","timeStampIntervalVar":"0
"}
```

- Example 8: Parse a W3C log.
  - Raw log:

```
content: 2018-12-26 00:00:00 W3SVC2 application001 192.168.0.0 HEAD / - 8000 - 10.0.0.0
HTTP/1.0 - - - - 404 0 64 0 19 0
```

  - Transformation rule:

    Unavailable fields in the W3C log are displayed as hyphens (-). In Grok expressions, hyphens (-) are used to match these fields.

```
e_regex("content",grok('%{DATE:data} %{TIME:time} %{WORD:s_sitename} %{WORD:s_computern
ame} %{IP:s_ip} %{WORD:cs_method} %{NOTSPACE:cs_uri_stem} - %{NUMBER:s_port} - %{IP:c_i
p} %{NOTSPACE:cs_version} - - - - %{NUMBER:sc_status} %{NUMBER:sc_substatus} %{NUMBER:s
c_win32_status} %{NUMBER:sc_bytes} %{NUMBER:cs_bytes} %{NUMBER:time_taken}'))
```

- Transformation result:

```
content: 2018-12-26 00:00:00 W3SVC2 application001 192.168.0.0 HEAD / - 8000 - 10.0.0.0
HTTP/1.0 - - - - 404 0 64 0 19 0
data: 18-12-26
time: 00:00:00
s_sitename: W3SVC2
s_computername: application001
s_ip: 192.168.0.0
cs_method: HEAD
cs_uri_stem: /
s_port: 8000
c_ip: 10.0.0.0
cs_version: HTTP/1.0
sc_status: 404
sc_substatus: 0
sc_win32_status: 64
sc_bytes: 0
cs_bytes: 19
time_taken: 0
```

12.6.10. Structured data functions

This topic describes the syntax and parameters of functions that parse JSON-formatted and XML-formatted data. This topic also provides examples on how to use the functions.

Functions

Type	Function	Description
JSON	json_select	Extracts or calculates a specific value from a JSON expression based on the JMESPath syntax.
	json_parse	Parses a specified value into a JSON object.
XML	xml_to_json	Converts XML-formatted data to JSON-formatted data and then expands the converted data.

json_select

The json_select function is used to extract or calculate a specific value from a JSON expression based on the JMESPath syntax.

- Syntax

```
json_select(value, jmes, default=None, restrict=False)
```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary	Yes	The JSON expression or field from which a field is extracted.

Parameter	Type	Required	Description
jmes	String	Yes	The JMESPath expression that specifies the field that is extracted.
default	Arbitrary	No	If the field that you want to extract does not exist, the value of the default parameter is returned. Default value: None.
restrict	Boolean	No	Specifies whether the restricted mode is enabled if the value of the field that you want to extract is in an invalid JSON format. Default value: False. Valid values: ◦ False: The invalid format issue is ignored and the system continues to transform the data. The value of the *default* parameter is returned. ◦ True: The invalid format issue is reported and the system stops transforming data. The log entry is dropped.

- Response

  The extracted field value is returned.

- Examples
  - Example 1: Extract the name field from the content field. The value of the name field is returned.
    - Raw log entry:

      ```
      content:  {"name": "xiaoming", "age": 10}
      ```

    - Transformation rule:

      ```
      e_set("json_filter",json_select(v("content"), "name"))
      ```

    - Result:

      ```
      content:  {"name": "xiaoming", "age": 10}
      json_filter:  xiaoming
      ```

- Example 2: Extract the name field from the content field. All values in the name field are returned.
  - Raw log entry:

    ```
    content:  {"name": ["xiaoming", "xiaowang", "xiaoli"], "age": 10}
    ```

  - Transformation rule:

    ```
    e_set("json_filter",json_select(v("content"), "name[*]"))
    ```

  - Result:

    ```
    content:  {"name": ["xiaoming", "xiaowang", "xiaoli"], "age": 10}
    json_filter:  ["xiaoming", "xiaowang", "xiaoli"]
    ```

- Example 3: Extract the value of the name3 field from the content field. If the name3 field does not exist, the value None is returned.
  - Raw log entry:

    ```
    content:  {"name": "xiaoming", "age": 10}
    ```

  - Transformation rule:

    ```
    e_set("json_filter",json_select(v("content"), "name3", default="None"))
    ```

  - Result:

    ```
    content:  {"name": "xiaoming", "age": 10}
    json_filter: None
    ```

json_parse

The json_parse function is used to parse a specified value into a JSON object.

- Syntax

  ```
  json_parse(value, default=None, restrict=False)
  ```

- Parameters

Parameter	Type	Required	Description
value	String	Yes	The field that you want to parse.
default	Arbitrary	No	If the field that you want to parse does not exist, the value of the default parameter is returned. Default value: None.

Parameter	Type	Required	Description
restrict	Boolean	No	Specifies whether the restricted mode is enabled if the value of the field that you want to extract is in an invalid JSON format. Default value: False. Valid values: ○ False: The invalid format issue is ignored and the system continues to transform the data. The value of the *default* parameter is returned. ○ True: The invalid format issue is reported and the system stops transforming data. The log entry is dropped.

- Response

  A JSON object is returned.

- Example
  - Raw log entry:

    ```
    content:  {"abc": 123, "xyz": "test" }
    ```

  - Transformation rule:

    ```
    e_set("json",json_parse(v("content")))
    ```

  - Result:

    ```
    content:  {"abc": 123, "xyz": "test" }
    json:  {"abc": 123, "xyz": "test"}
    ```

xml_to_json

The xml_to_json function is used to convert XML-formatted data to JSON-formatted data and then expand the converted data.

- Syntax

  ```
  xml_to_json(source)
  ```

- Parameters

Parameter	Type	Required	Description
source	String	Yes	The field that you want to convert.

- Response

  JSON-formatted data is returned.

- Example

- Raw log entry:

```
str : <data><country name="Liechtenstein"><rank>1</rank><year>2008</year><gdppc>141100</gdppc><neighbor name="Austria" direction="E"/><neighbor name="Switzerland" direction="W"/></country><country name="Singapore"><rank>4</rank><year>2011</year><gdppc>59900</gdppc><neighbor name="Malaysia" direction="N"/></country><country name="Panama"><rank>68</rank><year>2011</year><gdppc>13600</gdppc><neighbor name="Costa Rica" direction="W"/><neighbor name="Colombia" direction="E"/></country></data>
```

- Transformation rule:

```
e_set("str_json",xml_to_json(v("str")))
```

- Result:

```
str:<data><country name="Liechtenstein"><rank>1</rank><year>2008</year><gdppc>141100</gdppc><neighbor name="Austria" direction="E"/><neighbor name="Switzerland" direction="W"/></country><country name="Singapore"><rank>4</rank><year>2011</year><gdppc>59900</gdppc><neighbor name="Malaysia" direction="N"/></country><country name="Panama"><rank>68</rank><year>2011</year><gdppc>13600</gdppc><neighbor name="Costa Rica" direction="W"/><neighbor name="Colombia" direction="E"/></country></data>str_json:{"data": {"country": [{"@name": "Liechtenstein", "rank": "1", "year": "2008", "gdppc": "141100", "neighbor": [{"@name": "Austria", "@direction": "E"}, {"@name": "Switzerland", "@direction": "W"}]}, {"@name": "Singapore", "rank": "4", "year": "2011", "gdppc": "59900", "neighbor": {"@name": "Malaysia", "@direction": "N"}}, {"@name": "Panama", "rank": "68", "year": "2011", "gdppc": "13600", "neighbor": [{"@name": "Costa Rica", "@direction": "W"}, {"@name": "Colombia", "@direction": "E"}]}]}}
```

12.6.11. IP address parsing functions

This topic describes the syntax and parameters of IP address parsing functions. This topic also provides examples on how to use IP address parsing functions.

Functions

Function	Description
geo_parse	Parses an IP address into the information about the city, province, and country of the IP address.
ip_cidrmatch	Checks whether an IP address belongs to a CIDR block.
ip_version	Checks whether the version of an IP address is IPv4 or IPv6.
ip_type	Identifies the type of an IP address and checks whether the type of the IP address is private or public.
ip_makenet	Converts an IP address to a CIDR block.
ip_to_format	Converts a CIDR block to the format in which the prefix length or netmask of the CIDR block is specified.
ip_overlaps	Checks whether two CIDR blocks overlap.

Function	Description
ip2long	Converts an IP address to a value of the long type.
long2ip	Converts a value of the long type to an IP address.

geo_parse

The geo_parse function parses an IP address into the information about the city, province, and country of the IP address.

- Syntax

```
geo_parse(ip, ip_db="SLS-GeoIP", keep_fields=None, provider="ipip", ip_sep=None)
```

- Parameters

Parameter	Type	Required	Description
ip	String	Yes	The IP address that you want to parse to obtain the information about the city, province, and country of the IP address. If you want to enter multiple IP addresses, you can specify the delimiter by using the *ip_sep* parameter.
ip_db	String	Yes	The IP address database that is used to parse an IP address into the information about the city, province, and country of the IP address. Valid values: ◦ SLS-GeoIP: the built-in IP address database of Log Service. This is the default value. ◦ Custom IP address database: Set the value to `res_oss_file(endpoint, ak_id, ak_key, bucket, file, format='binary', change_detect_interval=0,fetch_interval=2,refresh_retry_max=60,encoding='utf8',error='ignore')`. For more information about the parameters in the res_oss_file function, see res_oss_file.

Parameter	Type	Required	Description
keep_fields	Tuple	No	The keys that are included in the response. ◦ If you use the built-in IP address database to parse an IP address, the following keys are included in the response by default: ■ city: the name of the city ■ province: the name of the province ■ country: the name of the country ■ city_en: the administrative region code or name of the city ■ province_en: the administrative region code or name of the province ■ country_en: the code or name of the country or region ■ isp: the name of the Internet service provider (ISP) ■ lat: the latitude of the location to which the IP address belongs ■ lon: the longitude of the location to which the IP address belongs ◦ If you use a custom IP address database to parse an IP address, the following keys are included in the response by default: ■ city: the name of the city ■ province: the name of the province ■ country: the name of the country Example: `keep_fields=("city","country")`, which indicates that the `city` and `country` keys are returned. The `keep_fields` parameter can also be used to rename the keys. Example: `(("city","cty"),("country","state"))`, which indicates that the city and country keys are renamed `cty` and `state` in the response.

Parameter	Type	Required	Description
provider	String	No	This parameter is valid only when the *ip_db* parameter is set to a *custom IP address database*. Valid values: ◦ ipip: The binary IP address database that is provided by IPIP in the IPDB format is used to parse an IP address. To download the database, visit ipip. This is the default value. ◦ ip2location: The global binary IP address database that is provided by IP2Location is used to parse an IP address. To download the database, visit IP2Location. Only a binary IP address database is supported.
ip_sep	String	No	The IP address delimiter. The delimiter is used to delimit a string of IP addresses into multiple IP addresses. The response is in the JSON format. Default value: None. This value specifies that a string of IP addresses is not delimited.

- Response

  A dictionary is returned in the following format:

```
{
  "city": "...",
  "province":"...",
  "country": "..."
}
```

- Examples
  - Example 1: Use the built-in IP address database of Log Service to query data.
    - Raw log:

```
ip : 203.0.113.1
```

    - Transformation rule:

```
e_set("geo", geo_parse(v("ip")))
```

    - Result:

```
ip : 203.0.113.1
geo: {"city":"Hangzhou","province":"Zhejiang province","country":"China","isp":"China Mobile","lat":30.16,"lon":120.12}
```

- Example 2: Use the built-in IP address database of Log Service to query data. The function parses a log field that contains multiple IP addresses and returns the information about the city, province, and country of each IP address.
  - Raw log:

    ```
    ip : 203.0.113.4, 192.0.2.2, 198.51.100.2
    ```

  - Transformation rule:

    ```
    e_set("geo", geo_parse(v("ip"), ip_sep=","))
    ```

  - Result:

    ```
    ip : 203.0.113.4, 192.0.2.2, 198.51.100.2
    geo : {"203.0.113.4": {"country_en": "CN", "province_en": "330000", "city_en": "33020
    0", "country": "China", "province": "Zhejiang province", "city": "Ningbo", "isp": "Ch
    ina Telecom", "lat": 29.8782, "lon": 121.549}, "192.0.2.2": {"country_en": "CN", "pro
    vince_en": "320000", "city_en": "321300", "country": "China", "province": "Jiangsu pr
    ovince", "city": "Suqian", "isp": "China Telecom", "lat": 33.9492, "lon": 118.296}, "
    198.51.100.2": {"country_en": "CN", "province_en": "330000", "city_en": "330500", "co
    untry": "China", "province": "Zhejiang province", "city": "Huzhou", "isp": "China Tel
    ecom", "lat": 30.8703, "lon": 120.093}}
    ```

- Example 3: Use a custom IP address database to query data.
  - Raw log:

    ```
    ip : 203.0.113.1
    ```

  - Transformation rule:

    ```
    e_set("geo",geo_parse(v("ip"), ip_db=res_oss_file(endpoint='http://oss-cn-hangzhou.al
    iyuncs.com',
                                                      ak_id='your ak_id',
                                                      ak_key='your ak_key',
                                                      bucket='your bucket', file='ipipfree
    .ipdb',
                                                                   format='binary',change
    _detect_interval=20)))
    ```

  - Result:

    ```
    ip : 203.0.113.1
    geo : {"city": "Hangzhou", "province":"Zhejiang province","country": "China"}
    ```

- Example 4: Use a custom IP address database to query data. The function returns the specified keys and renames the keys.
  - Raw log:

```
ip : 203.0.113.1
```

  - Transformation rule:

```
e_set("geo",geo_parse(v("ip"), ip_db=res_oss_file(endpoint='http://oss-cn-hangzhou.aliyuncs.com',
                                                  ak_id='your ak_id',
                                                  ak_key='your ak_key',
                                                  bucket='your bucket', file='ipipfree.ipdb',
                                                                        format='binary',change_detect_interval=20),keep_fields=(("city","cty"),("country","state"),("province","pro"))))
```

  - Result:

```
ip : 203.0.113.1
geo : { "state": "China","pro": "Zhejiang province","cty": "Hangzhou"}
```

- Example 5: Use a custom IP address database to query data. The function returns the specified keys.
  - Raw log:

```
ip : 203.0.113.1
```

  - Transformation rule:

```
e_set("geo",geo_parse(v("ip"), ip_db=res_oss_file(endpoint='http://oss-cn-hangzhou.aliyuncs.com',
                                                  ak_id='your ak_id',
                                                  ak_key='your ak_key',
                                                  bucket='your bucket', file='ipipfree.ipdb',
                                                                        format='binary',change_detect_interval=20),keep_fields=("country","province")))
```

  - Result:

```
ip : 203.0.113.1
geo : { "country": "China","province": "Zhejiang province"}
```

- Example 6: Use a custom IP address database to query data and use the global binary IP address database that is provided by IP2Location to parse the data. The function returns the specified keys.
  - Raw log:

    ```
    ip : 203.0.113.2
    ```

  - Transformation rule:

    ```
    e_set("geo", geo_parse(v("ip"), ip_db=res_oss_file(endpoint='http://oss-cn-hangzhou.aliyuncs.com',ak_id="your ak_id", ak_key="your ak_serect", bucket='log-etl-staging', file='your ip2location bin file', format='binary', change_detect_interval=20),provider="ip2location"))
    ```

  - Result:

    ```
    ip : 203.0.113.2
    geo : {"city":"Dearborn","province":"Michigan","country":"United States"}
    ```

  If you set the value of the provider parameter to ip2location, the open source SDK for Python that is provided by IP2Location is used in data transformation. The SDK for Python that is provided by IP2Location can be used to parse the following fields. If a field fails to be parsed, you must check whether the field is included in the IP address database provided by IP2Location.

  ```
  country_short
  country_long / The country field is specified for data transformation.
  region / The province field is specified for data transformation.
  city
  isp
  latitude
  longitude
  domain
  zipcode
  timezone
  netspeed
  idd_code
  area_code
  weather_code
  weather_name
  mcc
  mnc
  mobile_brand
  elevation
  usage_type
  ```

  For more information, visit IP2Location Python SDK.

- Example 7: Use a custom IP address database to query data. The function parses a log field that contains multiple IP addresses and returns the information about the city, province, and country of each IP address.
  - Raw log:

```
ip : 203.0.113.3, 192.0.2.1, 198.51.100.1
```

  - Transformation rule:

```
e_set("geo", geo_parse(v("ip"), ip_db=res_oss_file(endpoint='http://oss-cn-hangzhou.aliyuncs.com',
                                                   ak_id="ak_id",
                                                   ak_key="ak_serect",
                                                   bucket='log-etl-staging',
                                                   file='calendar.csv/IP2LOCATION-LITE-DB3.BIN',
                                                   format='binary', change_detect_interval=20),
                                provider="ip2location", ip_sep=","))
```

  - Result:

```
ip : 203.0.113.3, 192.0.2.1, 198.51.100.1
geo : {"203.0.113.3": {"city": "Dearborn", "province": "Michigan", "country": "United States"}, "192.0.2.1": {"city": "Hangzhou", "province": "Zhejiang", "country": "China"}, "198.51.100.1": {"city": "Hangzhou", "province": "Zhejiang", "country": "China"}}
```

ip_cidrmatch

The ip_cidrmatch function checks whether an IP address belongs to a CIDR block and returns a Boolean value. If the IP address belongs to the CIDR block, the function returns true. Otherwise, the function returns false. The IP address can be an IPv4 address or an IPv6 address.

- Syntax

```
ip_cidrmatch(cidr_subnet, ip, default="")
```

- Parameters

Parameter	Type	Required	Description
cidr_subnet	String	Yes	The CIDR block. Example: 203.0.113.1/25.
ip	String	Yes	The IP address.
default	Arbitrary	No	The default value. If the IP address does not belong to the CIDR block, the value of this parameter is returned. This parameter can be empty.

- Response

  If the specified IP address belongs to the specified CIDR block, the function returns true. Otherwise, the function returns false.

- Examples
  - Example 1: The specified IPv4 address belongs to the specified CIDR block. The function returns true.
    - Raw log:

      ```
      cidr_subnet: 192.168.1.0/24
      ip: 192.168.1.100
      ```

    - Transformation rule:

      ```
      e_set("is_belong",ip_cidrmatch(v("cidr_subnet"),v("ip")))
      ```

    - Result:

      ```
      cidr_subnet: 192.168.1.0/24
      ip: 192.168.1.100
      is_belong: true
      ```

  - Example 2: The specified IPv4 address does not belong to the specified CIDR block. The function returns false.
    - Raw log:

      ```
      cidr_subnet: 192.168.1.0/24
      ip: 10.10.1.100
      ```

    - Transformation rule:

      ```
      e_set("is_belong",ip_cidrmatch(v("cidr_subnet"),v("ip")))
      ```

    - Result:

      ```
      cidr_subnet: 192.168.1.0/24
      ip: 10.10.1.100
      is_belong: false
      ```

  - Example 3: The function cannot determine whether the specified IP address belongs to the specified CIDR block and returns unknown.
    - Raw log:

      ```
      cidr_subnet: -11
      ip: 10.10.1.100
      ```

    - Transformation rule:

      ```
      e_set("is_belong",ip_cidrmatch(v("cidr_subnet"),v("ip"),default="unknown"))
      ```

    - Result:

      ```
      cidr_subnet: -11
      ip: 10.10.1.100
      is_belong: unknown
      ```

ip_version

The ip_version function checks whether the version of an IP address is IPv4 or IPv6. If the version of the IP address is IPv4, the function returns IPv4. If the version of the IP address is IPv6, the function returns IPv6.

- Syntax

```
ip_version(ip, default="")
```

- Parameters

Parameter	Type	Required	Description
ip	String	Yes	The IP address.
default	Arbitrary	No	The default value. If the version of the IP address fails to be identified, the value of this parameter is returned. This parameter can be empty.

- Response

  IPv6 or IPv4 is returned.

- Examples
  - Example 1: The specified IP address is an IPv4 address. The function returns IPv4.
    - Raw log:

```
ip: 192.168.1.100
```

    - Transformation rule:

```
e_set("version",ip_version(v("ip")))
```

    - Result:

```
ip: 192.168.1.100
version: IPv4
```

- Example 2: The specified IP address is an IPv6 address. The function returns IPv6.
  - Raw log:

    ```
    ip: ::1
    ```

  - Transformation rule:

    ```
    e_set("version",ip_version(v("ip")))
    ```

  - Result:

    ```
    ip: ::1
    version: IPv6
    ```

ip_type

The ip_type function identifies the type of an IP address and checks whether the type of the IP address is private or public. Valid values: private, reserved, loopback, public, and allocated ripe ncc.

- Syntax

  ```
  ip_type(ip, default="")
  ```

- Parameters

Parameter	Type	Required	Description
ip	String	Yes	The IP address.
default	Arbitrary	No	The default value. If the type of the IP address fails to be identified, the value of this parameter is returned. This parameter can be empty.

- Response

  A value that indicates the type of an IP address is returned. Valid values: private, reserved, loopback, public, and allocated ripe ncc.

- Examples
  - Example 1: Identify the type of the specified IP address. The function returns loopback.
    - Raw log:

      ```
      ip: 127.0.0.1
      ```

    - Transformation rule:

      ```
      e_set("type",ip_type(v("ip")))
      ```

- Result:

```
ip: 127.0.0.1
type: loopback
```

- Example 2: Identify the type of the specified IP address. The function returns private.
  - Raw log:

```
ip: 192.168.1.1
```

  - Transformation rule:

```
e_set("type",ip_type(v("ip")))
```

  - Result:

```
ip: 192.168.1.1
type: private
```

- Example 3: Identify the type of the specified IP address. The function returns public.
  - Raw log:

```
ip: 195.185.1.2
```

  - Transformation rule:

```
e_set("type",ip_type(v("ip")))
```

  - Result:

```
ip: 195.185.1.2
type: public
```

- Example 4: Identify the type of the specified IPv6 address. The function returns loopback.
  - Raw log:

```
ip: ::1
```

  - Transformation rule:

```
e_set("type",ip_type(v("ip")))
```

  - Result:

```
ip: ::1
type: loopback
```

- Example 5: Identify the type of the specified IPv6 address. The function returns allocated ripe ncc.
  - Raw log:

    ```
    ip: 2001:0658:022a:cafe:0200::1
    ```

  - Transformation rule:

    ```
    e_set("type",ip_type(v("ip")))
    ```

  - Result:

    ```
    ip: 2001:0658:022a:cafe:0200::1
    type: allocated ripe ncc
    ```

ip_makenet

The ip_makenet function converts an IP address to a CIDR block.

- Syntax

  ```
  ip_makenet(ip, subnet_mask=None, default="")
  ```

- Parameters

Parameter	Type	Required	Description
ip	String	Yes	The IP address.
subnet_mask	String	Yes	The subnet mask. Example: 255.255.255.0. **Note** If you set the ip parameter to an IP address range, you can set the subnet_mask parameter to an empty string.
default	Arbitrary	No	The default value. If the IP address fails to be converted to a CIDR block, the value of this parameter is returned. This parameter can be empty.

- Response

A CIDR block is returned.

- Examples
  - Example 1: Convert the specified IP address to a CIDR block.
    - Raw log:

      ```
      ip: 192.168.1.0
      ```

    - Transformation rule:

      ```
      e_set("makenet",ip_makenet(v("ip"),"255.255.255.0"))
      ```

    - Result:

      ```
      ip: 192.168.1.0
      makenet: 192.168.1.0/24
      ```

  - Example 2: Convert the specified IP address range to a CIDR block.
    - Raw log:

      ```
      ip: 192.168.1.0-192.168.1.255
      ```

    - Transformation rule:

      ```
      e_set("makenet",ip_makenet(v("ip")))
      ```

    - Result:

      ```
      ip: 192.168.1.0-192.168.1.255
      makenet: 192.168.1.0/24
      ```

  - Example 3: Convert the specified IP address range to a CIDR block.
    - Raw log:

      ```
      ip: 192.168.1.0/255.255.255.0
      ```

    - Transformation rule:

      ```
      e_set("makenet",ip_makenet(v("ip")))
      ```

    - Result:

      ```
      ip: 192.168.1.0/255.255.255.0
      makenet: 192.168.1.0/24
      ```

ip_to_format

The ip_to_format function converts a CIDR block to the format in which the prefix length or netmask of the CIDR block is specified.

- Syntax

  ```
  ip_to_format(cidr_subnet, want_prefix_len=0, default="")
  ```

- Parameters

Parameter	Type	Required	Description
cidr_subnet	String	Yes	The CIDR block. Example: 192.168.1.0/24.
want_prefix_len	Int	No	The format of the output CIDR block. Default value: 0. Valid values: ◦ 0: returns the original CIDR block. ◦ 1: converts the CIDR block to the format in which the prefix length of the CIDR block is specified. ◦ 2: converts the CIDR block to the format in which the netmask of the CIDR block is specified. ◦ 3: converts the CIDR block to an IP address range.
default	Arbitrary	No	The default value. If the CIDR block fails to be converted to the format, the value of this parameter is returned. This parameter can be empty.

- Response

  A CIDR block of the specified format is returned.

- Examples

- Example 1: Return the original CIDR block.
  - Raw log:

```
ip: 192.168.1.0/24
```

  - Transformation rule:

```
e_set("strNormal",ip_to_format(v("ip"),0))
```

  - Result:

```
ip: 192.168.1.0/24
strNormal: 192.168.1.0/24
```

- Example 2: Convert the specified CIDR block to the format in which the prefix length of the CIDR block is specified.
  - Raw log:

```
 ip: 192.168.1.0/24
```

  - Transformation rule:

```
e_set("strNormal",ip_to_format(v("ip"),1))
```

  - Result:

```
ip: 192.168.1.0/24
strNormal: 192.168.1.0/24
```

- Example 3: Convert the specified CIDR block to the format in which the netmask of the CIDR block is specified.
  - Raw log:

```
ip: 192.168.1.0/24
```

  - Transformation rule:

```
e_set("strNormal",ip_to_format(v("ip"),2))
```

  - Result:

```
ip: 192.168.1.0/24
strNormal: 192.168.1.0/255.255.255.0
```

- Example 4: Convert the specified CIDR block to an IP address range.
  - Raw log:

    ```
    ip: 192.168.1.0/24
    ```

  - Transformation rule:

    ```
    e_set("strNormal",ip_to_format(v("ip"),3))
    ```

  - Result:

    ```
    ip: 192.168.1.0/24
    strNormal: 192.168.1.0-192.168.1.255
    ```

ip_overlaps

The ip_overlaps function checks whether two CIDR blocks overlap.

- Syntax

  ```
  ip_overlaps(cidr_subnet, cidr_subnet2, default="")
  ```

- Parameters

Parameter	Type	Required	Description
cidr_subnet	String	Yes	The first CIDR block.
cidr_subnet2	String	Yes	The second CIDR block.
default	Arbitrary	No	The default value. If the function cannot determine whether the CIDR blocks overlap, the value of this parameter is returned. This parameter can be empty.

- Response
  - If the specified CIDR blocks do not overlap, the function returns 0.
  - If the specified CIDR blocks overlap at the end of the blocks, the function returns 1.
  - If the specified CIDR blocks overlap at the start of the blocks, the function returns -1.
- Examples

- Example 1: The specified CIDR blocks do not overlap.
  - Raw log:

```
cidr1: 192.168.0.0/23
cidr2: 192.168.2.0/24
```

  - Transformation rule:

```
e_set("overlaps",ip_overlaps(v("cidr1"),v("cidr2")))
```

  - Result:

```
cidr1: 192.168.0.0/23
cidr2: 192.168.2.0/24
overlaps: 0
```

- Example 2: The specified CIDR blocks overlap at the start of the blocks.
  - Raw log:

```
cidr1: 192.168.1.0/24
cidr2: 192.168.0.0/23
```

  - Transformation rule:

```
e_set("overlaps",ip_overlaps(v("cidr1"),v("cidr2")))
```

  - Result:

```
cidr1: 192.168.1.0/24
cidr2: 192.168.0.0/23
overlaps: -1
```

- Example 3: The specified CIDR blocks overlap at the end of the blocks.
  - Raw log:

```
cidr1: 192.168.0.0/23
cidr2: 192.168.1.0/24
```

  - Transformation rule:

```
e_set("overlaps",ip_overlaps(v("cidr1"),v("cidr2")))
```

  - Result:

```
cidr1: 192.168.0.0/23
cidr2: 192.168.1.0/24
overlaps: 1
```

ip2long

The ip2long function converts an IP address to a value of the long type.

- Syntax

```
ip2long(value,default=0)
```

- Parameters

Parameter	Type	Required	Description
value	String	Yes	The value that you want to convert.
default	String	No	The value that is converted from an invalid IP address. You can use a custom value. Example: 0.

- Response

  The value that is converted from a valid IP address is returned. The value is of the long type.

- Examples
  - Example 1: Convert a valid IP address. This is the default scenario.
    - Raw log:

      ```
      ip: 192.168.0.100
      ```

    - Transformation rule:

      ```
      e_set("long_ip",ip2long(v("ip")))
      ```

    - Result:

      ```
      ip: 192.168.0.100
      long_ip: 167772160
      ```

  - Example 2: Convert an invalid IP address.
    - Raw log:

      ```
      ip: 333.1.1.1
      ```

    - Transformation rule:

      ```
      e_set("long_ip",ip2long(v("ip"), "ignore"))
      ```

    - Result:

      ```
      ip:333.1.1.1
      long_ip:ignore
      ```

long2ip

The long2ip function converts a value of the long type to an IP address.

- Syntax

  ```
  long2ip(value,default="")
  ```

- Parameters

Parameter	Type	Required	Description
value	String	Yes	The value that you want to convert.

Parameter	Type	Required	Description
default	String	No	The empty string that is converted from an invalid value of the long type. You can use a custom string.

- Response

  The IP address that is converted from a valid value of the long type is returned.

- Examples
  - Example 1: Convert a valid value of the long type. This is the default scenario.
    - Raw log:

      ```
      long: 167772160
      ```

    - Transformation rule:

      ```
      e_set("ip",long2ip(v("long")))
      ```

    - Result:

      ```
      long: 167772160
      ip: 192.168.0.100
      ```

  - Example 2: Convert an invalid value of the long type.
    - Raw log:

      ```
      long: 4294967296
      ```

    - Transformation rule:

      ```
      e_set("ip",long2ip(v("long")))
      ```

    - Result:

      ```
      long: 4294967296
      ip:
      ```

  - Example 3: Convert an invalid value of the long type and set the default parameter to a custom string.
    - Raw log:

      ```
      long: 4294967296
      ```

    - Transformation rule:

      ```
      e_set("ip",long2ip(v("long"),default="xxx"))
      ```

    - Result:

      ```
      long: 4294967296
      ip: xxx
      ```

12.6.12. Encoding and decoding functions

This topic describes the syntax and parameters of encoding and decoding functions. This topic also provides examples on how to use the functions.

Functions

Type	Subtype	Function	Description
Encoding and decoding	String	str_encode	Encodes a string by using a specified encoding format.
		str_decode	Decodes a string by using a specified encoding format.
	Base64	base64_encoding	Encodes data by using the Base64 algorithm.
		base64_decoding	Decodes data by using the Base64 algorithm.
	HTML	html_encoding	Encodes data in the HTML format.
		html_decoding	Decodes HTML-encoded data.
	URL	url_encoding	Encodes URL data.
		url_decoding	Decodes URL data.
Compression and decompression	Gzip	gzip_compress	Compresses data and then encodes the data by using the Base64 algorithm.
		gzip_decompress	Decodes data by using the Base64 algorithm and then decompresses the data.
	Zlib	zlib_compress	Compresses data and then encodes the data by using the Base64 algorithm.
		zlib_decompress	Decodes data by using the Base64 algorithm and then decompresses the data.
Encryption and decryption	AES	aes_encrypt	Encrypts data by using the AES algorithm.
		aes_decrypt	Decrypts data by using the AES algorithm.
Hash algorithm	MD5	md5_encoding	Encodes data by using the MD5 algorithm.
	SHA1	sha1_encoding	Encodes data by using the SHA1 algorithm.

str_encode

The str_encode function is used to encode a string by using a specified encoding format.

- Syntax

```
str_encode(value, "utf8", errors="ignore"))
```

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The value that you want to encode.
encoding	String	No	The encoding format. Default value: utf8. ASCII is supported.
errors	String	No	The method that is used to process characters if the characters cannot be recognized based on the encoding format. Valid values: ◦ ignore: The characters are not encoded. This is the default value. ◦ strict: An error is reported, and the data of the log is discarded. ◦ replace: The characters are replaced with question marks (?). ◦ xmlcharrefreplace: The characters are replaced with the related XML characters.

- Response

  An encoded string is returned.

- Examples
  - Example 1

    Raw log:

    ```
    test: asewds
    ```

    Transformation rule:

    ```
    e_set("f1", str_decode(str_encode("hello", "utf8"), "utf8"))
    ```

    Result:

    ```
    test: asewds
    f1: hello
    ```

- Example 2

  Raw log:

  ```
  f2: test test data
  ```

  Transformation rule:

  ```
  e_set("f1", str_encode(v("f2"), "ascii", errors="ignore"))
  ```

  Result:

  ```
  f1:test
  f2:test test data
  ```

- Example 3

  Raw log:

  ```
  f2: test test data
  ```

  Transformation rule:

  ```
  e_set("f1", str_encode(v("f2"), "ascii", errors="strict"))
  ```

  Result:

  ```
  An error is reported during execution.
  ```

- Example 4

  Raw log:

  ```
  f2: test test data
  ```

  Transformation rule:

  ```
  e_set("f1", str_encode(v("f2"), "ascii", errors="replace"))
  ```

  Result:

  ```
  f1:test ????
  f2:test test data
  ```

- Example 5

  Raw log:

  ```
  f2: test test data
  ```

  Transformation rule:

  ```
  e_set("f1", str_encode(v("f2"), "ascii", errors="xmlcharrefreplace"))
  ```

  Result:

  ```
  f1:test &#27979;&#35797;&#25968;&#25454;
  f2:test test data
  ```

str_decode

The str_decode function is used to decode an input value by using a specified encoding format.

- Syntax

```
str_decode(value, "utf8", errors="ignore"))
```

> **Note** The str_decode function can process only the data of the byte data type.

- Parameters

Parameter	Type	Required	Description
value	Arbitrary (automatically converted to the string type)	Yes	The value that you want to decode.
encoding	Arbitrary (automatically converted to the string type)	No	The encoding format. Default value: utf8. ASCII is supported.
errors	Arbitrary (automatically converted to the string type)	No	The method that is used to process characters if the characters cannot be recognized based on the encoding format. Valid values: ◦ ignore: The characters are not encoded. This is the default value. ◦ strict: An error is reported, and the data of the log is discarded. ◦ replace: The characters are replaced with question marks (?). ◦ xmlcharrefreplace: The characters are replaced with the related XML characters.

- Response

  A decoded value is returned.

- Examples

  Raw log:

```
test: asewds
```

Transformation rule:

```
e_set("encoding", str_decode(b'\xe4\xbd\xa0\xe5\xa5\xbd', "utf8",'strict'))
```

Result:

```
test: asewds
encoding: hello
```

base64_encoding

The base64_encoding function is used to encode data by using the Base64 algorithm.

- Syntax

```
base64_encoding(Value,format=None)
```

- Parameters

Parameter	Type	Required	Description
Value	String	Yes	The value that you want to encode.
format	String	No	The Base64 encoding scheme. Valid values: RFC 3548 and RFC 4648. Default value: RFC 3548.

- Response

  A Base64-encoded string is returned.

- Example
  - Raw log entry:

    ```
    str_en : data to be encoded
    ```

  - Transformation rule:

    ```
    e_set("str_base64",base64_encoding(v("str_en")))
    ```

  - Result:

    ```
    str_en : data to be encoded
    str_base64 : ZGF0YSB0byBiZSBlbmNvZGVk
    ```

base64_decoding

The base64_decoding function is used to decode data by using the Base64 algorithm.

- Syntax

```
base64_decoding(Value,format=None)
```

- Parameters

Parameter	Type	Required	Description
Value	String	Yes	The value that you want to decode.
format	String	No	The Base64 decoding scheme. Valid values: RFC 3548 and RFC 4648. Default value: RFC 3548. **Note** The Base64 decoding scheme in RFC 4648 uses an equal sign (=) to convert a decoded value to a multiple of 4 bytes.

- Response

  A Base64-decoded string is returned.

- Example
  - Raw log entry:

    ```
    str_de: ZGF0YSB0byBiZSBlbmNvZGVk
    ```

  - Transformation rule:

    ```
    e_set("str_de_base64",base64_decoding(v("str_de")))
    ```

  - Result:

    ```
    str_de: ZGF0YSB0byBiZSBlbmNvZGVk
    str_de_base64: data to be encoded
    ```

html_encoding

The html_encoding function is used to encode data in the HTML format.

- Syntax

  ```
  html_encoding(Value)
  ```

- Parameters

Parameter	Type	Required	Description
Value	String	Yes	The value that you want to encode.

- Response

  An HTML-encoded string is returned.

- Example

- Raw log entry:

```
str : <img class="size-medium wp-image-113" style="margin-left: 15px;" title="su1" src="http://aliyundoc.com/wp-content/uploads/2008/10/su1-300x194.jpg" alt="" width="300" height="194" />
```

- Transformation rule:

```
e_set("str_html_en",html_encoding(v("str")))
```

- Result:

```
str : <img class="size-medium wp-image-113" style="margin-left: 15px;" title="su1" src="http://aliyundoc.com/wp-content/uploads/2008/10/su1-300x194.jpg" alt="" width="300" height="194" />
str_html_en : &lt;img class=&quot;size-medium wp-image-113&quot; style=&quot;margin-left: 15px;&quot; title=&quot;su1&quot; src=&quot;http://aliyundoc.com/wp-content/uploads/2008/10/su1-300x194.jpg&quot; alt=&quot;&quot; width=&quot;300&quot; height=&quot;194&quot; /&gt;
```

html_decoding

The html_decoding function is used to decode HTML-encoded data.

- Syntax

```
html_decoding(Value)
```

- Parameters

Parameter	Type	Required	Description
Value	String	Yes	The value that you want to decode.

- Response

  An HTML-decoded string is returned.

- Example
  - Raw log entry:

```
str : &lt;img class=&quot;size-medium wp-image-113&quot; style=&quot;margin-left: 15px;&quot; title=&quot;su1&quot; src=&quot;http://aliyundoc.com/wp-content/uploads/2008/10/su1-300x194.jpg&quot; alt=&quot;&quot; width=&quot;300&quot; height=&quot;194&quot; /&gt;
```

  - Transformation rule:

```
e_set("str_html_de",html_decoding(v("str")))
```

- Result:

```
str : &lt;img class=&quot;size-medium wp-image-113&quot; style=&quot;margin-left: 15px;
&quot; title=&quot;su1&quot; src=&quot;http://aliyundoc.com/wp-content/uploads/2008/10/
su1-300x194.jpg&quot; alt=&quot;&quot; width=&quot;300&quot; height=&quot;194&quot; /&g
t;
str_html_de : <img class="size-medium wp-image-113" style="margin-left: 15px;" title="s
u1" src="http://aliyundoc.com/wp-content/uploads/2008/10/su1-300x194.jpg" alt="" width=
"300" height="194" />
```

url_encoding

The url_encoding function is used to encode URL data.

- Syntax

```
url_encoding(Value)
```

- Parameters

Parameter	Type	Required	Description
Value	String	Yes	The value that you want to encode.

- Response

  A URL-encoded string is returned.

- Example
  - Raw log entry:

```
content : https://www.example.org/hello/asdah
```

  - Transformation rule:

```
e_set("url",url_encoding(v("content")))
```

  - Result:

```
content : https://www.example.org/hello/asdah
url: https%3A%2F%www.example.org%2FHello%2Fasdah
```

url_decoding

The url_decoding function is used to decode URL data.

- Syntax

```
url_decoding(Value)
```

- Parameters

Parameter	Type	Required	Description
Value	String	Yes	The value that you want to decode.

- Response

  A URL-decoded string is returned.

- Example
  - Raw log entry:

    ```
    content : https%3A%2F%www.example.org%2FHello%2Fasdah
    ```

  - Transformation rule:

    ```
    e_set("URL",url_decoding(v("content")))
    ```

  - Result:

    ```
    content : https%3A%2F%www.example.org%2FHello%2Fasdah
    URL : https://www.example.org/hello/asdah
    ```

gzip_compress

The gzip_compress function is used to compress data and then encode the data by using the Base64 algorithm.

- Syntax

  ```
  gzip_compress(data, compresslevel=6, to_format="base64", encoding="utf-8")
  ```

- Parameters

Parameter	Type	Required	Description
data	Arbitrary	Yes	The data that you want to compress.
compresslevel	Int	No	The compression level. Valid values: 0 to 9. Default value: 6. ○ 1: Data is compressed at the highest speed but with the lowest compression ratio. ○ 9: Data is compressed at the lowest speed but with the highest compression ratio. ○ 0: Data is not compressed.
to_format	String	No	The encoding format for the compressed data. Only the Base64 algorithm is supported.

Parameter	Type	Required	Description
encoding	String	No	The encoding format for the raw data. Default value: utf-8. For more information about other encoding formats, see Standard encoding formats.

- Response

  Compressed data that is encoded in Base64 is returned.

- Example
  - Raw log entry:

    ```
    content: I always look forward to my holidays whether I travel or stay at home.
    ```

  - Transformation rule:

    ```
    e_set("base64_encode_gzip_compress",gzip_compress(v("content"),to_format="base64"))
    ```

  - Result:

    ```
    content: I always look forward to my holidays whether I travel or stay at home.
    base64_encode_gzip_compress: H4sIAA8JXl4C/xXK0QmAMAwFwFXeBO7RMQKNRExx5kAZDtle/7wbES3rDyR
    snoyQmklgNo1/ztzJN08BAhjzqYGCnNCS/tPR4AcgrnWVGAAAA
    ```

gzip_decompress

The gizp_decompress function is used to decode data by using the Base64 algorithm and then decompress the data.

- Syntax

  ```
  gzip_decompress(data, from_format="base64", encoding="utf-8")
  ```

- Parameters

Parameter	Type	Required	Description
data	Arbitrary	Yes	The data that you want to decompress.
from_format	String	No	The encoding format for the decompressed data. Only the Base64 algorithm is supported.

Parameter	Type	Required	Description
encoding	String	No	The encoding format for the raw data. Default value: utf-8. For more information about other encoding formats, see Standard encoding formats.

- Response

  Decompressed and Base64-decoded data is returned.

- Example
  - Raw log entry:

    ```
    content: H4sIAA8JXl4C/xXK0QmAMAwFwFXeBO7RMQKNREx5kAZDtle/7wbES3rDyRsnoyQmklgNo1/ztzJN08
    BAhjzqYGCnNCS/tPR4AcgrnWVGAAAA
    ```

  - Transformation rule:

    ```
    e_set("gzip_decompress",gzip_decompress(v("content"),from_fmat="base64"))
    ```

  - Result:

    ```
    content: H4sIAA8JXl4C/xXK0QmAMAwFwFXeBO7RMQKNREx5kAZDtle/7wbES3rDyRsnoyQmklgNo1/ztzJN08
    BAhjzqYGCnNCS/tPR4AcgrnWVGAAAA
    gzip_decompress: I always look forward to my holidays whether I travel or stay at home.
    ```

zlib_compress

The zlib_compress function is used to compress data and then encode the data by using the Base64 algorithm.

- Syntax

  ```
  zlib_compress(data, compresslevel=6, to_format="base64", encoding="utf-8")
  ```

- Parameters

Parameter	Type	Required	Description
data	Arbitrary	Yes	The data that you want to compress.

Parameter	Type	Required	Description
compresslevel	Int	No	The compression level. Valid values: 0 to 9. Default value: 6. ○ 1: Data is compressed at the highest speed but with the lowest compression ratio. ○ 9: Data is compressed at the lowest speed but with the highest compression ratio. ○ 0: Data is not compressed.
to_format	String	No	The encoding format for the compressed data. Only the Base64 algorithm is supported.
encoding	String	No	The encoding format for the raw data. Default value: utf-8. For more information about other encoding formats, see Standard encoding formats.

- Response

  Compressed and Base64-encoded data is returned.

- Example
  - Raw log entry:

    ```
    content: I always look forward to my holidays whether I travel or stay at home.
    ```

  - Transformation rule:

    ```
    e_set("zlib_compress", zlib_compress(v("content"), to_format="base64"))
    ```

  - Result:

    ```
    zlib_compress: "eJwVytEJgDAMBcBV3gTu0TECjURMeZAGQ7ZXv+8GxEt6w8kbJ6MkJpJYDaNf87cyTdPAQIY
    86mBgpzQkv7T0eAGNshln"
    content: "I always look forward to my holidays whether I travel or stay at home."
    ```

zlib_decompress

The zlib_decompress function is used to decode data by using the Base64 algorithm and then decompress the data.

- Syntax

```
zlib_decompress(data, from_format="base64", encoding="utf-8")
```

- Parameters

Parameter	Type	Required	Description
data	Arbitrary	Yes	The data that you want to decompress.
from_format	String	No	The encoding format for the decompressed data. Only the Base64 algorithm is supported.
encoding	String	No	The encoding format. Default value: utf-8. For more information about other encoding formats, see Standard encoding formats.

- Response

  Decompressed and Base64-decoded data is returned.

- Example
  - Raw log entry:

    ```
    content: "eJwVytEJgDAMBcBV3gTu0TECjURMeZAGQ7ZXv+8GxEt6w8kbJ6MkJpJYDaNf87cyTdPAQIY86mBgpzQkv7T0eAGNshln"
    ```

  - Transformation rule:

    ```
    e_set("zlib_decompress", zlib_decompress(v("content"), from_format="base64"))
    ```

  - Result:

    ```
    content: "eJwVytEJgDAMBcBV3gTu0TECjURMeZAGQ7ZXv+8GxEt6w8kbJ6MkJpJYDaNf87cyTdPAQIY86mBgpzQkv7T0eAGNshln"
    zlib_decompress: "I always look forward to my holidays whether I travel or stay at home."
    ```

aes_encrypt

The aes_encrypt function is used to encrypt data by using the AES algorithm. The Advanced Encryption Standard (AES) algorithm is the most common symmetric encryption algorithm. To ensure data security, you can use the AES algorithm to encrypt data.

- Syntax

```
aes_encrypt(data,key,mode,pad_style,pad_block,input_format,input_encoding,output_format,iv)
```

- Parameters

Parameter	Type	Required	Description
data	String	Yes	The data that you want to encrypt.
key	String	Yes	The key that you want to use to encrypt data.
mode	String	No	The encryption mode of the AES algorithm. ◦ CBC (default): Cipher Block Chaining ◦ ECB: Electronic Codebook Book ◦ CFB: Cipher Feedback ◦ OFB: Output Feedback ◦ CTR: Counter ◦ OPENPGP
pad_style	String	No	The padding mode. Default value: pkcs7. Valid values: iso7816, x923, and pkcs7.
input_format	String	No	The format of characters. Default value: raw. Valid values: ◦ raw: bytes ◦ hex: hexadecimal ◦ Base64: Base64 encoding format
input_encoding	String	No	The encoding format. This parameter is required only if you set the input_format parameter to raw. The encoding format. Default value: uft-8.
output_format	String	No	The format of characters. Default value: hex. Valid values: ◦ raw: bytes ◦ hex: hexadecimal ◦ Base64: Base64 encoding format
iv	Bytes	No	The offset that is used for encryption.

- Response

  The string that is encrypted from a value by using the AES algorithm is returned.

- Examples

- Example 1
  - Raw log entry:

    ```
    "test": "aliyuntest"
    ```

  - Transformation rule:

    ```
    e_set('result',aes_encrypt(v("test"), "qwertyuiopasdfgd", iv=b"xxywosjdapdiawdk", output_format="base64"))
    ```

  - Result:

    ```
    "result": "gXIqu0cBBtZHQxJBK8GLeA=="
    ```

- Example 2
  - Raw log entry:

    ```
    "test": "aliyuntest"
    ```

  - Transformation rule:

    ```
    e_set('result',aes_encrypt(v("test"), "qwertyuiopasdfgh", iv=b"ywisnjaduaqibdqi", mode="OFB"))
    ```

  - Result:

    ```
    "result":  "5cac3e9e1c42f713dc6d"
    ```

aes_decrypt

The aes_decrypt function is used to decrypt encrypted data by using the AES algorithm.

- Syntax

  ```
  aes_decrypt(data,key,mode,pad_style,input_format,input_encoding,output_format,iv,output_encoding)
  ```

- Parameters

Parameter	Type	Required	Description
data	String	Yes	The data that you want to decrypt.
key	String	Yes	The key that you want to use to decrypt data.

Parameter	Type	Required	Description
mode	String	No	The decryption mode of the AES algorithm. ◦ CBC (default): Cipher Block Chaining ◦ ECB: Electronic Codebook Book ◦ CFB: Cipher Feedback ◦ OFB: Output Feedback ◦ CTR: Counter ◦ OPENPGP
pad_style	String	No	The padding mode. Default value: pkcs7. Valid values: iso7816, x923, and pkcs7.
input_format	String	No	The format of characters. Default value: hex. Valid values: ◦ raw: bytes ◦ hex: hexadecimal ◦ Base64: Base64 encoding format
input_encoding	String	No	The encoding format. This parameter is required only if you set the input_format parameter to raw. The encoding format. Default value: uft-8.
output_format	String	No	The format of characters. Default value: raw. Valid values: ◦ raw: bytes ◦ hex: hexadecimal ◦ Base64: Base64 encoding format
iv	Bytes	No	The offset that is used for decryption.
output_encoding	String	No	The encoding format. Default value: None.

- Response

  The string that is decrypted from a value by using the AES algorithm is returned.

- Examples

- Example 1
  - Raw log entry:

    ```
    "test": "gXIqu0cBBtZHQxJBK8GLeA=="
    ```

  - Transformation rule:

    ```
    e_set('result', aes_decrypt(v("test"), "qwertyuiopasdfgd", iv=b"xxywosjdapdiawdk", input_format="base64"))
    ```

  - Result:

    ```
    "result": "aliyuntest"
    ```

- Example 2
  - Raw log entry:

    ```
    "test": "5cac3e9e1c42f713dc6d"
    ```

  - Transformation rule:

    ```
    e_set('result', aes_decrypt(v("test"), "qwertyuiopasdfgh", iv=b"ywisnjaduaqibdqi", mode="OFB"))
    ```

  - Result:

    ```
    "result": "aliyuntest"
    ```

md5_encoding

The md5_encoding function is used to encode data by using the MD5 algorithm.

- Syntax

  ```
  md5_encoding(Value,format="hex")
  ```

- Parameters

Parameter	Type	Required	Description
Value	String	Yes	The value that you want to encode.
format	String	No	Valid values: binary and hex. Default value: hex.

- Response

  An MD5 hash value is returned.

- Examples

- Example 1
  - Raw log entry:

    ```
    str : GeeksforGeeks
    ```

  - Transformation rule:

    ```
    e_set("str_md5_en",md5_encoding(v("str")))
    ```

  - Result:

    ```
    str : GeeksforGeeks
    str_md5_en : f1e069787ece74531d112559945c6871
    ```

- Example 2
  - Raw log entry:

    ```
    str : GeeksforGeeks
    ```

  - Transformation rule:

    ```
    e_set("str_md5_en",base64_encoding(md5_encoding(v("str"), format="binary")))
    ```

  - Result:

    ```
    str : GeeksforGeeks
    str_md5_en : 8eBpeH7OdFMdESVZlFxocQ==
    ```

sha1_encoding

The sha1_encoding function is used to encode data by using the SHA1 algorithm.

- Syntax

  ```
  sha1_encoding(Value,format=None)
  ```

- Parameters

Parameter	Type	Required	Description
Value	String	Yes	The value that you want to encode.
format	String	No	The SHA algorithm. Valid values: SHA1, SHA224, SHA256, SHA384, and SHA512. Default value: SHA1.

- Response

  A SHA1-encoded string is returned.

- Example
  - Raw log entry:

    ```
    str : GeeksforGeeks
    ```

- Transformation rule:

```
e_set("str_sha1",sha1_encoding(v("str")))
e_set("str_sha512",sha1_encoding(v("str"),format='SHA512'))
e_set("str_sha224",sha1_encoding(v("str"),format='SHA224'))
e_set("str_sha384",sha1_encoding(v("str"),format='SHA384'))
e_set("str_sha256",sha1_encoding(v("str"),format='SHA256'))
```

- Result:

```
str : GeeksforGeeks
str_sha1 : 4175a37afd561152fb60c305d4fa6026b7e79856
str_sha512 : 0d8fb9370a5bf7b892be4865cdf8b658a82209624e33ed71cae353b0df254a75db63d1baa3
5ad99f26f1b399c31f3c666a7fc67ecef3bdcdb7d60e8ada90b722
str_sha224 : 173994f309f727ca939bb185086cd7b36e66141c9e52ba0bdcfd145d
str_sha384 : d1e67b8819b009ec7929933b6fc1928dd64b5df31bcde6381b9d3f90488d253240490460c0
a5a1a873da8236c12ef9b3
str_sha256 : f6071725e7ddeb434fb6b32b8ec4a2b14dd7db0d785347b2fb48f9975126178f
```

12.6.13. List functions

This topic describes the syntax and parameters of list functions. This topic also provides examples on how to use the functions.

Functions

Function	Description
lst_make	Constructs a list.
lst_insert	Inserts elements to a specified position in a list.
lst_append	Adds elements to the end of a list.
lst_delete_at	Deletes the element at a specified position from a list.
lst_reverse	Reverses the order of elements in a list.
op_slice	Obtains elements at specified positions from a list.
lst_get	Obtains an element from a list or a tuple.
op_len	Calculates the number of elements in a list or a tuple.

lst_make

- Syntax

```
lst_make(Value 1, Value 2, ...)
```

- Parameters

Parameter	Type	Required	Description
Value 1	Arbitrary	Yes	The element in the list that you want to construct.
Value 2	Arbitrary	Yes	The element in the list that you want to construct.

- Response

  The constructed list is returned.

- Example

  Transformation rule:

```
e_set("hello", lst_make("k1","k2"))
```

  Result:

```
hello: ["k1","k2"]
```

lst_insert

- Syntax

```
lst_insert(List, Position, Value 1, Value 2, ...)
```

- Parameters

Parameter	Type	Required	Description
List	List	Yes	The list to which you want to insert elements.
Position	Number	Yes	The position at which you want to insert the elements.
Value 1	Arbitrary	Yes	The element that you want to insert.
Value 2	Arbitrary	No	The element that you want to insert.

- Response

  The updated list is returned.

- Example

  Raw log entry:

```
ctx: ["k1","k2"]
```

Transformation rule:

```
e_set("hello", lst_insert(v("ctx"), 0, "k0"))
```

Result:

```
ctx: ["k1","k2"]
hello: ["k0", "k1", "k2"]
```

lst_append

- Syntax

```
lst_append(List, Value 1, Value 2, ...)
```

- Parameters

Parameter	Type	Required	Description
List	List	Yes	The list to which you want to append elements.
Value 1	Arbitrary	Yes	The element that you want to append.
Value 2	Arbitrary	No	The element that you want to append.

- Response

  The updated list is returned.

- Example

  Raw log entry:

```
ctx: ["k1","k2"]
```

  Transformation rule:

```
e_set("hello", lst_append(v(ctx), "k3"))
```

  Result:

```
ctx: ["k1","k2"]
hello: ["k1", "k2", "k3"]
```

lst_delete_at

- Syntax

```
lst_delete_at(List, Position)
```

- Parameters

Parameter	Type	Required	Description
List	list	Yes	The list from which you want to delete an element.
Position	Number	Yes	The position of the element that you want to delete. The position of the first element is 0.

- Response

  The updated list is returned.

- Example

  Raw log entry:

  ```
  ctx: ["k1","k2"]
  ```

  Transformation rule:

  ```
  e_set("hello", lst_delete_at(v("ctx"),1))
  ```

  Result:

  ```
  ctx: ["k1","k2"]
  hello: ["k1"]
  ```

lst_reverse

- Syntax

  ```
  lst_reverse(List)
  ```

- Parameters

Parameter	Type	Required	Description
List	List	Yes	The list in which you want to reverse the order of elements.

- Response

  The updated list is returned.

- Example

  Raw log entry:

  ```
  ctx: ["v1","v2"]
  ```

  Transformation rule:

```
e_set("hello", lst_reverse(v("ctx")))
```

Result:

```
ctx: ["v1","v2"]
hello: ["v2","v1"]
```

lst_get

The lst_get function is used to obtain an element from a list or a tuple.

- Syntax

```
lst_get (List, Element index)
```

- Parameters

Parameter	Type	Required	Description
List	List	Yes	The list from which you want to obtain an element.
Element index	Int	Yes	The index of the first element is 0. For example, the ["a","b","c"] list includes the elements a, b, and c. In this case, the corresponding element indexes are 0, 1, and 2.

- Response

  An element is returned.

- Example

```
"""
Raw log entry:
  ctx: ["v1","v2"]
Result:
  ctx: ["v1","v2"]
  hello: "v2"
"""
e_set("hello", lst_get(v("ctx"),1))
```

12.6.14. Dictionary functions

This topic describes the syntax and parameters of dictionary functions. This topic also provides examples on how to use the functions.

Functions

Function	Description
dct_make	Constructs a dictionary.
dct_update	Updates a dictionary.
dct_delete	Deletes key-value pairs from a dictionary.
dct_keys	Obtains the keys of a dictionary.
dct_values	Obtains the values of a dictionary.
dct_get	Obtains the value that corresponds to a specified key in a dictionary.
op_len	Obtains the number of elements in a dictionary.

dct_make

- Syntax

```
dct_make(Key 1, Value 1, Key 2, Value 2, ...)
```

> **Note** The Key and Value parameters must be specified in pairs.

- Parameters

Parameter	Type	Required	Description
Key	String	Yes	The key in the dictionary that you want to construct.
Value	Arbitrary	Yes	The value in the dictionary that you want to construct.

- Response

  The constructed dictionary is returned.

- Example

  Transformation rule:

```
e_set("hello", dct_make("k1","v1","k2","v2"))
```

  Result:

```
hello: {"k1":"v1","k2":"v2"}
```

dct_update

- Syntax

```
dct_update(Dictionary 1, Dictionary 2)
```

- Parameters

Parameter	Type	Required	Description
Dictionary 1	dict	Yes	The dictionary that you want to update.
Dictionary 2	dict	Yes	The dictionary that is used to update the specified dictionary.

- Response

  The updated dictionary is returned.

- Example

  Raw log entry:

  ```
  ctx: {"k1":"v1","k2":"v2"}
  ```

  Transformation rule:

  ```
  e_set("hello", dct_update(v("ctx"), {"k3": "v3"}))
  ```

  Result:

  ```
  ctx: {"k1":"v1","k2":"v2"}
  hello: {"k1": "v1", "k2": "v2", "k3": "v3"}
  ```

dct_delete

- Syntax

  ```
  dct_delete(Dictionary, Key 1, Key 2, ...)
  ```

- Parameters

Parameter	Type	Required	Description
Dictionary	dict	Yes	The dictionary from which you want to delete specific key-value pairs.
Key 1	String	Yes	The key of the key-value pair that you want to delete from the dictionary.
Key 2	String	No	The key of the key-value pair that you want to delete from the dictionary.

- Response

  The dictionary from which the specified key-value pairs are deleted is returned.

- Example

  Raw log entry:

  ```
  ctx: {"k1":"v1","k2":"v2"}
  ```

  Transformation rule:

  ```
  e_set("hello", dct_delete(v("ctx"), "k2"))
  ```

  Result:

  ```
  ctx: {"k1":"v1","k2":"v2"}
  hello: {"k1":"v1"}
  ```

dct_keys

- Syntax

  ```
  dct_keys(Dictionary)
  ```

- Parameters

Parameter	Type	Required	Description
Dictionary	dict	Yes	The dictionary from which you want to obtain keys.

- Response

  The keys of the dictionary are returned.

- Example

  Raw log entry:

  ```
  ctx: {"k1":"v1","k2":"v2"}
  ```

  Transformation rule:

  ```
  e_set("hello", dct_keys(v("ctx")))
  ```

  Result:

  ```
  ctx: {"k1":"v1","k2":"v2"}
  hello: ["k1","k2"]
  ```

dct_values

- Syntax

  ```
  dct_values(Dictionary)
  ```

- Parameters

Parameter	Type	Required	Description
Dictionary	dict	Yes	The dictionary from which you want to obtain keys.

- Response

  The values of the dictionary are returned.

- Example

  Raw log entry:

```
ctx: {"k1":"v1","k2":"v2"}
```

  Transformation rule:

```
e_set("hello", dct_values(v("ctx")))
```

  Result:

```
ctx: {"k1":"v1","k2":"v2"}
hello: ["v1","v2"]
```

dct_get

- Syntax

```
dct_get(Dictionary,key,default=None)
```

- Parameters

Parameter	Type	Required	Description
Dictionary	dict	Yes	The dictionary from which you want to obtain the value that corresponds to a specified key.
key	Arbitrary	Yes	The key whose value you want to obtain.
default	Arbitrary	No	If the specified key does not exist, the value of the default parameter is returned.

- Response

  The value that corresponds to a specified key in a dictionary is returned.

- Examples

- Example 1

  Raw log entry:

  ```
  ctx: {"k1":"v1","k2":"v2"}
  ```

  Transformation rule:

  ```
  e_set("hello", dct_get(v("ctx"), "k1"))
  ```

  Result:

  ```
  ctx: {"k1":"v1","k2":"v2"}
  hello: v1
  ```

- Example 2: The specified key does not exist and the value of the default parameter is returned.

  Raw log entry:

  ```
  ctx: {"k1":"v1","k2":"v2"}
  ```

  Transformation rule:

  ```
  e_set("hello", dct_get(v("ctx"), "k3",default="123"))
  ```

  Result:

  ```
  ctx: {"k1":"v1","k2":"v2"}
  hello: 123
  ```

12.6.15. Table functions

This topic describes the syntax of table functions and provides parameter descriptions and function examples.

Functions

Type	Function	Description
Text-to-table conversion	tab_parse_csv	Constructs a table from a delimited text file.
Table-to-dictionary conversion	tab_to_dict	Constructs a dictionary from a table.

tab_parse_csv

- Syntax

  ```
  tab_parse_csv(data, sep=',', quote='"', lstrip=True, headers=None, case_insensitive=True)
  ```

- Parameters

Parameter	Type	Required	Description
data	String	Yes	The text data in a delimited format. Generally, the data is in comma-separated values (CSV) format.
sep	String	No	The delimiter used to separate values. By default, the delimiter is a comma (,).
quote	String	No	The character used to enclose a value when the value contains the delimiter. Double quotation marks ("") are used by default.
lstrip	Bool	No	Specifies whether to trim the leading space characters from each value. Default value: True.
headers	String\String List	No	The column headers used to parse data. By default, the system retrieves the headers from the first row of the data. If the first row of the data is not used to store headers, you can pass the headers to the function through this parameter. This parameter can be set to a string or a string list.
case_insensitive	Bool	No	Specifies whether field names are matched in a case-insensitive manner. Default value: True.

- Response

  The constructed table is returned.

- Examples
  - Example 1: Construct a table and map a field in a raw log entry to the table data. The source Logstore must contain fields in the table so that the mapping between the log data and the table data can be established.

    Raw log entry:

    ```
    city:  nanjing
    ```

    Transformation rule:

    ```
    e_table_map(tab_parse_csv("province,city,pop,gdp\nshanghai,shanghai,2000,1000\njiangsu,
    nanjing,800,500"), "city", "province")
    ```

    Result:

    ```
    city:  nanjing
    province:  jiangsu
    ```

- Example 2: Construct a table and map multiple fields in a raw log entry to the table data.

  Raw log entry:

  ```
  city:  nanjing
  province:  jiangsu
  ```

  Transformation rule:

  ```
  e_table_map(tab_parse_csv("province,city,pop,gdp\nshanghai,shanghai,2000,1000\njiangsu,
  nanjing,800,500"), ["province", "city"], ["pop", "gdp"])
  ```

  Result:

  ```
  city:  nanjing
  gdp:  500
  pop:  800
  province:  jiangsu
  ```

- Example 3: Construct a table and map multiple fields in a raw log entry to the table data. In this example, multiple fields in the raw log entry is not the same as the fields in the table. In the parentheses that include the source fields, the first field is a field of the raw log entry and the second field is a field of the table. In the parentheses that include the destination fields, the first field is the field of the table and the second field is a new field.

  Raw log entry:

  ```
  city:  nanjing
  province:  jiangsu
  ```

  Transformation rule:

  ```
  e_table_map(tab_parse_csv("prov,city,pop,gdp\nshanghai,shanghai,2000,1000\njiangsu,nanj
  ing,800,500"), [("province","prov"), "city"], [("pop", "population"), ("gdp", "GDP")])
  ```

  Result:

  ```
  GDP:  500
  city:  nanjing
  population:  800
  province:  jiangsu
  ```

tab_to_dict

- Syntax

  ```
  tab_to_dict(table, key_field, value_field, key_join=",", value_join=",")
  ```

- Parameters

Parameter	Type	Required	Description
table	Table	Yes	The table data from which the dictionary is constructed.

Parameter	Type	Required	Description
key_field	String\String List	Yes	The columns used to construct keys in the dictionary. Separate multiple columns with the character specified by `key_join`.
value_field	String\String List	Yes	The columns used to construct values in the dictionary. Separate multiple columns with the character specified by `value_join`.
key_join	String	No	The string used to separate multiple columns that are used as keys in the dictionary. By default, the string is a comma (,).
value_join	String	No	The string used to separate multiple columns that are used as values in the dictionary. By default, the string is a comma (,).

- Response

  The constructed dictionary is returned.

- Examples
  - Example 1:

    Raw log entry:

    ```
    k1: v1
    city: nj
    ```

    Transformation rule:

    ```
    e_dict_map(tab_to_dict(tab_parse_csv("city,pop\nsh,2000\nnj,800"), "city", "pop"), "city", "popu")
    ```

    Result:

    ```
    k1: v1
    city: nj
    popu: 800
    ```

- Example 2:

  Raw log entry:

```
k1: v1
city: js,nj
```

  Transformation rule:

```
e_dict_map(tab_to_dict(tab_parse_csv("province,city,pop\nsh,sh,2000\njs,nj,800"), ["province", "city"], "pop"), "city", "popu")
```

  Result:

```
k1: v1
city: js,nj
popu: 800
```

12.6.16. Resource functions

This topic describes the syntax and parameters of resource functions. This topic also provides examples on how to use the functions.

Functions

> **Notice** When you call the following resource functions, you must configure the Advanced preview mode to pull the data that you require. For more information about how to configure the Advanced preview mode, see Advanced preview.

Function	Description
res_local	Pulls the values of advanced parameters from the current data transformation task.
res_rds_mysql	Pulls data from a specified table in a database that is created on an ApsaraDB RDS for MySQL instance or obtains the execution result of an SQL statement. The data can be updated at regular intervals.
res_log_logstore_pull	Pulls data from another Logstore when you transform data in a Logstore. You can pull data in a continuous manner.
res_oss_file	Pulls data from an object in a specified Object Storage Service (OSS) bucket. The data can be updated at regular intervals.

res_local

The res_local function is used to pull the values of advanced parameters from the current data transformation task.

- Syntax

```
res_local(param,default=None,type="auto")
```

- Parameters

Parameter	Type	Required	Description
param	String	Yes	The key that is specified in **Advanced Parameter Settings** for the current data transformation task.
default	Arbitrary	No	If the value of the param parameter does not exist, the value of the default parameter is returned. Default value: *None*.
type	String	No	The format of the output data. Valid values: ○ auto: Raw data is converted to a JSON string. If the conversion fails, the raw data is returned. This is the default value. ○ JSON: Raw data is converted to a JSON string. If the conversion fails, the value of the default parameter is returned. ○ raw: Raw data is returned.

- Response

  A JSON or raw string is returned based on the parameter settings.

- Conversions to JSON strings
  - Successful conversions

Raw data	Return value	Data type
1	1	Integer
1.2	1.2	Float
true	True	Boolean
false	False	Boolean
"123"	123	String
null	None	None
["v1", "v2", "v3"]	["v1", "v2", "v3"]	List
["v1", 3, 4.0]	["v1", 3, 4.0]	List
{"v1": 100, "v2": "good"}	{"v1": 100, "v2": "good"}	List
{"v1": {"v11": 100, "v2": 200}, "v3": "good"}	{"v1": {"v11": 100, "v2": 200}, "v3": "good"}	List

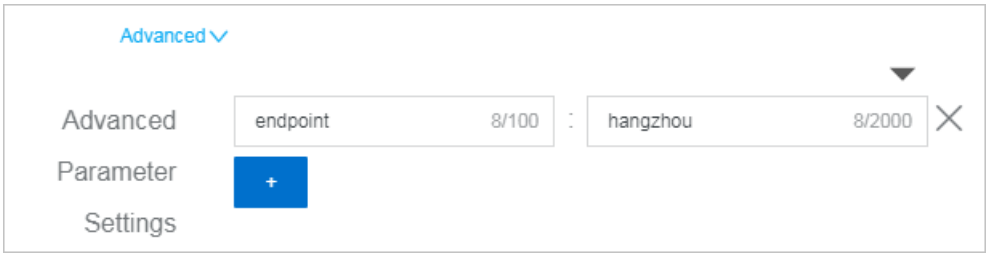
- Failed conversions

  The following table provides some examples of failed conversions. The following raw data fails to be converted to JSON strings, and raw strings are returned.

Raw data	Return value	Description
(1,2,3)	"(1,2,3)"	Tuples are not supported. Lists must be used.
True	"True"	A value of the Boolean data type can only be true or false.
{1: 2, 3: 4}	"{1: 2, 3: 4}"	A dictionary keyword can only be a string.

- Example: Obtain the value of the key that is specified in **Advanced Parameter Settings** and set the local parameter to the obtained value.
  - Advanced Parameter Settings

    In **Advanced Parameter Settings**, the key is *endpoint*, and the value is *hangzhou*.

  - Raw log

    ```
    content: 1
    ```

  - Transformation rule

    ```
    e_set("local", res_local('endpoint'))
    ```

  - Result

    ```
    content: 1
    local: hangzhou
    ```

res_rds_mysql

The res_rds_mysql function is used to pull data from a specified table in a database that is created on an ApsaraDB RDS for MySQL instance or obtains the execution result of an SQL statement. Log Service allows you to pull data by using the following three methods.

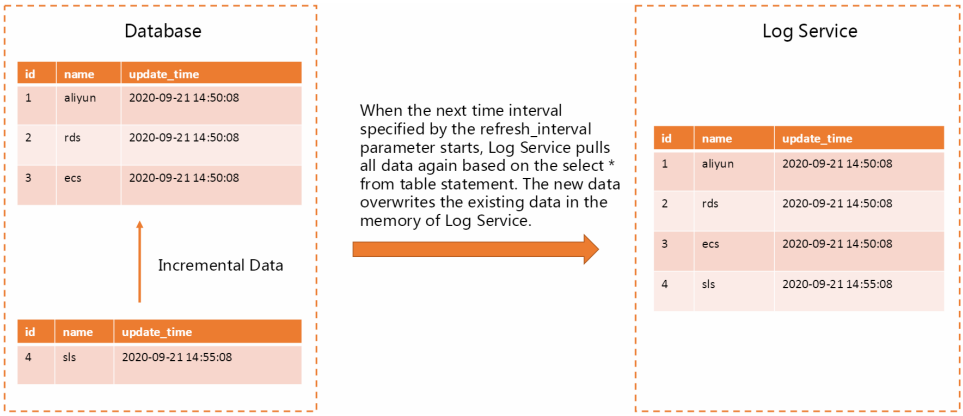
> **Notice**
>
> - If you use the res_rds_mysql function to pull data from a database that is created on an ApsaraDB RDS for MySQL instance, you must create a whitelist on the instance and add `0.0.0.0` to the whitelist. This allows access to the database from all IP addresses. However, this may introduce risks to the database. If you want to add the required IP address to the whitelist, you can submit a ticket. Log Service will provide a more secure method to support the whitelist feature in the future.
> - Log Service can access a database that is created on an ApsaraDB RDS for MySQL instance by using either a public or internal endpoint of the instance. If an internal endpoint is used, you must configure advanced parameters. For more information, see Obtain data from an ApsaraDB RDS for MySQL database over the internal network.

- Pull all data only once

  When you run a data transformation task for the first time, Log Service pulls all data from a specified table and then no longer pulls data. If your database is not updated, we recommend that you use this method.

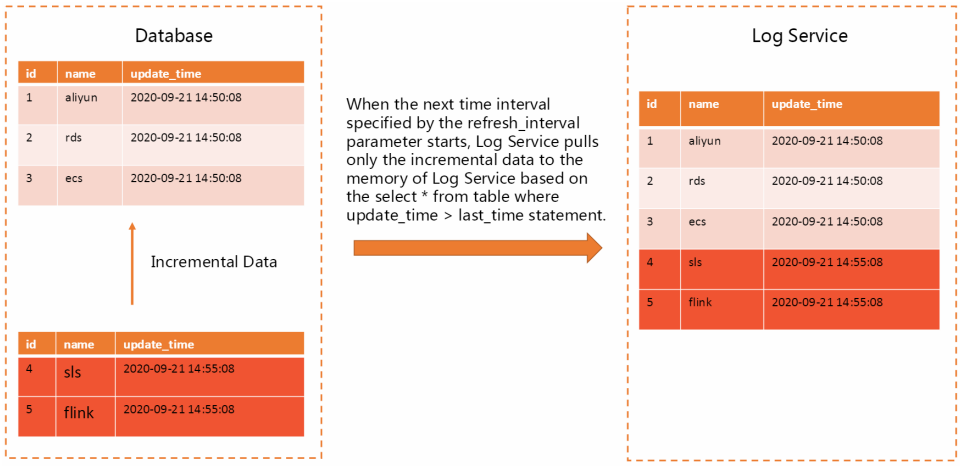
- Pull all data at regular intervals

  When you run a data transformation task, Log Service pulls all data from a specified table at regular intervals. This way, Log Service can synchronize data with your database in a timely manner. However, this method requires a long period of time. If the data volume of your database is less than or equal to 2 GB and the value of the refresh_interval parameter is greater than or equal to 300 seconds, we recommend that you use this method.



- Pull incremental data at regular intervals

When you run a data transformation task, Log Service pulls only incremental data based on the timestamp field of a specified database. If you use this method, Log Service pulls only the newly added data. This method is efficient. You can set the refresh_interval parameter to 1 second to synchronize data within seconds. If the data volume of your database is large, your data is frequently updated, or you require data to be pulled in a timely manner, we recommend that you use this method.



- Syntax

```
res_rds_mysql(address=The address of the database from which data is pulled, username=The username used to connect to the database, password=The password used to connect to the database, database=The name of the database, table=None, sql=None, fields=None, fetch_include_data=None, fetch_exclude_data=None, refresh_interval=0, base_retry_back_off=1, max_retry_back_off=60, primary_keys=None, use_ssl=false, update_time_key=None, deleted_flag_key=None)
```

> **Note** You can also use the res_rds_mysql function to pull data from a database that is created in an AnalyticDB for MySQL or PolarDB for MySQL cluster. In these scenarios, you need to replace only the address, username, password, and name of the database in the transformation rule with actual values.

- Parameters

Parameter	Type	Required	Description
address	String	Yes	The endpoint or IP address of the database to which you want to connect. If the port number is not 3306, specify a value in the **IP address:Port** format. For more information, see View and change the internal and public endpoints and port numbers of an ApsaraDB RDS for SQL Server instance.
username	String	Yes	The username that is used to connect to the database.
password	String	Yes	The password that is used to connect to the database.

Parameter	Type	Required	Description
database	String	Yes	The name of the database to which you want to connect.
table	String	Yes	The name of the table from which data is pulled. If the sql parameter is specified, this parameter is not required.
sql	String	Yes	The SQL SELECT statement that is used to query data. If the table parameter is specified, this parameter is not required.
fields	String list or string alias list	No	The string list or string alias list. If you do not set this parameter, all columns returned by the sql or table parameter are used. For example, if you want to rename the name column in the ["user_id", "province", "city", "name", "age"] list to user_name, you must set the fields parameter to ["user_id", "province", "city", ("name", "user_name"), ("nickname", "nick_name"), "age"]. **Note** If you set the sql, table, and fields parameters at the same time, the SQL statement in the sql parameter is executed. The table and fields parameters do not take effect.
fetch_include_data	String	No	The field whitelist. Logs whose fields match the fetch_include_data parameter are retained. Logs whose fields do not match this parameter are discarded. ○ If you do not set this parameter or set this parameter to None, the field whitelist feature is disabled. ○ If you set this parameter to a specific field and field value, the logs that contain the field and field value are retained.

Parameter	Type	Required	Description
fetch_exclude_data	String	No	The field blacklist. Logs whose fields match the fetch_exclude_data parameter are discarded. Logs whose fields do not match this parameter are retained. ◦ If you do not set this parameter or set this parameter to None, the field blacklist feature is disabled. ◦ If you set this parameter to a specific field and field value, the logs that contain the field and field value are discarded. **Note** If you set both the fetch_include_data and fetch_exclude_data parameters, data is pulled first based on the setting of the fetch_include_data parameter and then based on the setting of the fetch_exclude_data parameter.
refresh_interval	Numeric string or number	No	The interval at which data is pulled from ApsaraDB RDS for MySQL. Unit: seconds. Default value: 0. This value indicates that all data is pulled only once.
base_retry_back_off	Number	No	The interval at which the system attempts to pull data again after a pulling failure. Default value: 1. Unit: seconds.
max_retry_back_off	int	No	The maximum interval between two consecutive retries after a pulling failure. Default value: 60. Unit: seconds. We recommend that you use the default value.
primary_keys	String/List	No	The primary key. If you set this parameter, data in the table is saved to memory as a dictionary in the Key:Value format. A key in the dictionary is the value of the primary_keys parameter. A value in the dictionary is an entire row of data in the table. **Note** ◦ We recommend that you set this parameter if the table contains a large amount of data. ◦ The value of the primary_keys parameter must exist in the fields that are pulled from the table.

Parameter	Type	Required	Description
use_ssl	Boolean	No	Specifies whether to use the SSL protocol to connect to the ApsaraDB RDS for MySQL instance. Default value: false. **Note** If SSL encryption is enabled on the ApsaraDB RDS for MySQL instance, Log Service connects to the instance over SSL. However, the CA certificate of the server is not verified. You cannot use the CA certificate that is provided by the server to establish the connection.
update_time_key	String	No	The time field that is used to pull incremental data. If you do not set this parameter, all data is pulled. For example, the value update_time in update_time_key="update_time" indicates the time field of data update in the database. The time field supports the following data types: datetime, timestamp, integer, float, and decimal. Make sure that the value of the time field increases in chronological order. **Note** Log Service pulls incremental data based on the value of the time field. You must make sure that an index is configured for this field in the table. If the index is not configured, a full table scan is performed. In addition, an error is reported and Log Service cannot pull incremental data.

Parameter	Type	Required	Description
deleted_flag_key	String	No	The data that does not need to be transformed and is discarded when incremental data is pulled. For example, if the value key in update_time_key="key" meets the following condition, the data is parsed as deleted data: ◦ Boolean: true ◦ Datetime and timestamp: not empty ◦ Char and varchar: 1, true, t, yes, and y ◦ Integer: non-zero **Note** ◦ You must set the deleted_flag_key parameter together with the update_time_key parameter. ◦ If you set the update_time_key parameter but do not set the deleted_flag_key parameter, no data is discarded when incremental data is pulled.
connector	String	No	The connector that is used to remotely connect to the database. Valid values: mysql and pgsql. Default value: mysql.

- Response

  A table that contains multiple columns is returned. The columns are defined by the fields parameter.

- Error handling

  If an error occurs when data is pulled, the error is reported, but the data transformation task continues to run. Retries are performed based on the value of the base_retry_back_off parameter. For example, the first retry interval is 1 second and the first retry fails. The second retry interval is twice the length of the first one. The process continues until the interval reaches the value of the max_retry_back_off parameter. If the error persists, retries are performed based on the value of the max_retry_back_off parameter. If a retry succeeds, the retry interval resets to the initial value (1 second).

- Examples
  - Pull all data

- Example 1: Pull data from the test_table table in the test_db database every 300 seconds.

```
res_rds_mysql(
    address="rm-uf6wjk5****.mysql.rds.aliyuncs.com",
    username="test_username",
    password="****",
    database="test_db",
    table="test_table",
    refresh_interval=300,
)
```

- Example 2: Pull data from the test_table table, excluding the data records whose status value is *delete*.

```
res_rds_mysql(
    address="rm-uf6wjk5****.mysql.rds.aliyuncs.com",
    username="test_username",
    password="****",
    database="test_db",
    table="test_table",
    refresh_interval=300,
    fetch_exclude_data="'status':'delete'",
)
```

- Example 3: Pull the data records whose status value is *exit* from the test_table table.

```
res_rds_mysql(
    address="rm-uf6wjk5****.mysql.rds.aliyuncs.com",
    username="test_username",
    password="****",
    database="test_db",
    table="test_table",
    refresh_interval=300,
    fetch_include_data="'status':'exit'",
)
```

- Example 4: Pull the data records whose status value is *exit* from the test_table table, excluding the data records whose name value is *aliyun*.

```
res_rds_mysql(
    address="rm-uf6wjk5****.mysql.rds.aliyuncs.com",
    username="test_username",
    password="****",
    database="test_db",
    table="test_table",
    refresh_interval=300,
    fetch_include_data="'status':'exit'",
    fetch_exclude_data="'name':'aliyun'",
)
```

- Example 5: Use the pgsql connector to connect to a Hologres database and pull data from the test_table table.

```
res_rds_mysql(
    address="hgpostcn-cn-****-cn-hangzhou.hologres.aliyuncs.com:80",
    username="test_username",
    password="****",
    database="aliyun",
    table="test_table",
    connector="pgsql",
)
```

- Example 6: Pull data by using the primary_keys parameter.

  If you set the primary_keys parameter, its value is extracted as a key. The data pulled from the table is saved to memory in the {"10001": {"userid":"10001","city_name":"beijing","city_number":"12345"}} format. In this case, data is pulled at a high speed. If you want to pull a large amount of data in an efficient manner, we recommend that you use this method. If you do not set the primary_keys parameter, the function traverses the table by row. Then, the function pulls data and saves the data to memory in the [{"userid":"10001","city_name":"beijing","city_number":"12345"}] format. In this case, data is pulled at a low speed, but only a small portion of memory is occupied. If you want to pull only a small amount of data, we recommend that you use this method.

  - Table

userid	city_name	city_number
10001	beijing	12345

  - Raw log

```
# Data record 1
userid:10001
gdp:1000
# Data record 2
userid:10002
gdp:800
```

  - Transformation rule

```
e_table_map(res_rds_mysql(address="rm-uf6wjk5****.mysql.rds.aliyuncs.com",username="test_username",password="****",database="test_db",table="test_table", primary_keys="userid"),"userid",["city_name","city_number"])
```

  - Result

```
# Data record 1
userid:10001
gdp:1000
city_name: beijing
city_number:12345
# Data record 2
userid:10002
gdp:800
```

- Pull incremental data
  - Example 1: Pull incremental data.

> **Note** You can pull incremental data only if the following conditions are met:
> - The table has a unique primary key and a time field, such as the item_id and update_time fields.
> - The primary_keys, refresh_interval, and update_time_key parameters are specified.

    - Table

item_id	item_name	price
1001	Orange	10
1002	Apple	12
1003	Mango	16

    - Raw log

```
# Data record 1
item_id: 1001
total: 100
# Data record 2
item_id: 1002
total: 200
# Data record 3
item_id: 1003
total: 300
```

    - Transformation rule

```
e_table_map(res_rds_mysql(address="rm-uf6wjk5****.mysql.rds.aliyuncs.com",username="test_username",password="****", database="test_db",table="test_table",primary_key="item_id",refresh_interval=1,update_time_key="update_time"),"item_id",["item_name","price"])
```

- Result

```
# Data record 1
item_id: 1001
total: 100
item_name: Orange
price:10
# Data record 2
item_id: 1002
total: 200
item_name: Apple
price:12
# Data record 3
item_id: 1003
total: 300
item_name: Mango
price:16
```

- Example 2: Set the deleted_flag_key parameter to discard specified data when incremental data is pulled.
  - Table

item_id	item_name	price	update_time	Is_deleted
1001	Orange	10	1603856138	False
1002	Apple	12	1603856140	False
1003	Mango	16	1603856150	False

  - Raw log

```
# Data record 1
item_id: 1001
total: 100
# Data record 2
item_id: 1002
total: 200
# Data record 3
item_id: 1003
total: 300
```

  - Transformation rule

```
e_table_map(res_rds_mysql(address="rm-uf6wjk5****.mysql.rds.aliyuncs.com",username="test_username",password="****",database="test_db",table="test_table",primary_key="item_id",refresh_interval=1,update_time_key="update_time",deleted_flag_key="is_deleted"),"item_id",["item_name","price"])
```

- Result

  The res_rds_mysq function pulls three data records from the table to the memory of the server on which Log Service runs. These data records are compared with the existing data records in the source Logstore to check whether the records match. If you want to discard the data record whose item_id is 1001, find the data record whose item_id is 1001 in the table and change the value of the Is_deleted field to true. This way, the data record 1001 is discarded the next time when the in-memory dimension table is updated.

```
# Data record 2
item_id: 1002
total: 200
item_name: Apple
price:12
# Data record 3
item_id: 1003
total: 300
item_name: Mango
price:1
```

res_log_logstore_pull

The res_log_logstore_pull function is used to pull data from another Logstore when you transform data in a Logstore.

- Syntax

```
res_log_logstore_pull(endpoint, ak_id, ak_secret, project, logstore, fields, from_time="begin", to_time="end", fetch_include_data=None, fetch_exclude_data=None, primary_keys=None, upsert_data=None, delete_data=None, max_retry_back_off=60, fetch_interval=2, base_retry_back_off=1)
```

- Parameters

Parameter	Type	Required	Description
endpoint	String	Yes	The endpoint. For more information, see Endpoints. By default, an HTTPS endpoint is used. You can also use an HTTP endpoint. In special cases, you can use a port other than port 80 or 443.
ak_id	String	Yes	The AccessKey ID of your Alibaba Cloud account. To ensure data security, we recommend that you set this parameter in **Advanced Parameter Settings**. For more information about how to set the advanced parameters, see Create a data transformation job.
ak_secret	String	Yes	The AccessKey secret of your Alibaba Cloud account. To ensure data security, we recommend that you set this parameter in **Advanced Parameter Settings**. For more information about how to set the advanced parameters, see Create a data transformation job.

Parameter	Type	Required	Description
project	String	Yes	The name of the project from which you want to pull data.
logstore	String	Yes	The name of the Logstore from which you want to pull data.
fields	String list or string alias list	Yes	The string list or string alias list. If a log does not contain a specified field, the value of this field is empty. For example, if you want to rename the name column in the ["user_id", "province", "city", "name", "age"] list to user_name, you must set this parameter to ["user_id", "province", "city", ("name", "user_name"), ("nickname", "nick_name"), "age"].
from_time	String	No	The server time when the first data pulling from the Logstore starts. Default value: *begin*. This value indicates that Log Service starts pulling data from the first log. The following time formats are supported: ◦ UNIX timestamp. ◦ Time string. ◦ Custom string, such as *begin* or *end*. ◦ Expression: the time that is returned by the dt_ function. For example, the function dt_totimestamp(dt_truncate(dt_today(tz="Asia/Shanghai"), day=op_neg(-1))) returns the start time of data pulling in one day before the current day. If the current time is 2019-5-5 10:10:10 (UTC+8), this function returns 2019-5-4 10:10:10 (UTC+8).

Parameter	Type	Required	Description
to_time	String	No	The server time when the first data pulling from the Logstore ends. Default value: *end*. This value indicates that Log Service stops pulling data at the last log. The following time formats are supported: ◦ UNIX timestamp. ◦ Time string. ◦ Custom string, such as *begin* or *end*. ◦ Expression: the time that is returned by the dt_ function. If you do not set this parameter or set this parameter to *None*, data is pulled from the latest logs in a continuous manner. **Note** If you set this parameter to a point in time in the future, only the existing data in the Logstore is pulled. Data that is written after the pull starts is not pulled.
fetch_include_data	String	No	The field whitelist. Logs whose fields match the fetch_include_data parameter are retained. Logs whose fields do not match this parameter are discarded. ◦ If you do not set this parameter or set this parameter to None, the field whitelist feature is disabled. ◦ If you set this parameter to a specific field and field value, the logs that contain the field and field value are retained.

Parameter	Type	Required	Description
fetch_exclude_data	String	No	The field blacklist. Logs whose fields match the fetch_exclude_data parameter are discarded. Logs whose fields do not match this parameter are retained. ◦ If you do not set this parameter or set this parameter to None, the field blacklist feature is disabled. ◦ If you set this parameter to a specific field and field value, the logs that contain the field and field value are discarded. **Note** If you set both the fetch_include_data and fetch_exclude_data parameters, data is pulled first based on the setting of the fetch_include_data parameter and then based on the setting of the fetch_exclude_data parameter.
primary_keys	String list	No	The list of primary key fields that are used to maintain a table. If you change the name of a primary key field by using the fields parameter, you must use the new name to specify a primary key field for this parameter. **Note** ◦ The value of the primary_keys parameter must be a single string that is specified in the value of the fields parameter. ◦ This parameter is valid when only one shard exists in the Logstore from which data is pulled.
fetch_interval	Int	No	The interval between two consecutive data pulling requests when data is pulled in a continuous manner. Default value: 2. Unit: seconds. The value must be at least 1 second.
delete_data	String	No	The operation to delete data from the table. Data records that meet specified conditions and contain the value of `primary_keys` are deleted. For more information, see Query string syntax.

Parameter	Type	Required	Description
base_retry_back_off	Number	No	The interval at which the system attempts to pull data again after a pulling failure. Default value: 1. Unit: seconds.
max_retry_back_off	Int	No	The maximum interval between two consecutive retries after a pulling failure. Default value: 60. Unit: seconds. We recommend that you use the default value.
ttl	Int	No	Data pulling starts when log data is generated and ends ttl seconds after the point in time when the log data is generated. Unit: seconds. Default value: None. This value indicates that all log data is pulled.

- Response

  A table that contains multiple columns is returned.

- Error handling

  If an error occurs when data is pulled, the error is reported, but the data transformation task continues to run. Retries are performed based on the value of the base_retry_back_off parameter. For example, the first retry interval is 1 second and the first retry fails. The second retry interval is twice the length of the first one. The process continues until the interval reaches the value of the max_retry_back_off parameter. If the error persists, retries are performed based on the value of the max_retry_back_off parameter. If a retry succeeds, the retry interval resets to the initial value (1 second).

- Examples
  - In this example, the data of fields key1 and key2 is pulled from the test_logstore Logstore of the test_project project. Data pulling starts when log data is written to the Logstore. Data pulling ends when the data write operation is complete. The data is pulled only once.

```
res_log_logstore_pull(
    "cn-hangzhou.log.aliyuncs.com",
    "LT****Gw",
    "ab****uu",
    "test_project",
    "test_logstore",
    ["key1", "key2"],
    from_time="begin",
    to_time="end",
)
```

- In this example, the data of fields key1 and key2 is pulled from the test_logstore Logstore of the test_project project. Data pulling starts when log data is written to the Logstore. Data pulling ends when the data write operation is complete. The data is pulled every 30 seconds in a continuous manner.

```
res_log_logstore_pull(
    "cn-hangzhou.log.aliyuncs.com",
    "LT****Gw",
    "ab****uu",
    "test_project",
    "test_logstore",
    ["key1", "key2"],
    from_time="begin",
    to_time=None,
    fetch_interval=30,
)
```

- In this example, a blacklist is configured to skip the data records that contain key1:value1 when data is pulled from a Logstore.

```
res_log_logstore_pull(
    "cn-hangzhou.log.aliyuncs.com",
    "LT****Gw",
    "ab****uu",
    "test_project",
    "test_logstore",
    ["key1", "key2"],
    from_time="begin",
    to_time=None,
    fetch_interval=30,
    fetch_exclude_data="key1:value1",
)
```

- In this example, a whitelist is configured to pull data records that contain key1:value1 from a Logstore.

```
res_log_logstore_pull(
    "cn-hangzhou.log.aliyuncs.com",
    "LT****Gw",
    "ab****uu",
    "test_project",
    "test_logstore",
    ["key1", "key2"],
    from_time="begin",
    to_time=None,
    fetch_interval=30,
    fetch_include_data="key1:value1",
)
```

- In this example, the data of fields key1 and key2 is pulled from the test_logstore Logstore of the test_project project. Data pulling starts when log data is generated. Data pulling ends 40,000,000 seconds after the point in time when the log data is generated.

```
res_log_logstore_pull(
    "cn-hangzhou.log.aliyuncs.com",
    "LTAI*****Cajvr",
    "qO0Tp*****jJ9",
    "test_project",
    "test_logstore",
    fields=["key1","key2"],
    ttl="40000000"
)
```

res_oss_file

The res_oss_file function is used to pull data from an object in a specified OSS bucket. The data can be updated at regular intervals.

> **Note** We recommend that you use a Log Service project that resides in the same region as the OSS bucket. This way, the object data in the bucket can be pulled over an internal network of Alibaba Cloud. An internal network is stable and fast.

- Syntax

```
res_oss_file(endpoint, ak_id, ak_key, bucket, file, format='text', change_detect_interval=0, base_retry_back_off=1, max_retry_back_off=60, encoding='utf8', error='ignore')
```

- Parameters

Parameter	Type	Required	Description
endpoint	String	Yes	The endpoint of the OSS bucket. For more information, see Regions and endpoints. By default, an HTTPS endpoint is used. You can also use an HTTP endpoint. In special cases, you can use a port other than port 80 or 443.
ak_id	String	Yes	The AccessKey ID of your Alibaba Cloud account. To ensure data security, we recommend that you set this parameter in **Advanced Parameter Settings**. For more information about how to set the advanced parameters, see Create a data transformation job.
ak_key	String	Yes	The AccessKey secret of your Alibaba Cloud account. To ensure data security, we recommend that you set this parameter in **Advanced Parameter Settings**. For more information about how to set the advanced parameters, see Create a data transformation job.

Parameter	Type	Required	Description
bucket	String	Yes	The name of the OSS bucket from which you want to pull the object data.
file	String	Yes	The path of the object from which you want to pull data. Example: *test/data.txt*. Do not enter a forward slash (/) at the start of the path.
format	String	Yes	The format of the output file. Valid values: ○ Text: text format ○ Binary: byte stream format
change_detect_interval	String	No	The interval at which Log Service pulls the object data from OSS. Unit: seconds. The system checks whether the object is updated when data is pulled. If the object is updated, the incremental data is pulled. Default value: 0. This value indicates that no incremental data is pulled. All data is pulled only once.
base_retry_back_off	Number	No	The interval at which the system attempts to pull data again after a pulling failure. Default value: 1. Unit: seconds.
max_retry_back_off	int	No	The maximum interval between two consecutive retries after a pulling failure. Default value: 60. Unit: seconds. We recommend that you use the default value.
encoding	String	No	The encoding format. If you set the format parameter to *Text*, this parameter is automatically set to *utf8*.
error	String	No	The method that is used to handle errors. This parameter is valid only when the UnicodeError message is reported. Valid values: ○ ignore: The system skips the data whose format is invalid and continues to encode data. ○ xmlcharrefreplace: The system replaces the characters that cannot be encoded with appropriate XML character references. For more information, see Error Handlers.

Parameter	Type	Required	Description
decompress	String	No	Specifies whether to decompress the object. Valid values: ○ None: The object is not decompressed. This is the default value. ○ gzip: The object is decompressed by using gzip.

- Response

  Object data is returned in the byte stream or text format.

- Error handling

  If an error occurs when data is pulled, the error is reported, but the data transformation task continues to run. Retries are performed based on the value of the base_retry_back_off parameter. For example, the first retry interval is 1 second and the first retry fails. The second retry interval is twice the length of the first one. The process continues until the interval reaches the value of the max_retry_back_off parameter. If the error persists, retries are performed based on the value of the max_retry_back_off parameter. If a retry succeeds, the retry interval resets to the initial value (1 second).

- Example 1: Pull JSON data from OSS.
  - JSON data:

```
{
  "users": [
    {
        "name": "user1",
        "login_historys": [
          {
            "date": "2019-10-10 0:0:0",
            "login_ip": "203.0.113.10"
          },
          {
            "date": "2019-10-10 1:0:0",
            "login_ip": "203.0.113.10"
          }
        ]
    },
    {
        "name": "user2",
        "login_historys": [
          {
            "date": "2019-10-11 0:0:0",
            "login_ip": "203.0.113.20"
          },
          {
            "date": "2019-10-11 1:0:0",
            "login_ip": "203.0.113.30"
          },
          {
            "date": "2019-10-11 1:1:0",
            "login_ip": "203.0.113.50"
          }
        ]
    }
  ]
}
```

- Raw log

```
content: 123
```

- Transformation rule

```
e_set("json_parse",json_parse(res_oss_file(endpoint='http://oss-cn-hangzhou.aliyuncs.com',ak_id='LT****Gw',
                                                                    ak_key='ab****uu',
                                                                    bucket='log-etl-staging', file='testjson.json')))
```

- Result

```
content: 123
    prjson_parse:
'{
  "users": [
    {
        "name": "user1",
        "login_historys": [
          {
            "date": "2019-10-10 0:0:0",
            "login_ip": "203.0.113.10"
          },
          {
            "date": "2019-10-10 1:0:0",
            "login_ip": "203.0.113.10"
          }
        ]
    },
    {
        "name": "user2",
        "login_historys": [
          {
            "date": "2019-10-11 0:0:0",
            "login_ip": "203.0.113.20"
          },
          {
            "date": "2019-10-11 1:0:0",
            "login_ip": "203.0.113.30"
          },
          {
            "date": "2019-10-11 1:1:0",
            "login_ip": "203.0.113.50"
          }
        ]
    }
  ]
}'
```

- Example 2: Pull text content from OSS.
  - Text content:

```
Test bytes
```

  - Raw log

```
content: 123
```

- Transformation rule

```
e_set("test_txt",res_oss_file(endpoint='http://oss-cn-hangzhou.aliyuncs.com',ak_id='LT*
***Gw',
                                                              ak_key='ab****uu',
                                                              bucket='log-etl-staging', fi
le='test.txt'))
```

- Result

```
content: 123
test_txt: Test bytes
```

- Example 3: Pull data from an OSS object and decompress the object.
  - Content of the compressed object:

```
Test bytes\nupdate\n123
```

  - Raw log

```
content:123
```

  - Transformation rule

```
e_set(
    "text",
    res_oss_file(
        endpoint="http://oss-cn-hangzhou.aliyuncs.com",
        ak_id="LT****Gw",
        ak_key="ab****uu",
        bucket="log-etl-staging",
        file="test.txt.gz",
        format="binary",
        change_detect_interval=30,
        decompress="gzip",
    ),
)
```

  - Result

```
content:123
text: Test bytes\nupdate\n123
```

- Example 4: Access an object in an OSS bucket whose ACL is public-read-write. No AccessKey pairs are used.
  - Content of the compressed object:

```
Test bytes
```

  - Raw log

```
content:123
```

- Transformation rule

```
e_set(
    "test_txt",
    res_oss_file(
        endpoint="http://oss-cn-hangzhou.aliyuncs.com",
        bucket="log-etl-staging",
        file="test.txt",
    ),
)
```

- Result

```
content: 123
test_txt: Test bytes
```

12.6.17. Parsing functions

This topic describes the syntax and parameters of the functions that are used to parse the User-Agent HTTP header. The topic also provides examples of these functions.

Functions

Function	Description
ua_parse_device	Parses the device information in the User-Agent HTTP header.
ua_parse_os	Parses the operating system information in the User-Agent HTTP header.
ua_parse_agent	Parses the browser information in the User-Agent HTTP header.
ua_parse_all	Parses all the information in the User-Agent HTTP header.

Note The parsing functions of the User-Agent HTTP header delete fields whose parsed values are None. For example, if the device information is parsed into {'brand': None, 'family': 'Other', 'model': None}, the brand and model fields are deleted and the parsing result is {'family': 'Other'}.

ua_parse_device

Parses the device information in the User-Agent HTTP header.

- Syntax

```
ua_parse_device(value)
```

- Parameters

Parameter	Data type	Required	Description

Parameter	Data type	Required	Description
value	String	Yes	The User-Agent HTTP header in the format of a string, for example, ua_parse_agent(v("http_user_agent")).

- Response

  JSON-formatted data is returned.

- Examples
  - Raw log entry:

    ```
    http_user_agent:Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/192.168.0.0 Safari/537.36
    ```

  - Transformation rule:

    ```
    e_set("new_column",ua_parse_device(v("http_user_agent")))
    ```

  - Result:

    ```
    http_user_agent:Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/192.168.0.0 Safari/537.36
    new_column:{'family': 'Other'}
    ```

ua_parse_os

Parses the operating system information in the User-Agent HTTP header.

- Syntax

  ```
  ua_parse_os(value)
  ```

- Parameters

Parameter	Data type	Required	Description
value	String	Yes	The User-Agent HTTP header in the format of a string, for example, ua_parse_os(v("http_user_agent")).

- Response

  JSON-formatted data is returned.

- Examples
  - Raw log entry:

    ```
    http_user_agent:Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/192.168.0.0 Safari/537.36
    ```

- Transformation rule:

```
e_set("new_column",ua_parse_os(v("http_user_agent")))
```

- Result:

```
http_user_agent:Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/192.168.0.0 Safari/537.36
new_column:{'family': 'Mac OS X',
                    'major': '10',
                    'minor': '9',
                    'patch': '4'}
```

ua_parse_agent

Parses the browser information in the User-Agent HTTP header.

- Syntax

```
ua_parse_agent(value)
```

- Parameters

Parameter	Data type	Required	Description
value	String	Yes	The User-Agent HTTP header in the format of a string, for example, ua_parse_all(v("http_user_agent")).

- Response

  JSON-formatted data is returned.

- Examples
  - Raw log entry:

```
http_user_agent:Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/192.168.0.0 Safari/537.36
```

  - Transformation rule:

```
e_set("new_column",ua_parse_agent(v("http_user_agent")))
```

  - Result:

```
http_user_agent:Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/192.168.0.0 Safari/537.36
new_column:{'family': 'Chrome', 'major': '192', 'minor': '168', 'patch': '0'}
```

ua_parse_all

Parses all the information in the User-Agent HTTP header.

- Syntax

```
ua_parse_all(value)
```

- Parameters

Parameter	Data type	Required	Description
value	String	Yes	The User-Agent HTTP header in the format of a string, for example, ua_parse_all(v("http_user_agent")).

- Response

  JSON-formatted data is returned.

- Examples
  - Raw log entry:

    ```
    http_user_agent:Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/192.168.0.0 Safari/537.36
    ```

  - Transformation rule:

    ```
    e_set("new_column",ua_parse_all(v("http_user_agent")))
    ```

  - Result:

    ```
    http_user_agent:Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/192.168.0.0 Safari/537.36
    new_column:{
        'device': {'family': 'Other',},
        'os': {   'family': 'Mac OS X',
                  'major': '10',
                  'minor': '9',
                  'patch': '4'},
        'user_agent': {   'family': 'Chrome',
                          'major': '192',
                          'minor': '168',
                          'patch': '0'}}
    ```

12.7. General reference

12.7.1. Standard encoding formats

This topic lists the standard encoding formats.

Encoding	Alias	Language
ascii	646, US-ASCII	English
big5	BIG5-tw, csBIG5	Traditional Chinese

Encoding	Alias	Language
big5hkscs	BIG5-HKSCS, HKSCS	Traditional Chinese
cp037	IBM037, IBM039	English
cp273	273, IBM273, and csIBM273	German
cp424	EBCDIC-CP-HE, IBM424	Hebrew
cp437	437, IBM437	English
cp500	EBCDIC-CP-BE, EBCDIC-CP-CH, and IBM500	Western Europe
cp720	None	Arabic
cp737	None	Greek
cp775	None	IBM775
cp850	850, IBM850	Western Europe
cp852	852, IBM852	Central and Eastern Europe
cp855	855, IBM855	Bulgarian, Belarusian, Croatian, Russian, and Serbian
cp856	None	Hebrew
cp857	857, IBM857	Turkish
cp858	858, IBM858	Western Europe
cp860	860, IBM860	Portuguese
cp861	861, CP-IS, and IBM861	Icelandic
cp862	862, IBM862	Hebrew
cp863	863, IBM863	Canadian
cp864	IBM864	Arabic
cp865	865, IBM865	Danish, Norwegian
cp866	866, IBM866	Russian
cp869	869, CP-GR, IBM869	Greek
cp874	None	Thai
cp875	None	Greek

Encoding	Alias	Language
cp932	932, MS932, MSKanji, and MS-Kanji	Japanese
cp949	949, MS949, and UHC	Korean
cp950	950, MS950	Traditional Chinese
cp1006	None	Urdu
cp1026	ibm1026	Turkish
cp1125	1125, IBM1125, CP866U, and RUSCII	Ukrainian
cp1140	IBM1140	Western Europe
cp1250	Windows-1250	Central and Eastern Europe
cp1251	Windows-1251	Bulgarian, Belarusian, Croatian, Russian, and Serbian
cp1252	Windows-1252	Western Europe
cp1253	Windows-1253	Greek
cp1254	Windows-1254	Turkish
cp1255	Windows-1255	Hebrew
cp1256	Windows-1256	Arabic
cp1257	Windows-1257	Baltic
cp1258	Windows-1258	Vietnamese
cp65001	None	Windows only: Windows UTF-8 (CP_UTF8)
euc_jp	EUC-JP, UJIS, U-JIS	Japanese
euc_jis_2004	JIS X 0213, EUC-JIS-2004	Japanese
euc_jisx0213	EUCJISX0213	Japanese
euc_kr	EUC-KR, Korean, KSC5601, KS C 5601, KS C 5601-1987, KSX1001, and KS X 1001	Korean
gb2312	Chinese, csISO58GB231280, EUC-CN, EUCCN, EUGB2312-CN, GB2312-1980, GB2312-80, and ISO-IR-58	Simplified Chinese

Encoding	Alias	Language
gbk	936, CP936, MS936	Unified Chinese
gb18030	GB18030-2000	Unified Chinese
hz	HZGB, HZ-GB, and HZ-GB-2312	Simplified Chinese
iso2022_jp	csISO2022JP, ISO2022JP, and ISO-2022-JP	Japanese
iso2022_jp_1	ISO2022JP-1, ISO-2022-JP-1	Japanese
iso2022_jp_2	ISO2022JP-2, ISO-2022-JP-2	Japanese, Korean, Simplified Chinese, Western Europe, and Greek
iso2022_jp_2004	ISO2022JP-2004, ISO-2022-JP-2004	Japanese
iso2022_jp_3	ISO2022JP-3, ISO-2022-JP-3	Japanese
iso2022_jp_ext	ISO2022JP-EXT, ISO-2022-JP-EXT	Japanese
iso2022_kr	csISO2022KR, ISO2022KR, and ISO-2022-KR	Korean
latin_1	ISO-8859-1, ISO8859-1, 8859, CP819, Latin, Latin1, and L1	Western Europe
iso8859_2	ISO-8859-2, Latin2, and L2	Central and Eastern Europe
iso8859_3	ISO-8859-3, Latin3, and L3	Esperanto, Maltese
iso8859_4	ISO-8859-4, Latin4, and L4	Baltic
iso8859_5	ISO-8859-5, Cyrillic	Bulgarian, Belarusian, Croatian, Russian, and Serbian
iso8859_6	ISO-8859-6, Arabic	Arabic
iso8859_7	ISO-8859-7, Greek, Greek8	Greek
iso8859_8	ISO-8859-8, Hebrew	Hebrew
iso8859_9	ISO-8859-9, Latin5, and L5	Turkish
iso8859_10	ISO-8859-10, Latin6, and L6	Nordic
iso8859_11	ISO-8859-11, Thai	Thai
iso8859_13	ISO-8859-13, Latin7, and L7	Baltic
iso8859_14	ISO-8859-14, Latin8, and L8	Celtic

Encoding	Alias	Language
iso8859_15	ISO-8859-15, Latin9, and L9	Western Europe
iso8859_16	ISO-8859-16, Latin10, and L10	Southeast Europe
johab	CP1361, MS1361	Korean
koi8_r	None	Russian
koi8_t	None	Tajik
koi8_u	None	Ukrainian
kz1048	KZ_1048, STRK1048_2002, and RK1048	Kazakh
mac_cyrillic	maccyrillic	Bulgarian, Belarusian, Croatian, Russian, and Serbian
mac_greek	macgreek	Greek
mac_iceland	maciceland	Icelandic
mac_latin2	MacLatin2, MacCentralEurope	Central and Eastern Europe
mac_roman	MacRoman, MacIntosh	Western Europe
mac_turkish	Macturkish	Turkish
ptcp154	csPTCP154, PT154, CP154, Cyrillic-Asian	Kazakh
shift_jis	csShiftJIS, ShiftJIS, SJIS, and S JIS	Japanese
shift_jis_2004	ShiftJIS2004, SJIS 2004, and SJIS2004	Japanese
shift_jisx0213	ShiftJISX0213, SJISX0213, and S JISX0213	Japanese
utf_32	U32, UTF32	All languages
utf_32_be	UTF-32BE	All languages
utf_32_le	UTF-32LE	All languages
utf_16	U16, utf16	All languages
utf_16_be	UTF-16BE	All languages
utf_16_le	UTF-16LE	All languages
utf_7	U7, Unicode-1-1-UTF-7	All languages

Encoding	Alias	Language
utf_8	U8, UTF, and UTF8	All languages
utf_8_sig	None	All languages

12.7.2. Query string syntax

Query strings are used in the domain-specific language (DSL) for Log Service to filter log data in an efficient manner and simplify condition matching. This topic describes the rules for specifying query strings.

Functions

The following table describes the functions that use query strings.

Category	Function	Scenario
Event check functions	e_search	Query strings are used to check whether the value of a field in an event meets specified conditions.
Resource functions	res_log_logstore_pull	Query strings are used to configure a field blacklist or a field whitelist to filter data from a Logstore and return a table.
	res_rds_mysql	Query strings are used to configure a field blacklist or a field whitelist to filter data from a specified table of an ApsaraDB RDS for MySQL database and return a table.
Event mapping functions	e_search_table_map and e_search_dict_map	Query strings are used to match key-value pairs in a dictionary.

Features

The following table lists the search features that support field search and full-text search.

Feature	Field search	Full-text search
Search for substrings	Supported	Supported
Search for strings by using wildcard characters, which support asterisks (`*`) and question marks (?)	Supported	Supported
Exact match	Supported	Not supported
Search for strings by using regular expressions	Supported	Not supported
Search for strings by comparing numeric ranges	Supported	Not supported

Feature	Field search	Full-text search
Search for strings by comparing numeric values	Supported	Not supported
Search for strings by using logical operators (AND, OR, and NOT), or a combination of these operators	Supported	Supported

Escape special characters

Special characters, such as asterisks (*) and backslashes (\), must be escaped in query strings.

- Escape special characters in a field name

  Field names cannot be enclosed in double quotation marks (""). Special characters in a field name must be escaped by using backslashes (\). Examples:

  - `\*\(1+1\)\?: abc`. Special characters are escaped by using backslashes (\).
  - `__tag__\:__container_name__: abc`. Special characters are escaped by using backslashes (\).
  - `"content": abc`. In this example, the field name is invalid. The field name cannot be enclosed in double quotation marks ("").

- Escape special characters in a field value
  - To query a field value that contains special characters such as quotation marks (") and backslashes (\), you must escape the special characters by using backslashes (\). Example: `content: "abc\"xy\\z"`.

    Note A field value must be enclosed in double quotation marks (""). You can use single quotation marks('') to enclose the string and double quotation marks ("") to enclose the field value. For example, `e_search("domain: '/url/test.jsp'")` is invalid, and `e_search('domain: "/url/test.jsp"')` is valid.

  - To query a field value that contains special characters such as asterisks (*) and question marks (?), you must escape the special characters by using backslashes (\). If you do not escape the special characters by using backslashes (\), the special characters are used as wildcard characters for matching.
  - To query a field value that contains only letters, digits, underscores (_), hyphens (-), asterisks (*), and question marks (?), you do not need to enclose the field value in double quotation marks (""). When other characters are used, you must enclose the field value in double quotation marks (""). Examples:
    - `status: "\*\?()[]:="`. The field value is enclosed in double quotation marks (""). The asterisk (*) and question mark (?) are escaped by using backslashes (\). Characters other than the asterisk (*) and question mark (?) are not escaped in the field value.
    - `content: ()[]:=`. The field value is invalid. The field value must be enclosed in double quotation marks ("").
    - `status: active\*test` and `status: active\?test`. The field values contain only letters, an asterisk (*) and a question mark (?). The field values do not need to be enclosed in double quotation marks (""). The asterisk (*) and question mark (?) in the field values are escaped by using backslashes (\).

Search for substrings

- Full-text search

  Search for substrings in all fields.

  - Syntax

    ```
    e_search('substring')
    ```

  - Examples
    - `e_search('"error"')` : searches for a substring.
    - `e_search('"active error"')` : searches for a substring that contains a space.
    - `e_search('active error')` : searches for multiple substrings. The logical operator OR is used by default.

- Field search

  Search for substrings in specific fields.

  - Syntax

    ```
    e_search('...')
    ```

  - Examples
    - `e_search('status: active')` : searches for a substring.
    - `e_search('author: "john smith"')` : searches for a substring that contains a space.

    > **Note** `e_search('field: active error')` : searches for active in the field field or searches for error in all fields. In this example, the query string is equivalent to `field:active OR "error"` .

Search for strings by using wildcard characters

An asterisk (*) specifies zero or multiple characters. A question mark (?) specifies one character or one wide character.

- Full-text search

  Search for substrings in all fields.

  - Syntax

    ```
    e_search('substring')
    ```

- Examples
  - `e_search('active*test')`. The asterisk (*) is used to match zero or multiple characters. The query string does not need to be enclosed in double quotation marks ("") because the query string contains only letters and an asterisk (*).
  - `e_search('*error occurs')`. The asterisk (*) is used to match zero or multiple characters. For example, the `error occurs` and `critical error occurs` strings can be matched.
  - `e_search('active?good')`. The question mark (?) is used to match one character. The query string does not need to be enclosed in double quotation marks ("") because the query string contains only letters and a question mark (?).
  - `e_search('ac*tive?good')`. The query string is used to perform an exact match by using an asterisk (*) and a question mark (?).
  - `e_search('ac*tive??go*od')`. The query string is used to perform an exact match by using multiple asterisks (*) and question marks (?).

- Field search

  Search for substrings in specific fields.

  - Syntax

    ```
    e_search('field name: substring')
    ```

  - Examples
    - `e_search('status: active*test')`. The asterisk (*) is used to match zero or multiple characters.
    - `e_search('status: active?good')`: The question mark (?) is used to match one character.

Exact match

In exact match, the entire field value is matched.

- Syntax

  ```
  e_search('field name==string that must be exactly matched')
  ```

- Examples
  - `e_search('author== "john smith"')`. The value of the author field must be john smith.
  - `e_search('status== ac*tive?good')`. The query string contains wildcard characters and is used for exact match.

Search for strings by using regular expressions

Regular expressions are more efficient than wildcard characters in matching.

- Syntax

  ```
  e_search('field name~=regular expression')
  ```

> **Note**
> - Regular expressions may contain backslashes (\). We recommend that you use `r` to prevent the system from escaping the backslashes (\).
> - By default, Log Service performs fuzzy match. To enable exact match, you must specify a regular expression that includes a caret (`^`) as a prefix and a dollar sign (`$`) as a suffix.

- Examples
  - `e_search('status~= "\d+"')` . The value of the status field contains digits.
  - `e_search('status~= "^\d+$"')` . The value of the status field is a number.

Search for strings by comparing numeric values or numeric ranges

You can search for field values by comparing field values with specified numeric values or numeric ranges.

- Numeric value comparison

  You can compare field values with specified numeric values by using the following operators. The operators are greater-than (`>`), greater-than-or-equal-to (`>=`), equal-to (`=`), less-than (`<`), and less-than-or-equal-to (`<=`).

```
e_search('age >= 18')  #  >=18
e_search('age > 18')   #  > 18
e_search('age = 18')   #  = 18
e_search('age <= 18')  #  <=18
e_search('age < 18')   #  < 18
```

- Numeric range comparison

  You can search for field values that are within a closed interval. An asterisk (*) can be used to specify an infinite interval.

```
e_search('count: [100, 200]') # >=100 and  <=200
e_search('count: [*, 200]')   # <=200
e_search('count: [200, *]')   # >=200
```

Search for strings by judging logical relationships

Logical operators can be used among multiple search conditions. Parentheses `()` are used to nest search conditions.

Logical operator	Keyword
AND	`and` , `AND` , and `&&` . The keywords are not case-sensitive.
OR	`or` and `OR` . The keywords are not case-sensitive.
NOT	`not` , `NOT` , and `!` . The keywords are not case-sensitive.

Examples:

```
e_search('abc OR xyz')    # The logical operator is not case-sensitive.
e_search('abc and (xyz or zzz)')
e_search('abc and not (xyz and not zzz)')
e_search('abc && xyz')    # and
e_search('abc || xyz')    # or
e_search('abc || !xyz')   # or not
```

Local operators can also be used to match substrings.

```
e_search('field: (abc OR xyz)')      # The field value contains abc or xyz.
e_search('field: (abc OR not xyz)')  # The field value contains abc or does not contain xyz
.
e_search('field: (abc && !xyz)')     # The field value contains abc and does not contain xy
z.
```

Field check

You can use query strings to check fields.

- `e_search('field: *')` : checks whether a field exists.
- `e_search('not field:*')` : checks whether a field does not exist.
- `e_search('not field:""')` : checks whether a field does not exist.
- `e_search('field: "?"')` : checks whether a field exists and whether the field is not empty.
- `e_search('field==""')` : checks whether a field exists and whether the field is empty.
- `e_search('field~=".+"')` : checks whether a field exists and whether the field is not empty.
- `e_search('not field~=".+"')` : checks whether a field does not exist or whether the field is empty.
- `e_search('not field==""')` : checks whether a field does not exist or whether the field is not empty.

12.7.3. Field extraction modes

This topic describes the parameters of field extraction modes in different functions.

Related functions

The following table lists the functions that use the mode parameter.

Category	Function	Default value of mode
Field processing functions	e_set	overwrite
Value extraction functions	e_regex	fill-auto
	e_json	fill-auto
	e_kv	fill-auto
	e_csv, e_psv, and e_tsv	fill-auto

Category	Function	Default value of mode
	e_kv_delimit	fill-auto
	e_anchor	overwrite
	e_syslogrfc	overwrite
Mapping and enrichment functions	e_dict_map	fill-auto
	e_table_map	fill-auto
	e_search_dict_map	overwrite
	e_search_table_map	fill-auto

Field check and overwrite modes

The following table describes the values of the mode parameter.

Value	Description
fill	Sets the destination field if the field does not exist or the field value is an empty string.
fill-auto	Sets the destination field if the new value is not an empty string and the destination field does not exist or its value is an empty string.
add	Sets the destination field if the field does not exist.
add-auto	Sets the destination field if the new value is not an empty string and the destination field does not exist.
overwrite	Always sets the destination field.
overwrite-auto	Sets the destination field if the new value is not an empty string.

The following examples show how these modes work.

```
Raw log entry:
  a:         # An empty string
  b: 100
```

Mode	Example	Result
add	`e_set("c", "123", mode='add')`	The c field is added as `c: 123` to the raw log entry.
	`e_set("c", "", mode='add')`	The c field is added as `c:` to the raw log entry.
	`e_set("a", "123", mode='add')`	The a field remains `a:`.

Mode	Example	Result
add-auto	`e_set("c", "", mode='add-auto')`	The `c` field is not added to the raw log entry.
fill	`e_set("c", "123", mode='fill')`	The c field is added as `c: 123` to the raw log entry.
	`e_set("c", "", mode='fill')`	The c field is added as `c:` to the raw log entry.
	`e_set("a", "123", mode='fill')`	The a field is modified to `a: 123`.
	`e_set("b", "123", mode='fill')`	The b field remains `b: 100`.
fill-auto	`e_set("c", "", mode='fill-auto')`	The `c` field is not added to the raw log entry.
overwrite	`e_set("c", "123", mode='overwrite')`	The c field is added as `c: 123` to the raw log entry.
	`e_set("c", "", mode='overwrite')`	The c field is added as `c:` to the raw log entry.
	`e_set("b", "200", mode='overwrite')`	The b field is modified to `b: 200`.
	`e_set("b", "", mode='overwrite')`	The b field is modified to `b:`.
overwrite-auto	`e_set("b", "", mode='overwrite-auto')`	The b field remains `b: 100`.

Constraints on field name extraction

The constraints apply to functions such as e_json, e_kv, e_kv_delimit, and e_regex.

Only the fields whose names meet the constraints can be extracted. The fields that do not meet the constraints are discarded. The regular expression `u'_*[\u4e00-\u9fa5\u0800-\u4e00a-zA-Z][\u4e00-\u9fa5\u0800-\u4e00\\w\\.\\-]*'` is not supported. For example, the fields `123=abc __1__:100 1k=200 {"123": "456"}` will be discarded.

The default constraints are used in the following example.

- Raw log entry:

```
data: {"k1": 100, "k2": {"k3": 200, "k4": {"k5": 300} } }
```

- Transformation rule:

```
e_json("data", fmt='parent', sep="@", prefix="__", suffix="__",include_node=r"[\u4e00-\u9fa5\u0800-\u4e00a-zA-Z][\w\-\.]*", mode='fill-auto' )
```

- Result:

```
data: {"k1": 100, "k2": {"k3": 200, "k4": {"k5": 300} } }
data@__k1__: 100
k2@__k3__: 200
k4@__k5__: 300
```

12.7.4. Regular expressions

This topic describes the matching modes of regular expressions, how to escape special characters in regular expressions, and the group concept of regular expressions.

Full match

If a regular expression matches an entire string, the string fully matches the regular expression. For example, the `1234` string fully matches the `\d+` regular expression. However, the `abc123` string partially matches the `\d+` regular expression.

Some functions use regular expressions in partial match mode. You can enclose a regular expression in a caret (`^`) and a dollar sign (`$`) in the format of `^Regular expression$` to forcibly enable full match. For more information about the regular expression syntax, see Regular expression operations.

The following table lists the matching modes for different functions.

Type	Function	Matching mode
Global processing functions	e_regex	Partial match
	e_keep_fields	Full match
	e_drop_fields	Full match
	e_rename	Full match
	e_kv	Partial match
	e_search_dict_map	Partial match
	e_search_table_map	Partial match
Expression functions	e_match	The matching mode is controlled by a parameter and is full match by default.
	e_search	Partial match
	regex_select	Partial match
	regex_findall	Partial match
	regex_match	The matching mode is controlled by a parameter and is partial match by default.
	regex_replace	Partial match

Type	Function	Matching mode
	regex_split	Partial match

Matching mode examples:

- `reg_match("abc123", r"\d+")`: The string matches the regular expression. The matching mode is partial match, which is the default mode.
- `reg_match("abc123", r"\d+", full=True)`: The string does not match the regular expression. The matching mode is full match.
- `reg_match("abc123", r"^\d+$")`: The string does not match the regular expression. The matching mode is full match based on the syntax of the regular expression.
- `e_search(r'status~="\d+"')`: Whether the value of the `status` field matches the regular expression depends on the actual value. The matching mode is partial match based on the syntax of the regular expression.
- `e_search(r'status~="^\d+$"')`: Whether the value of the `status` field matches the regular expression depends on the actual value. The matching mode is full match based on the syntax of the regular expression.

Escape special characters

Regular expressions may contain special characters. You must escape these characters if you want to use their literal meanings.

- Escape special characters by using backslashes (`\`).
- Escape special characters by using the `str_regex_escape` function.

  For example, the `e_drop_fields(str_regex_escape("abc.test")` function drops the `abc.test` field. The `e_drop_fields("abc.test")` function drops fields whose names match `abc? test`, where the question mark (`?`) represents any character.

Group

You can use parentheses `()` to enclose a subexpression in a regular expression to create a group. The group can be repeatedly referenced. The following example shows a regular expression without a group and a regular expression with a group:

```
"""
Raw log entry:
SourceIP: 192.0.2.1
Result:
SourceIP: 192.0.2.1
ip: 192.0.2.1
"""
# The regular expression without a group:
e_regex("SourceIP",r"\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}","ip")
# The regular expression with a group:
e_regex("SourceIP", "\d{1,3}(.\d{1,3}){3}", "ip")
```

Capturing group

The text content that matches a capturing group is cached in the memory. The cached text content can be reused by using a backreference. A group is a capturing group if the subexpression in parentheses () does not start with a question mark followed by a colon (`?:`).

By default, all capturing groups are numbered from left to right based on an opening parenthesis. The first group is numbered 1, the second group is numbered 2, and so on. For example:

```
(\d{4})-(\d{2}-(\d{2}))
1     1 2      3     32
```

If a regular expression contains both common capturing groups and named capturing groups, the named capturing groups are numbered after the common capturing groups.

You can directly reference the custom name of a capturing group in regular expressions or programs.

Non-capturing group

The text content that matches a non-capturing group is not cached in the memory. A group is a non-capturing group if the subexpression in parentheses () starts with a question mark followed by a colon (`?:`).

For example, the regular expression `pro(gram|ject)` matches `program` and `project` . If you do not want to cache the matched content in the memory, you can write the regular expression as `pro(?:gram|ject)` . Non-capturing groups simplify the matching process and occupy less memory.

> **Note** `(?:x)` indicates that content is matched with `x` but the matched content is not cached. You can define a subexpression in the (?:x) format and use it with operators in a regular expression.

12.7.5. Grok patterns

Grok is a tool that combines multiple predefined regular expressions to match and split text and map the text segments to keys. Grok can be used to process log data. This topic describes the Grok patterns and provides several examples of basic syntax.

The following table lists the Grok patterns.

> **Note** Named parameters cannot be used in some Grok patterns and their combinations, such as SYSLOGBASE, COMMONAPACHELOG, COMBINEDAPACHELOG, HTTPD20_ERRORLOG, HTTPD24_ERRORLOG, and HTTPD_ERRORLOG.

Type	Pattern	Description
	EXTRACTJSON	Matches JSON data.
	CHINAID	Matches the numbers of identity cards of Chinese residents.
	USERNAME	Matches content that contains letters, digits, and `. _-` .

Type	Pattern	Description
Common patterns	USER	Matches content that contains letters, digits, and `. _-` .
	EMAILLOCALPART	Matches the characters before the at sign (@) in an email address. For example, in the email address 123456@alibaba.com, the matched content is 123456.
	EMAILADDRESS	Matches email addresses.
	HTTPDUSER	Matches email addresses or usernames.
	INT	Matches integers.
	BASE10NUM	Matches decimal numbers.
	NUMBER	Matches numbers.
	BASE16NUM	Matches hexadecimal numbers.
	BASE16FLOAT	Matches hexadecimal floating-point numbers.
	POSINT	Matches positive integers.
	NONNEGINT	Matches non-negative integers.
	WORD	Matches letters, digits, and underscores (_).
	NOTSPACE	Matches characters that are not spaces.
	SPACE	Matches spaces.
	DATA	Matches line feeds.
	GREEDYDATA	Matches zero or multiple characters that are not line feeds.
	QUOTEDSTRING	Matches quoted content. For example, in the `I am "Iron Man"` string, the matched content is `Iron Man` .
	UUID	Matches universally unique identifiers (UUIDs).
	MAC	Matches MAC addresses.
	CISCOMAC	Matches Cisco MAC addresses.
	WINDOWSMAC	Matches Windows MAC addresses.
	COMMONMAC	Matches common MAC addresses.
	IPV6	Matches IPv6 addresses.

Type	Pattern	Description
Networking	IPV4	Matches IPv4 addresses.
	IP	Matches IPv6 or IPv4 addresses.
	HOSTNAME	Matches hostnames.
	IPORHOST	Matches IP addresses or hostnames.
	HOSTPORT	Matches IP addresses, hostnames, or positive integers.
Paths	PATH	Matches UNIX paths or Windows paths.
	UNIXPATH	Matches UNIX paths.
	WINPATH	Matches Windows paths.
	URIPROTO	Matches URI schemes. For example, in `http://hostname.domain.tld/_astats?application=&inf.name=eth0`, the matched content is `http`.
	TTY	Matches tty paths.
	URIHOST	Matches IP addresses, hostnames, or positive integers. For example, in `http://hostname.domain.tld/_astats?application=&inf.name=eth0`, the matched content is `hostname.domain.tld`.
	URI	Matches URIs.
Date	MONTH	Matches months that are in the numeric, abbreviated, or full-name format.
	MONTHNUM	Matches months that are in the numeric format.
	MONTHDAY	Matches days in a month.
	DAY	Matches weekdays that are in the abbreviated or full-name format.
	YEAR	Matches years.
	HOUR	Matches hours.
	MINUTE	Matches minutes.
	SECOND	Matches seconds.
	TIME	Matches time.

Type	Pattern	Description
Time	DATE_US	Matches dates in the Month-Day-Year or Month/Day/Year format.
	DATE_EU	Matches dates in the Day-Month-Year, Day/Month/Year or Day.Month.Year format.
	ISO8601_TIMEZONE	Matches hours and minutes that are in the ISO 8601 format.
	ISO8601_SECOND	Matches seconds that are in the ISO 8601 format.
	TIMESTAMP_ISO8601	Matches time that is in the ISO 8601 format.
	DATE	Matches dates that are in the US or EU format.
	DATESTAMP	Matches dates and time.
	TZ	Matches UTC time zones.
	DATESTAMP_RFC822	Matches time that is in the RFC 822 format.
	DATESTAMP_RFC2822	Matches time that is in the RFC 2822 format.
	DATESTAMP_OTHER	Matches time that is in other formats.
	DATESTAMP_EVENTLOG	Matches time that is in the EventLog format.
	HTTPDERROR_DATE	Matches time that is in the httpd error format.
SYSLOG	SYSLOGTIMESTAMP	Matches time that is in the Syslog format.
	PROG	Matches programs.
	SYSLOGPROG	Matches programs and process identifiers (PIDs).
	SYSLOGHOST	Matches IP addresses or hostnames.
	SYSLOGFACILITY	Matches facilities.
	HTTPDATE	Matches dates and time that are in the HTTP format.
LOGFORMATL	LOGFORMAT	Matches Syslog logs that are in the traditional format.
	COMMONAPACHELOG	Matches common Apache logs.
	COMBINEDAPACHELOG	Matches combined Apache logs.
	HTTPD20_ERRORLOG	Matches httpd20 logs.
	HTTPD24_ERRORLOG	Matches httpd24 logs.

Type	Pattern	Description
	HTTPD_ERRORLOG	Matches httpd logs.
LOGLEVELS	LOGLEVELS	Matches log levels, such as warn and debug.

General GROK patterns

```
EXTRACTJSON (? <json>(?:\{\s*"(?:\\"|[^"])+"\s*:\s*(?:(? P>json)|"(?:\\"|[^"])+"|[-+]?( 0|[
1-9]\d*)(?:\.[-+]?( 0|[1-9]\d*))?(?:[eE][-+]?( 0|[1-9]\d*))? |(?:true|false)|null)(?:\s*,\s
*"(?:\\"|[^"])+"\s*:\s*(?:(? P>json)|"(?:\\"|[^"])+"|[-+]?( 0|[1-9]\d*)(?:\.[-+]?( 0|[1-9]\
d*))?(?:[eE][-+]?( 0|[1-9]\d*))? |(?:true|false)|null))*\s*\}|\[\s*(?:(? P>json)|"(?:\\"|[^
"])+"|[-+]?( 0|[1-9]\d*)(?:\.[-+]?( 0|[1-9]\d*))?(?:[eE][-+]?( 0|[1-9]\d*))? |(?:true|false
)|null)(?:\s*,\s*(?:(? P>json)|"(?:\\"|[^"])+"|[-+]?( 0|[1-9]\d*)(?:\.[-+]?( 0|[1-9]\d*))?(
?:[eE][-+]?( 0|[1-9]\d*))? |(?:true|false)|null))*\s*\]))
CHINAID [1-9]\d{5})((18|19|([23]\d))\d{2}((0[1-9])|(10|11|12))(([0-2][1-9])|10|20|30|31)\d{
3}[0-9Xx]$)|(^[1-9]\d{5}\d{2}((0[1-9])|(10|11|12))(([0-2][1-9])|10|20|30|31)\d{3}
USERNAME [a-zA-Z0-9. _-]+
USER %{USERNAME}
EMAILLOCALPART [a-zA-Z][a-zA-Z0-9_. +-=:]+
EMAILADDRESS %{EMAILLOCALPART}@%{HOSTNAME}
HTTPDUSER %{EMAILADDRESS}|%{USER}
INT (?:[+-]?(?:[0-9]+))
BASE10NUM (? <![ 0-9. +-])(? >[+-]?(?:(?:[0-9]+(?:\.[ 0-9]+)?) |(?:\.[ 0-9]+)))
NUMBER (?:%{BASE10NUM})
BASE16NUM (? <![ 0-9A-Fa-f])(?:[+-]?(?:0x)?(?:[0-9A-Fa-f]+))
BASE16FLOAT \b(? <![ 0-9A-Fa-f.])(?:[+-]?(?:0x)?(?:(?:[0-9A-Fa-f]+(?:\.[ 0-9A-Fa-f]*)?) |(?
:\.[ 0-9A-Fa-f]+)))\b
POSINT \b(?:[1-9][0-9]*)\b
NONNEGINT \b(?:[0-9]+)\b
WORD \b\w+\b
NOTSPACE \S+
SPACE \s*
DATA . *?
GREEDYDATA . *
QUOTEDSTRING (? >(? <! \\)(? >"(? >\\. |[^\\"]+)+"|""|(? >'(? >\\. |[^\\']+)+')|''|(? >`(?
>\\. |[^\\`]+)+`)|``))
UUID [A-Fa-f0-9]{8}-(?:[A-Fa-f0-9]{4}-){3}[A-Fa-f0-9]{12}
"""Networking"""
MAC (?:%{CISCOMAC}|%{WINDOWSMAC}|%{COMMONMAC})
CISCOMAC (?:(?:[A-Fa-f0-9]{4}\.){2} [A-Fa-f0-9]{4})
WINDOWSMAC (?:(?:[A-Fa-f0-9]{2}-){5}[A-Fa-f0-9]{2})
COMMONMAC (?:(?:[A-Fa-f0-9]{2}:){5}[A-Fa-f0-9]{2})
IPV6 ((([0-9A-Fa-f]{1,4}:){7}([0-9A-Fa-f]{1,4}|:))|(([0-9A-Fa-f]{1,4}:){6}(:[0-9A-Fa-f]{1,4
}|((25[0-5]|2[0-4]\d|1\d\d|[1-9]? \d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]? \d)){3})|:))|(([0-9A
-Fa-f]{1,4}:){5}(((:[0-9A-Fa-f]{1,4}){1,2})|:((25[0-5]|2[0-4]\d|1\d\d|[1-9]? \d)(\.(25[0-5]
|2[0-4]\d|1\d\d|[1-9]? \d)){3})|:))|(([0-9A-Fa-f]{1,4}:){4}(((:[0-9A-Fa-f]{1,4}){1,3})|((:[
0-9A-Fa-f]{1,4})?:((25[0-5]|2[0-4]\d|1\d\d|[1-9]? \d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]? \d))
{3})|:))|(([0-9A-Fa-f]{1,4}:){3}(((:[0-9A-Fa-f]{1,4}){1,4})|((:[0-9A-Fa-f]{1,4}){0,2}:((25
[0-5]|2[0-4]\d|1\d\d|[1-9]? \d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]? \d)){3}))|:))|(([0-9A-Fa-f
]{1,4}:){2}(((:[0-9A-Fa-f]{1,4}){1,5})|((:[0-9A-Fa-f]{1,4}){0,3}:((25[0-5]|2[0-4]\d|1\d\d|[
1-9]? \d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]? \d)){3}))|:))|(([0-9A-Fa-f]{1,4}:){1}(((:[0-9A-F
a-f]{1,4}){1,6})|((:[0-9A-Fa-f]{1,4}){0,4}:((25[0-5]|2[0-4]\d|1\d\d|[1-9]? \d)(\.(25[0-5]|2
```

```
[0-4]\d|1\d\d|[1-9]? \d)){3}))|:))|(:(((:[0-9A-Fa-f]{1,4}){1,7})|((:[0-9A-Fa-f]{1,4}){0,5}:
((25[0-5]|2[0-4]\d|1\d\d|[1-9]? \d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]? \d)){3}))|:)))(%. +)?
IPV4 (? <![ 0-9])(?:(?:[0-1]?[ 0-9]{1,2}|2[0-4][0-9]|25[0-5])[.](?:[0-1]?[ 0-9]{1,2}|2[0-4]
[0-9]|25[0-5])[.](?:[0-1]?[ 0-9]{1,2}|2[0-4][0-9]|25[0-5])[.](?:[0-1]?[ 0-9]{1,2}|2[0-4][0-
9]|25[0-5]))(?![ 0-9])
IP (?:%{IPV6}|%{IPV4})
HOSTNAME \b(?:[0-9A-Za-z][0-9A-Za-z-]{0,62})(?:\.(?:[0-9A-Za-z][0-9A-Za-z-]{0,62}))*(\.? |\
b)
IPORHOST (?:%{IP}|%{HOSTNAME})
HOSTPORT %{IPORHOST}:%{POSINT}
"""paths"""
PATH (?:%{UNIXPATH}|%{WINPATH})
UNIXPATH (/([\w_%! $@:.,~-]+|\\.) *)+
TTY (?:/dev/(pts|tty([pq])?)( \w+)? /?(?:[0-9]+))
WINPATH (? >[A-Za-z]+:|\\)(?:\\[^\\? *]*)+
URIPROTO [A-Za-z]+(\+[A-Za-z+]+)?
URIHOST %{IPORHOST}(?::%{POSINT:port})?
"""uripath comes loosely from RFC1738, but mostly from what Firefox"""
"""doesn't turn into %XX"""
URIPATH (?:/[A-Za-z0-9$. +! *'(){},~:;=@#%_\-]*)+
"""URIPARAM \?(?:[A-Za-z0-9]+(?:=(?:[^&]*))?(?:&(?:[A-Za-z0-9]+(?:=(?:[^&]*))?)?) *)?"""
URIPARAM \?[ A-Za-z0-9$. +! *'|(){},~@#%&/=:;_? \-\[\]<>]*
URIPATHPARAM %{URIPATH}(?:%{URIPARAM})?
URI %{URIPROTO}://(?:%{USER}(?::[^@]*)? @)?(?:%{URIHOST})?(?:%{URIPATHPARAM})?
""" Months: January, Feb, 3, 03, 12, December"""
MONTH \b(?:Jan(?:uary|uar)? |Feb(?:ruary|ruar)? |M(?:a|ä)? r(?:ch|z)? |Apr(?:il)? |Ma(?:y|i
)? |Jun(?:e|i)? |Jul(?:y)? |Aug(?:ust)? |Sep(?:tember)? |O(?:c|k)? t(?:ober)? |Nov(?:ember)
? |De(?:c|z)(?:ember)?) \b
MONTHNUM (?:0?[ 1-9]|1[0-2])
MONTHNUM2 (?:0[1-9]|1[0-2])
MONTHDAY (?:(?:0[1-9])|(?:[12][0-9])|(?:3[01])|[1-9])
"""Days: Monday, Tue, Thu, etc..."""
DAY (?:Mon(?:day)? |Tue(?:sday)? |Wed(?:nesday)? |Thu(?:rsday)? |Fri(?:day)? |Sat(?:urday)?
|Sun(?:day)?)
""" Years?"""
YEAR (? >\d\d){1,2}
HOUR (?:2[0123]|[01]?[ 0-9])
MINUTE (?:[0-5][0-9])
"""'60' is a leap second in most time standards and thus is valid."""
SECOND (?:(?:[0-5]?[ 0-9]|60)(?:[:.,][0-9]+)?)
TIME (?! <[0-9])%{HOUR}:%{MINUTE}(?::%{SECOND})(?![ 0-9])
"""datestamp is YYYY/MM/DD-HH:MM:SS.UUUU (or something like it)"""
DATE_US %{MONTHNUM}[/-]%{MONTHDAY}[/-]%{YEAR}
DATE_EU %{MONTHDAY}[. /-]%{MONTHNUM}[. /-]%{YEAR}
ISO8601_TIMEZONE (?:Z|[+-]%{HOUR}(?::? %{MINUTE}))
ISO8601_SECOND (?:%{SECOND}|60)
TIMESTAMP_ISO8601 %{YEAR}-%{MONTHNUM}-%{MONTHDAY}[T ]%{HOUR}:? %{MINUTE}(?::? %{SECOND})? %
{ISO8601_TIMEZONE}?
DATE %{DATE_US}|%{DATE_EU}
DATESTAMP %{DATE}[- ]%{TIME}
TZ (?:[PMCE][SD]T|UTC)
DATESTAMP_RFC822 %{DAY} %{MONTH} %{MONTHDAY} %{YEAR} %{TIME} %{TZ}
DATESTAMP_RFC2822 %{DAY}, %{MONTHDAY} %{MONTH} %{YEAR} %{TIME} %{ISO8601_TIMEZONE}
DATESTAMP_OTHER %{DAY} %{MONTH} %{MONTHDAY} %{TIME} %{TZ} %{YEAR}
DATESTAMP_EVENTLOG %{YEAR}%{MONTHNUM2}%{MONTHDAY}%{HOUR}%{MINUTE}%{SECOND}
```

```
DATESTAMP_EVENTLOG %{YEAR}%{MONTHNUM2}%{MONTHDAY}%{HOUR}%{MINUTE}%{SECOND}
HTTPDERROR_DATE %{DAY} %{MONTH} %{MONTHDAY} %{TIME} %{YEAR}
"""Syslog Dates: Month Day HH:MM:SS"""
SYSLOGTIMESTAMP %{MONTH} +%{MONTHDAY} %{TIME}
PROG [\x21-\x5a\x5c\x5e-\x7e]+
SYSLOGPROG %{PROG:program}(?:\[%{POSINT:pid}\])?
SYSLOGHOST %{IPORHOST}
SYSLOGFACILITY <%{NONNEGINT:facility}. %{NONNEGINT:priority}>
HTTPDATE %{MONTHDAY}/%{MONTH}/%{YEAR}:%{TIME} %{INT}
"""Shortcuts"""
QS %{QUOTEDSTRING}
"""Log formats"""
SYSLOGBASE %{SYSLOGTIMESTAMP:timestamp} (?:%{SYSLOGFACILITY} )? %{SYSLOGHOST:logsource} %{S
YSLOGPROG}:
COMMONAPACHELOG %{IPORHOST:clientip} %{HTTPDUSER:ident} %{USER:auth} \[%{HTTPDATE:timestamp
}\] "(?:%{WORD:verb} %{NOTSPACE:request}(?: HTTP/%{NUMBER:httpversion})? |%{DATA:rawrequest
})" %{NUMBER:response} (?:%{NUMBER:bytes}|-)
COMBINEDAPACHELOG %{COMMONAPACHELOG} %{QS:referrer} %{QS:agent}
HTTPD20_ERRORLOG \[%{HTTPDERROR_DATE:timestamp}\] \[%{LOGLEVEL:loglevel}\] (?:\[client %{IP
ORHOST:clientip}\] ){0,1}%{GREEDYDATA:errormsg}
HTTPD24_ERRORLOG \[%{HTTPDERROR_DATE:timestamp}\] \[%{WORD:module}:%{LOGLEVEL:loglevel}\] \
[pid %{POSINT:pid}:tid %{NUMBER:tid}\]( \(%{POSINT:proxy_errorcode}\)%{DATA:proxy_errormess
age}:)?( \[client %{IPORHOST:client}:%{POSINT:clientport}\])? %{DATA:errorcode}: %{GREEDYDA
TA:message}
HTTPD_ERRORLOG %{HTTPD20_ERRORLOG}|%{HTTPD24_ERRORLOG}
"""Log Levels"""
LOGLEVEL ([Aa]lert|ALERT|[Tt]race|TRACE|[Dd]ebug|DEBUG|[Nn]otice|NOTICE|[Ii]nfo|INFO|[Ww]ar
n?(?:ing)? |WARN?(?:ING)? |[Ee]rr?(?:or)? |ERR?(?:OR)? |[Cc]rit?(?:ical)? |CRIT?(?:ICAL)? |
[Ff]atal|FATAL|[Ss]evere|SEVERE|EMERG(?:ENCY)? |[Ee]merg(?:ency)?)
```

Examples of basic syntax

```
CHINAID [1-9]\d{5})((18|19|([23]\d))\d{2}((0[1-9])|(10|11|12))(([0-2][1-9])|10|20|30|31)\d{
3}[0-9Xx]$)|(^[1-9]\d{5}\d{2}((0[1-9])|(10|11|12))(([0-2][1-9])|10|20|30|31)\d{3}
USERNAME [a-zA-Z0-9. _-]+
USER %{USERNAME}
EMAILLOCALPART [a-zA-Z][a-zA-Z0-9_. +-=:]+
EMAILADDRESS %{EMAILLOCALPART}@%{HOSTNAME}
HTTPDUSER %{EMAILADDRESS}|%{USER}
INT (?:[+-]?(?:[0-9]+))
BASE10NUM (? <![ 0-9. +-])(? >[+-]?(?:(?:[0-9]+(?:\.[ 0-9]+)?) |(?:\.[ 0-9]+)))
NUMBER (?:%{BASE10NUM})
BASE16NUM (? <![ 0-9A-Fa-f])(?:[+-]?(?:0x)?(?:[0-9A-Fa-f]+))
BASE16FLOAT \b(? <![ 0-9A-Fa-f.])(?:[+-]?(?:0x)?(?:(?:[0-9A-Fa-f]+(?:\.[ 0-9A-Fa-f]*)?) |(?
:\.[ 0-9A-Fa-f]+)))\b
POSINT \b(?:[1-9][0-9]*)\b
NONNEGINT \b(?:[0-9]+)\b
WORD \b\w+\b
NOTSPACE \S+
SPACE \s*
DATA . *?
GREEDYDATA . *
QUOTEDSTRING (? >(? <! \\)(? >"(? >\\. |[^\\"]+)+"|""|(? >'(? >\\. |[^\\']+)+')|''|(? >`(?
>\\. |[^\\`]+)+`)|``))
```

```
UUID [A-Fa-f0-9]{8}-(?:[A-Fa-f0-9]{4}-){3}[A-Fa-f0-9]{12}
"""Networking"""
MAC (?:%{CISCOMAC}|%{WINDOWSMAC}|%{COMMONMAC})
CISCOMAC (?:(?:[A-Fa-f0-9]{4}\.){2} [A-Fa-f0-9]{4})
WINDOWSMAC (?:(?:[A-Fa-f0-9]{2}-){5}[A-Fa-f0-9]{2})
COMMONMAC (?:(?:[A-Fa-f0-9]{2}:){5}[A-Fa-f0-9]{2})
IPV6 ((([0-9A-Fa-f]{1,4}:){7}([0-9A-Fa-f]{1,4}|:))|(([0-9A-Fa-f]{1,4}:){6}(:[0-9A-Fa-f]{1,4
}|((25[0-5]|2[0-4]\d|1\d\d|[1-9]? \d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]? \d)){3})|:))|(([0-9A
-Fa-f]{1,4}:){5}(((:[0-9A-Fa-f]{1,4}){1,2})|:((25[0-5]|2[0-4]\d|1\d\d|[1-9]? \d)(\.(25[0-5]
|2[0-4]\d|1\d\d|[1-9]? \d)){3})|:))|(([0-9A-Fa-f]{1,4}:){4}(((:[0-9A-Fa-f]{1,4}){1,3})|((:[
0-9A-Fa-f]{1,4})?:((25[0-5]|2[0-4]\d|1\d\d|[1-9]? \d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]? \d))
{3}))|:))|(([0-9A-Fa-f]{1,4}:){3}(((:[0-9A-Fa-f]{1,4}){1,4})|((:[0-9A-Fa-f]{1,4}){0,2}:((25
[0-5]|2[0-4]\d|1\d\d|[1-9]? \d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]? \d)){3}))|:))|(([0-9A-Fa-f
]{1,4}:){2}(((:[0-9A-Fa-f]{1,4}){1,5})|((:[0-9A-Fa-f]{1,4}){0,3}:((25[0-5]|2[0-4]\d|1\d\d|[
1-9]? \d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]? \d)){3}))|:))|(([0-9A-Fa-f]{1,4}:){1}(((:[0-9A-F
a-f]{1,4}){1,6})|((:[0-9A-Fa-f]{1,4}){0,4}:((25[0-5]|2[0-4]\d|1\d\d|[1-9]? \d)(\.(25[0-5]|2
[0-4]\d|1\d\d|[1-9]? \d)){3}))|:))|(:(((:[0-9A-Fa-f]{1,4}){1,7})|((:[0-9A-Fa-f]{1,4}){0,5}:
((25[0-5]|2[0-4]\d|1\d\d|[1-9]? \d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]? \d)){3}))|:))(%. +)?
IPV4 (? <![ 0-9])(?:(?:[0-1]?[ 0-9]{1,2}|2[0-4][0-9]|25[0-5])[.](?:[0-1]?[ 0-9]{1,2}|2[0-4]
[0-9]|25[0-5])[.](?:[0-1]?[ 0-9]{1,2}|2[0-4][0-9]|25[0-5])[.](?:[0-1]?[ 0-9]{1,2}|2[0-4][0-
9]|25[0-5]))(?![ 0-9])
IP (?:%{IPV6}|%{IPV4})
HOSTNAME \b(?:[0-9A-Za-z][0-9A-Za-z-]{0,62})(?:\.(?:[0-9A-Za-z][0-9A-Za-z-]{0,62}))*(\.? |\
b)
IPORHOST (?:%{IP}|%{HOSTNAME})
HOSTPORT %{IPORHOST}:%{POSINT}
"""paths"""
PATH (?:%{UNIXPATH}|%{WINPATH})
UNIXPATH (/([\w_%! $@:.,~-]+|\\.) *)+
TTY (?:/dev/(pts|tty([pq])?)( \w+)? /?(?:[0-9]+))
WINPATH (? >[A-Za-z]+:|\\)(?:\\[^\\? *]*)+
URIPROTO [A-Za-z]+(\+[A-Za-z+]+)?
URIHOST %{IPORHOST}(?::%{POSINT:port})?
"""uripath comes loosely from RFC1738, but mostly from what Firefox"""
"""doesn't turn into %XX"""
URIPATH (?:/[A-Za-z0-9$. +! *'(){},~:;=@#%_\-]*)+
"""URIPARAM \?(?:[A-Za-z0-9]+(?:=(?:[^&]*))?(?:&(?:[A-Za-z0-9]+(?:=(?:[^&]*))?)?) *)?"""
URIPARAM \?[ A-Za-z0-9$. +! *'|(){},~@#%&/=:;_? \-\[\]<>]*
URIPATHPARAM %{URIPATH}(?:%{URIPARAM})?
URI %{URIPROTO}://(?:%{USER}(?::[^@]*)? @)?(?:%{URIHOST})?(?:%{URIPATHPARAM})?
""" Months: January, Feb, 3, 03, 12, December"""
MONTH \b(?:Jan(?:uary|uar)? |Feb(?:ruary|ruar)? |M(?:a|ä)? r(?:ch|z)? |Apr(?:il)? |Ma(?:y|i
)? |Jun(?:e|i)? |Jul(?:y)? |Aug(?:ust)? |Sep(?:tember)? |O(?:c|k)? t(?:ober)? |Nov(?:ember)
? |De(?:c|z)(?:ember)?) \b
MONTHNUM (?:0?[ 1-9]|1[0-2])
MONTHNUM2 (?:0[1-9]|1[0-2])
MONTHDAY (?:(?:0[1-9])|(?:[12][0-9])|(?:3[01])|[1-9])
"""Days: Monday, Tue, Thu, etc..."""
DAY (?:Mon(?:day)? |Tue(?:sday)? |Wed(?:nesday)? |Thu(?:rsday)? |Fri(?:day)? |Sat(?:urday)?
|Sun(?:day)?)
""" Years?"""
YEAR (? >\d\d){1,2}
HOUR (?:2[0123]|[01]?[ 0-9])
MINUTE (?:[0-5][0-9])
```

```
"""'60' is a leap second in most time standards and thus is valid."""
SECOND (?:(?:[0-5]?[ 0-9]|60)(?:[:.,][0-9]+)?)
TIME (?! <[0-9])%{HOUR}:%{MINUTE}(?::%{SECOND})(?![ 0-9])
"""datestamp is YYYY/MM/DD-HH:MM:SS.UUUU (or something like it)"""
DATE_US %{MONTHNUM}[/-]%{MONTHDAY}[/-]%{YEAR}
DATE_EU %{MONTHDAY}[. /-]%{MONTHNUM}[. /-]%{YEAR}
ISO8601_TIMEZONE (?:Z|[+-]%{HOUR}(?::? %{MINUTE}))
ISO8601_SECOND (?:%{SECOND}|60)
TIMESTAMP_ISO8601 %{YEAR}-%{MONTHNUM}-%{MONTHDAY}[T ]%{HOUR}:? %{MINUTE}(?::? %{SECOND})? %
{ISO8601_TIMEZONE}?
DATE %{DATE_US}|%{DATE_EU}
DATESTAMP %{DATE}[- ]%{TIME}
TZ (?:[PMCE][SD]T|UTC)
DATESTAMP_RFC822 %{DAY} %{MONTH} %{MONTHDAY} %{YEAR} %{TIME} %{TZ}
DATESTAMP_RFC2822 %{DAY}, %{MONTHDAY} %{MONTH} %{YEAR} %{TIME} %{ISO8601_TIMEZONE}
DATESTAMP_OTHER %{DAY} %{MONTH} %{MONTHDAY} %{TIME} %{TZ} %{YEAR}
DATESTAMP_EVENTLOG %{YEAR}%{MONTHNUM2}%{MONTHDAY}%{HOUR}%{MINUTE}%{SECOND}
HTTPDERROR_DATE %{DAY} %{MONTH} %{MONTHDAY} %{TIME} %{YEAR}
"""Syslog Dates: Month Day HH:MM:SS"""
SYSLOGTIMESTAMP %{MONTH} +%{MONTHDAY} %{TIME}
PROG [\x21-\x5a\x5c\x5e-\x7e]+
SYSLOGPROG %{PROG:program}(?:\[%{POSINT:pid}\])?
SYSLOGHOST %{IPORHOST}
SYSLOGFACILITY <%{NONNEGINT:facility}. %{NONNEGINT:priority}>
HTTPDATE %{MONTHDAY}/%{MONTH}/%{YEAR}:%{TIME} %{INT}
"""Shortcuts"""
QS %{QUOTEDSTRING}
"""Log formats"""
SYSLOGBASE %{SYSLOGTIMESTAMP:timestamp} (?:%{SYSLOGFACILITY} )? %{SYSLOGHOST:logsource} %{S
YSLOGPROG}:
COMMONAPACHELOG %{IPORHOST:clientip} %{HTTPDUSER:ident} %{USER:auth} \[%{HTTPDATE:timestamp
}\] "(?:%{WORD:verb} %{NOTSPACE:request}(?: HTTP/%{NUMBER:httpversion})? |%{DATA:rawrequest
})" %{NUMBER:response} (?:%{NUMBER:bytes}|-)
COMBINEDAPACHELOG %{COMMONAPACHELOG} %{QS:referrer} %{QS:agent}
HTTPD20_ERRORLOG \[%{HTTPDERROR_DATE:timestamp}\] \[%{LOGLEVEL:loglevel}\] (?:\[client %{IP
ORHOST:clientip}\] ){0,1}%{GREEDYDATA:errormsg}
HTTPD24_ERRORLOG \[%{HTTPDERROR_DATE:timestamp}\] \[%{WORD:module}:%{LOGLEVEL:loglevel}\] \
[pid %{POSINT:pid}:tid %{NUMBER:tid}\]( \(%{POSINT:proxy_errorcode}\)%{DATA:proxy_errormess
age}:)?( \[client %{IPORHOST:client}:%{POSINT:clientport}\])? %{DATA:errorcode}: %{GREEDYDA
TA:message}
HTTPD_ERRORLOG %{HTTPD20_ERRORLOG}|%{HTTPD24_ERRORLOG}
"""Log Levels"""
LOGLEVEL ([Aa]lert|ALERT|[Tt]race|TRACE|[Dd]ebug|DEBUG|[Nn]otice|NOTICE|[Ii]nfo|INFO|[Ww]ar
n?(?:ing)? |WARN?(?:ING)? |[Ee]rr?(?:or)? |ERR?(?:OR)? |[Cc]rit?(?:ical)? |CRIT?(?:ICAL)? |
[Ff]atal|FATAL|[Ss]evere|SEVERE|EMERG(?:ENCY)? |[Ee]merg(?:ency)?)
```

Sample patterns for AWS

```
S3_REQUEST_LINE (?:%{WORD:verb} %{NOTSPACE:request}(?: HTTP/%{NUMBER:httpversion})? |%{DATA:rawrequest})
S3_ACCESS_LOG %{WORD:owner} %{NOTSPACE:bucket} \[%{HTTPDATE:timestamp}\] %{IP:clientip} %{NOTSPACE:requester} %{NOTSPACE:request_id} %{NOTSPACE:operation} %{NOTSPACE:key} (?:"%{S3_REQUEST_LINE}"|-) (?:%{INT:response:int}|-) (?:-|%{NOTSPACE:error_code}) (?:%{INT:bytes:int}|-) (?:%{INT:object_size:int}|-) (?:%{INT:request_time_ms:int}|-) (?:%{INT:turnaround_time_ms:int}|-) (?:%{QS:referrer}|-) (?:"? %{QS:agent}"? |-) (?:-|%{NOTSPACE:version_id})
ELB_URIPATHPARAM %{URIPATH:path}(?:%{URIPARAM:params})?
ELB_URI %{URIPROTO:proto}://(?:%{USER}(?::[^@]*)? @)?(?:%{URIHOST:urihost})?(?:%{ELB_URIPATHPARAM})?
ELB_REQUEST_LINE (?:%{WORD:verb} %{ELB_URI:request}(?: HTTP/%{NUMBER:httpversion})? |%{DATA:rawrequest})
ELB_ACCESS_LOG %{TIMESTAMP_ISO8601:timestamp} %{NOTSPACE:elb} %{IP:clientip}:%{INT:clientport:int} (?:(%{IP:backendip}:?:%{INT:backendport:int})|-) %{NUMBER:request_processing_time:float} %{NUMBER:backend_processing_time:float} %{NUMBER:response_processing_time:float} %{INT:response:int} %{INT:backend_response:int} %{INT:received_bytes:int} %{INT:bytes:int} "%{ELB_REQUEST_LINE}"
```

Sample patterns for Bacula

```
BACULA_TIMESTAMP %{MONTHDAY}-%{MONTH} %{HOUR}:%{MINUTE}
BACULA_HOST [a-zA-Z0-9-]+
BACULA_VOLUME %{USER}
BACULA_DEVICE %{USER}
BACULA_DEVICEPATH %{UNIXPATH}
BACULA_CAPACITY %{INT}{1,3}(,%{INT}{3})*
BACULA_VERSION %{USER}
BACULA_JOB %{USER}
BACULA_LOG_MAX_CAPACITY User defined maximum volume capacity %{BACULA_CAPACITY} exceeded on device \"%{BACULA_DEVICE:device}\" \(%{BACULA_DEVICEPATH}\)
BACULA_LOG_END_VOLUME End of medium on Volume \"%{BACULA_VOLUME:volume}\" Bytes=%{BACULA_CAPACITY} Blocks=%{BACULA_CAPACITY} at %{MONTHDAY}-%{MONTH}-%{YEAR} %{HOUR}:%{MINUTE}.
BACULA_LOG_NEW_VOLUME Created new Volume \"%{BACULA_VOLUME:volume}\" in catalog.
BACULA_LOG_NEW_LABEL Labeled new Volume \"%{BACULA_VOLUME:volume}\" on device \"%{BACULA_DEVICE:device}\" \(%{BACULA_DEVICEPATH}\).
BACULA_LOG_WROTE_LABEL Wrote label to prelabeled Volume \"%{BACULA_VOLUME:volume}\" on device \"%{BACULA_DEVICE}\" \(%{BACULA_DEVICEPATH}\)
BACULA_LOG_NEW_MOUNT New volume \"%{BACULA_VOLUME:volume}\" mounted on device \"%{BACULA_DEVICE:device}\" \(%{BACULA_DEVICEPATH}\) at %{MONTHDAY}-%{MONTH}-%{YEAR} %{HOUR}:%{MINUTE}.
BACULA_LOG_NOOPEN \s+Cannot open %{DATA}: ERR=%{GREEDYDATA:berror}
BACULA_LOG_NOOPENDIR \s+Could not open directory %{DATA}: ERR=%{GREEDYDATA:berror}
BACULA_LOG_NOSTAT \s+Could not stat %{DATA}: ERR=%{GREEDYDATA:berror}
BACULA_LOG_NOJOBS There are no more Jobs associated with Volume \"%{BACULA_VOLUME:volume}\". Marking it purged.
BACULA_LOG_ALL_RECORDS_PRUNED All records pruned from Volume \"%{BACULA_VOLUME:volume}\"; marking it \"Purged\"
BACULA_LOG_BEGIN_PRUNE_JOBS Begin pruning Jobs older than %{INT} month %{INT} days .
BACULA_LOG_BEGIN_PRUNE_FILES Begin pruning Files.
BACULA_LOG_PRUNED_JOBS Pruned %{INT} Jobs* for client %{BACULA_HOST:client} from catalog.
BACULA_LOG_PRUNED_FILES Pruned Files from %{INT} Jobs* for client %{BACULA_HOST:client} from catalog.
BACULA_LOG_ENDPRUNE End auto prune.
BACULA_LOG_STARTJOB Start Backup JobId %{INT}, Job=%{BACULA_JOB:job}
```

```
BACULA_LOG_STARTRESTORE Start Restore Job %{BACULA_JOB:job}
BACULA_LOG_USEDEVICE Using Device \"%{BACULA_DEVICE:device}\"
BACULA_LOG_DIFF_FS \s+%{UNIXPATH} is a different filesystem. Will not descend from %{UNIXPA
TH} into it.
BACULA_LOG_JOBEND Job write elapsed time = %{DATA:elapsed}, Transfer rate = %{NUMBER} (K|M|
G)? Bytes/second
BACULA_LOG_NOPRUNE_JOBS No Jobs found to prune.
BACULA_LOG_NOPRUNE_FILES No Files found to prune.
BACULA_LOG_VOLUME_PREVWRITTEN Volume \"%{BACULA_VOLUME:volume}\" previously written, moving
to end of data.
BACULA_LOG_READYAPPEND Ready to append to end of Volume \"%{BACULA_VOLUME:volume}\" size=%{
INT}
BACULA_LOG_CANCELLING Cancelling duplicate JobId=%{INT}.
BACULA_LOG_MARKCANCEL JobId %{INT}, Job %{BACULA_JOB:job} marked to be canceled.
BACULA_LOG_CLIENT_RBJ shell command: run ClientRunBeforeJob \"%{GREEDYDATA:runjob}\"
BACULA_LOG_VSS (Generate )? VSS (Writer)?
BACULA_LOG_MAXSTART Fatal error: Job canceled because max start delay time exceeded.
BACULA_LOG_DUPLICATE Fatal error: JobId %{INT:duplicate} already running. Duplicate job not
allowed.
BACULA_LOG_NOJOBSTAT Fatal error: No Job status returned from FD.
BACULA_LOG_FATAL_CONN Fatal error: bsock.c:133 Unable to connect to (Client: %{BACULA_HOST:
client}|Storage daemon) on %{HOSTNAME}:%{POSINT}. ERR=(? <berror>%{GREEDYDATA})
BACULA_LOG_NO_CONNECT Warning: bsock.c:127 Could not connect to (Client: %{BACULA_HOST:clie
nt}|Storage daemon) on %{HOSTNAME}:%{POSINT}. ERR=(? <berror>%{GREEDYDATA})
BACULA_LOG_NO_AUTH Fatal error: Unable to authenticate with File daemon at %{HOSTNAME}. Pos
sible causes:
BACULA_LOG_NOSUIT No prior or suitable Full backup found in catalog. Doing FULL backup.
BACULA_LOG_NOPRIOR No prior Full backup Job record found.
BACULA_LOG_JOB (Error: )? Bacula %{BACULA_HOST} %{BACULA_VERSION} \(%{BACULA_VERSION}\):
BACULA_LOGLINE %{BACULA_TIMESTAMP:bts} %{BACULA_HOST:hostname} JobId %{INT:jobid}: (%{BACUL
A_LOG_MAX_CAPACITY}|%{BACULA_LOG_END_VOLUME}|%{BACULA_LOG_NEW_VOLUME}|%{BACULA_LOG_NEW_LABE
L}|%{BACULA_LOG_WROTE_LABEL}|%{BACULA_LOG_NEW_MOUNT}|%{BACULA_LOG_NOOPEN}|%{BACULA_LOG_NOOP
ENDIR}|%{BACULA_LOG_NOSTAT}|%{BACULA_LOG_NOJOBS}|%{BACULA_LOG_ALL_RECORDS_PRUNED}|%{BACULA_
LOG_BEGIN_PRUNE_JOBS}|%{BACULA_LOG_BEGIN_PRUNE_FILES}|%{BACULA_LOG_PRUNED_JOBS}|%{BACULA_LO
G_PRUNED_FILES}|%{BACULA_LOG_ENDPRUNE}|%{BACULA_LOG_STARTJOB}|%{BACULA_LOG_STARTRESTORE}|%{
BACULA_LOG_USEDEVICE}|%{BACULA_LOG_DIFF_FS}|%{BACULA_LOG_JOBEND}|%{BACULA_LOG_NOPRUNE_JOBS}
|%{BACULA_LOG_NOPRUNE_FILES}|%{BACULA_LOG_VOLUME_PREVWRITTEN}|%{BACULA_LOG_READYAPPEND}|%{B
ACULA_LOG_CANCELLING}|%{BACULA_LOG_MARKCANCEL}|%{BACULA_LOG_CLIENT_RBJ}|%{BACULA_LOG_VSS}|%
{BACULA_LOG_MAXSTART}|%{BACULA_LOG_DUPLICATE}|%{BACULA_LOG_NOJOBSTAT}|%{BACULA_LOG_FATAL_CO
NN}|%{BACULA_LOG_NO_CONNECT}|%{BACULA_LOG_NO_AUTH}|%{BACULA_LOG_NOSUIT}|%{BACULA_LOG_JOB}|%
{BACULA_LOG_NOPRIOR})
```

Sample patterns for Bro

```
"""https://www.bro.org/sphinx/script-reference/log-files.html"""
"""http.log"""
BRO_HTTP %{NUMBER:ts}\t%{NOTSPACE:uid}\t%{IP:orig_h}\t%{INT:orig_p}\t%{IP:resp_h}\t%{INT:resp_p}\t%{INT:trans_depth}\t%{GREEDYDATA:method}\t%{GREEDYDATA:domain}\t%{GREEDYDATA:uri}\t%{GREEDYDATA:referrer}\t%{GREEDYDATA:user_agent}\t%{NUMBER:request_body_len}\t%{NUMBER:response_body_len}\t%{GREEDYDATA:status_code}\t%{GREEDYDATA:status_msg}\t%{GREEDYDATA:info_code}\t%{GREEDYDATA:info_msg}\t%{GREEDYDATA:filename}\t%{GREEDYDATA:bro_tags}\t%{GREEDYDATA:username}\t%{GREEDYDATA:password}\t%{GREEDYDATA:proxied}\t%{GREEDYDATA:orig_fuids}\t%{GREEDYDATA:orig_mime_types}\t%{GREEDYDATA:resp_fuids}\t%{GREEDYDATA:resp_mime_types}
"""dns.log"""
BRO_DNS %{NUMBER:ts}\t%{NOTSPACE:uid}\t%{IP:orig_h}\t%{INT:orig_p}\t%{IP:resp_h}\t%{INT:resp_p}\t%{WORD:proto}\t%{INT:trans_id}\t%{GREEDYDATA:query}\t%{GREEDYDATA:qclass}\t%{GREEDYDATA:qclass_name}\t%{GREEDYDATA:qtype}\t%{GREEDYDATA:qtype_name}\t%{GREEDYDATA:rcode}\t%{GREEDYDATA:rcode_name}\t%{GREEDYDATA:AA}\t%{GREEDYDATA:TC}\t%{GREEDYDATA:RD}\t%{GREEDYDATA:RA}\t%{GREEDYDATA:Z}\t%{GREEDYDATA:answers}\t%{GREEDYDATA:TTLs}\t%{GREEDYDATA:rejected}
"""conn.log"""
BRO_CONN %{NUMBER:ts}\t%{NOTSPACE:uid}\t%{IP:orig_h}\t%{INT:orig_p}\t%{IP:resp_h}\t%{INT:resp_p}\t%{WORD:proto}\t%{GREEDYDATA:service}\t%{NUMBER:duration}\t%{NUMBER:orig_bytes}\t%{NUMBER:resp_bytes}\t%{GREEDYDATA:conn_state}\t%{GREEDYDATA:local_orig}\t%{GREEDYDATA:missed_bytes}\t%{GREEDYDATA:history}\t%{GREEDYDATA:orig_pkts}\t%{GREEDYDATA:orig_ip_bytes}\t%{GREEDYDATA:resp_pkts}\t%{GREEDYDATA:resp_ip_bytes}\t%{GREEDYDATA:tunnel_parents}
"""files.log"""
BRO_FILES %{NUMBER:ts}\t%{NOTSPACE:fuid}\t%{IP:tx_hosts}\t%{IP:rx_hosts}\t%{NOTSPACE:conn_uids}\t%{GREEDYDATA:source}\t%{GREEDYDATA:depth}\t%{GREEDYDATA:analyzers}\t%{GREEDYDATA:mime_type}\t%{GREEDYDATA:filename}\t%{GREEDYDATA:duration}\t%{GREEDYDATA:local_orig}\t%{GREEDYDATA:is_orig}\t%{GREEDYDATA:seen_bytes}\t%{GREEDYDATA:total_bytes}\t%{GREEDYDATA:missing_bytes}\t%{GREEDYDATA:overflow_bytes}\t%{GREEDYDATA:timedout}\t%{GREEDYDATA:parent_fuid}\t%{GREEDYDATA:md5}\t%{GREEDYDATA:sha1}\t%{GREEDYDATA:sha256}\t%{GREEDYDATA:extracted}
```

Sample patterns for Exim

```
EXIM_MSGID [0-9A-Za-z]{6}-[0-9A-Za-z]{6}-[0-9A-Za-z]{2}
EXIM_FLAGS (<=|[-=>*]>|[*]{2}|==)
EXIM_DATE %{YEAR:exim_year}-%{MONTHNUM:exim_month}-%{MONTHDAY:exim_day} %{TIME:exim_time}
EXIM_PID \[%{POSINT}\]
EXIM_QT ((\d+y)?( \d+w)?( \d+d)?( \d+h)?( \d+m)?( \d+s)?)
EXIM_EXCLUDE_TERMS (Message is frozen|(Start|End) queue run| Warning: | retry time not reached | no (IP address|host name) found for (IP address|host) | unexpected disconnection while reading SMTP command | no immediate delivery: |another process is handling this message)
EXIM_REMOTE_HOST (H=(%{NOTSPACE:remote_hostname} )?( \(%{NOTSPACE:remote_heloname}\) )? \[%{IP:remote_host}\])
EXIM_INTERFACE (I=\[%{IP:exim_interface}\](:%{NUMBER:exim_interface_port}))
EXIM_PROTOCOL (P=%{NOTSPACE:protocol})
EXIM_MSG_SIZE (S=%{NUMBER:exim_msg_size})
EXIM_HEADER_ID (id=%{NOTSPACE:exim_header_id})
EXIM_SUBJECT (T=%{QS:exim_subject})
```

Sample patterns for Cisco firewalls

```
""" NetScreen firewall logs"""
NETSCREENSESSIONLOG %{SYSLOGTIMESTAMP:date} %{IPORHOST:device} %{IPORHOST}: NetScreen device_id=%{WORD:device_id}%{DATA}: start_time=%{QUOTEDSTRING:start_time} duration=%{INT:duratio
```

```
n} policy_id=%{INT:policy_id} service=%{DATA:service} proto=%{INT:proto} src zone=%{WORD:src_zone} dst zone=%{WORD:dst_zone} action=%{WORD:action} sent=%{INT:sent} rcvd=%{INT:rcvd} src=%{IPORHOST:src_ip} dst=%{IPORHOST:dst_ip} src_port=%{INT:src_port} dst_port=%{INT:dst_port} src-xlated ip=%{IPORHOST:src_xlated_ip} port=%{INT:src_xlated_port} dst-xlated ip=%{IPORHOST:dst_xlated_ip} port=%{INT:dst_xlated_port} session_id=%{INT:session_id} reason=%{GREEDYDATA:reason}
"""== Cisco ASA =="""
CISCO_TAGGED_SYSLOG ^<%{POSINT:syslog_pri}>%{CISCOTIMESTAMP:timestamp}( %{SYSLOGHOST:syslog host})? ?: %%{CISCOTAG:ciscotag}:
CISCOTIMESTAMP %{MONTH} +%{MONTHDAY}(?: %{YEAR})? %{TIME}
CISCOTAG [A-Z0-9]+-%{INT}-(?:[A-Z0-9_]+)
"""Common Particles"""
CISCO_ACTION Built|Teardown|Deny|Denied|denied|requested|permitted|denied by ACL|discarded|est-allowed|Dropping|created|deleted
CISCO_REASON Duplicate TCP SYN|Failed to locate egress interface|Invalid transport field|No matching connection|DNS Response|DNS Query|(?:%{WORD}\s*)*
CISCO_DIRECTION Inbound|inbound|Outbound|outbound
CISCO_INTERVAL first hit|%{INT}-second interval
CISCO_XLATE_TYPE static|dynamic
"""ASA-1-104001"""
CISCOFW104001 \((?:Primary|Secondary)\) Switching to ACTIVE - %{GREEDYDATA:switch_reason}
"""ASA-1-104002"""
CISCOFW104002 \((?:Primary|Secondary)\) Switching to STANDBY - %{GREEDYDATA:switch_reason}
"""ASA-1-104003"""
CISCOFW104003 \((?:Primary|Secondary)\) Switching to FAILED\.
"""ASA-1-104004"""
CISCOFW104004 \((?:Primary|Secondary)\) Switching to OK\.
"""ASA-1-105003"""
CISCOFW105003 \((?:Primary|Secondary)\) Monitoring on [Ii]nterface %{GREEDYDATA:interface_name} waiting
"""ASA-1-105004"""
CISCOFW105004 \((?:Primary|Secondary)\) Monitoring on [Ii]nterface %{GREEDYDATA:interface_name} normal
"""ASA-1-105005"""
CISCOFW105005 \((?:Primary|Secondary)\) Lost Failover communications with mate on [Ii]nterface %{GREEDYDATA:interface_name}
"""ASA-1-105008"""
CISCOFW105008 \((?:Primary|Secondary)\) Testing [Ii]nterface %{GREEDYDATA:interface_name}
"""ASA-1-105009"""
CISCOFW105009 \((?:Primary|Secondary)\) Testing on [Ii]nterface %{GREEDYDATA:interface_name} (?:Passed|Failed)
"""ASA-2-106001"""
CISCOFW106001 %{CISCO_DIRECTION:direction} %{WORD:protocol} connection %{CISCO_ACTION:action} from %{IP:src_ip}/%{INT:src_port} to %{IP:dst_ip}/%{INT:dst_port} flags %{GREEDYDATA:tcp_flags} on interface %{GREEDYDATA:interface}
"""ASA-2-106006, ASA-2-106007, ASA-2-106010"""
CISCOFW106006_106007_106010 %{CISCO_ACTION:action} %{CISCO_DIRECTION:direction} %{WORD:protocol} (?:from|src) %{IP:src_ip}/%{INT:src_port}(\(%{DATA:src_fwuser}\))? (?:to|dst) %{IP:dst_ip}/%{INT:dst_port}(\(%{DATA:dst_fwuser}\))? (?:on interface %{DATA:interface}|due to %{CISCO_REASON:reason})
"""ASA-3-106014"""
CISCOFW106014 %{CISCO_ACTION:action} %{CISCO_DIRECTION:direction} %{WORD:protocol} src %{DATA:src_interface}:%{IP:src_ip}(\(%{DATA:src_fwuser}\))? dst %{DATA:dst_interface}:%{IP:dst_ip}(\(%{DATA:dst_fwuser}\))? \(type %{INT:icmp_type}, code %{INT:icmp_code}\)
"""ASA-6-106015"""
```

```
"""ASA-6-106015"""
CISCOFW106015 %{CISCO_ACTION:action} %{WORD:protocol} \(%{DATA:policy_id}\) from %{IP:src_ip}/%{INT:src_port} to %{IP:dst_ip}/%{INT:dst_port} flags %{DATA:tcp_flags}  on interface %{GREEDYDATA:interface}
"""ASA-1-106021"""
CISCOFW106021 %{CISCO_ACTION:action} %{WORD:protocol} reverse path check from %{IP:src_ip} to %{IP:dst_ip} on interface %{GREEDYDATA:interface}
"""ASA-4-106023"""
CISCOFW106023 %{CISCO_ACTION:action}( protocol)? %{WORD:protocol} src %{DATA:src_interface}:%{DATA:src_ip}(/%{INT:src_port})?( \(%{DATA:src_fwuser}\))? dst %{DATA:dst_interface}:%{DATA:dst_ip}(/%{INT:dst_port})?( \(%{DATA:dst_fwuser}\))?( \(type %{INT:icmp_type}, code %{INT:icmp_code}\))? by access-group "? %{DATA:policy_id}"? \[%{DATA:hashcode1}, %{DATA:hashcode2}\]
"""ASA-4-106100, ASA-4-106102, ASA-4-106103"""
CISCOFW106100_2_3 access-list %{NOTSPACE:policy_id} %{CISCO_ACTION:action} %{WORD:protocol} for user '%{DATA:src_fwuser}' %{DATA:src_interface}/%{IP:src_ip}\(%{INT:src_port}\) -> %{DATA:dst_interface}/%{IP:dst_ip}\(%{INT:dst_port}\) hit-cnt %{INT:hit_count} %{CISCO_INTERVAL:interval} \[%{DATA:hashcode1}, %{DATA:hashcode2}\]
"""ASA-5-106100"""
CISCOFW106100 access-list %{NOTSPACE:policy_id} %{CISCO_ACTION:action} %{WORD:protocol} %{DATA:src_interface}/%{IP:src_ip}\(%{INT:src_port}\)(\(%{DATA:src_fwuser}\))? -> %{DATA:dst_interface}/%{IP:dst_ip}\(%{INT:dst_port}\)(\(%{DATA:src_fwuser}\))? hit-cnt %{INT:hit_count} %{CISCO_INTERVAL:interval} \[%{DATA:hashcode1}, %{DATA:hashcode2}\]
"""ASA-6-110002"""
CISCOFW110002 %{CISCO_REASON:reason} for %{WORD:protocol} from %{DATA:src_interface}:%{IP:src_ip}/%{INT:src_port} to %{IP:dst_ip}/%{INT:dst_port}
"""ASA-6-302010"""
CISCOFW302010 %{INT:connection_count} in use, %{INT:connection_count_max} most used
"""ASA-6-302013, ASA-6-302014, ASA-6-302015, ASA-6-302016"""
CISCOFW302013_302014_302015_302016 %{CISCO_ACTION:action}(?: %{CISCO_DIRECTION:direction})? %{WORD:protocol} connection %{INT:connection_id} for %{DATA:src_interface}:%{IP:src_ip}/%{INT:src_port}( \(%{IP:src_mapped_ip}/%{INT:src_mapped_port}\))?( \(%{DATA:src_fwuser}\))? to %{DATA:dst_interface}:%{IP:dst_ip}/%{INT:dst_port}( \(%{IP:dst_mapped_ip}/%{INT:dst_mapped_port}\))?( \(%{DATA:dst_fwuser}\))?( duration %{TIME:duration} bytes %{INT:bytes})?(?: %{CISCO_REASON:reason})?( \(%{DATA:user}\))?
"""ASA-6-302020, ASA-6-302021"""
CISCOFW302020_302021 %{CISCO_ACTION:action}(?: %{CISCO_DIRECTION:direction})? %{WORD:protocol} connection for faddr %{IP:dst_ip}/%{INT:icmp_seq_num}(?:\(%{DATA:fwuser}\))? gaddr %{IP:src_xlated_ip}/%{INT:icmp_code_xlated} laddr %{IP:src_ip}/%{INT:icmp_code}( \(%{DATA:user}\))?
"""ASA-6-305011"""
CISCOFW305011 %{CISCO_ACTION:action} %{CISCO_XLATE_TYPE:xlate_type} %{WORD:protocol} translation from %{DATA:src_interface}:%{IP:src_ip}(/%{INT:src_port})?( \(%{DATA:src_fwuser}\))? to %{DATA:src_xlated_interface}:%{IP:src_xlated_ip}/%{DATA:src_xlated_port}
"""ASA-3-313001, ASA-3-313004, ASA-3-313008"""
CISCOFW313001_313004_313008 %{CISCO_ACTION:action} %{WORD:protocol} type=%{INT:icmp_type}, code=%{INT:icmp_code} from %{IP:src_ip} on interface %{DATA:interface}( to %{IP:dst_ip})?
"""ASA-4-313005"""
CISCOFW313005 %{CISCO_REASON:reason} for %{WORD:protocol} error message: %{WORD:err_protocol} src %{DATA:err_src_interface}:%{IP:err_src_ip}(\(%{DATA:err_src_fwuser}\))? dst %{DATA:err_dst_interface}:%{IP:err_dst_ip}(\(%{DATA:err_dst_fwuser}\))? \(type %{INT:err_icmp_type}, code %{INT:err_icmp_code}\) on %{DATA:interface} interface\.  Original IP payload: %{WORD:protocol} src %{IP:orig_src_ip}/%{INT:orig_src_port}(\(%{DATA:orig_src_fwuser}\))? dst %{IP:orig_dst_ip}/%{INT:orig_dst_port}(\(%{DATA:orig_dst_fwuser}\))?
"""ASA-5-321001"""
```

```
CISCOFW321001 Resource '%{WORD:resource_name}' limit of %{POSINT:resource_limit} reached for system
"""ASA-4-402117"""
CISCOFW402117 %{WORD:protocol}: Received a non-IPSec packet \(protocol= %{WORD:orig_protocol}\) from %{IP:src_ip} to %{IP:dst_ip}
"""ASA-4-402119"""
CISCOFW402119 %{WORD:protocol}: Received an %{WORD:orig_protocol} packet \(SPI= %{DATA:spi}, sequence number= %{DATA:seq_num}\) from %{IP:src_ip} \(user= %{DATA:user}\) to %{IP:dst_ip} that failed anti-replay checking
"""ASA-4-419001"""
CISCOFW419001 %{CISCO_ACTION:action} %{WORD:protocol} packet from %{DATA:src_interface}:%{IP:src_ip}/%{INT:src_port} to %{DATA:dst_interface}:%{IP:dst_ip}/%{INT:dst_port}, reason: %{GREEDYDATA:reason}
"""ASA-4-419002"""
CISCOFW419002 %{CISCO_REASON:reason} from %{DATA:src_interface}:%{IP:src_ip}/%{INT:src_port} to %{DATA:dst_interface}:%{IP:dst_ip}/%{INT:dst_port} with different initial sequence number
"""ASA-4-500004"""
CISCOFW500004 %{CISCO_REASON:reason} for protocol=%{WORD:protocol}, from %{IP:src_ip}/%{INT:src_port} to %{IP:dst_ip}/%{INT:dst_port}
"""ASA-6-602303, ASA-6-602304"""
CISCOFW602303_602304 %{WORD:protocol}: An %{CISCO_DIRECTION:direction} %{GREEDYDATA:tunnel_type} SA \(SPI= %{DATA:spi}\) between %{IP:src_ip} and %{IP:dst_ip} \(user= %{DATA:user}\) has been %{CISCO_ACTION:action}
"""ASA-7-710001, ASA-7-710002, ASA-7-710003, ASA-7-710005, ASA-7-710006"""
CISCOFW710001_710002_710003_710005_710006 %{WORD:protocol} (?:request|access) %{CISCO_ACTION:action} from %{IP:src_ip}/%{INT:src_port} to %{DATA:dst_interface}:%{IP:dst_ip}/%{INT:dst_port}
"""ASA-6-713172"""
CISCOFW713172 Group = %{GREEDYDATA:group}, IP = %{IP:src_ip}, Automatic NAT Detection Status:\s+Remote end\s*%{DATA:is_remote_natted}\s*behind a NAT device\s+This\s+end\s*%{DATA:is_local_natted}\s*behind a NAT device
"""ASA-4-733100"""
CISCOFW733100 \[\s*%{DATA:drop_type}\s*\] drop %{DATA:drop_rate_id} exceeded. Current burst rate is %{INT:drop_rate_current_burst} per second, max configured rate is %{INT:drop_rate_max_burst}; Current average rate is %{INT:drop_rate_current_avg} per second, max configured rate is %{INT:drop_rate_max_avg}; Cumulative total count is %{INT:drop_total_count}
"""== End Cisco ASA =="""
"""Shorewall firewall logs"""
SHOREWALL (%{SYSLOGTIMESTAMP:timestamp}) (%{WORD:nf_host}) kernel:. *Shorewall:(%{WORD:nf_action1})?:(%{WORD:nf_action2})?. *IN=(%{USERNAME:nf_in_interface})?. *(OUT= *MAC=(%{COMMONMAC:nf_dst_mac}):(%{COMMONMAC:nf_src_mac})? |OUT=%{USERNAME:nf_out_interface}). *SRC=(%{IPV4:nf_src_ip}). *DST=(%{IPV4:nf_dst_ip}). *LEN=(%{WORD:nf_len}).? *TOS=(%{WORD:nf_tos}).? *PREC=(%{WORD:nf_prec}).? *TTL=(%{INT:nf_ttl}).? *ID=(%{INT:nf_id}).? *PROTO=(%{WORD:nf_protocol}).? *SPT=(%{INT:nf_src_port}?. *DPT=%{INT:nf_dst_port}?. *)
"""== End Shorewall"""
```

Sample patterns for HAProxy

```
""" These patterns were tested w/ haproxy-1.4.15"""
""" Documentation of the haproxy log formats can be found at the following links:"""
""" http://code.google.com/p/haproxy-docs/wiki/HTTPLogFormat"""
""" http://code.google.com/p/haproxy-docs/wiki/TCPLogFormat"""
HAPROXYTIME (?! <[0-9])%{HOUR:haproxy_hour}:%{MINUTE:haproxy_minute}(?::%{SECOND:haproxy_se
cond})(?![ 0-9])
HAPROXYDATE %{MONTHDAY:haproxy_monthday}/%{MONTH:haproxy_month}/%{YEAR:haproxy_year}:%{HAPR
OXYTIME:haproxy_time}. %{INT:haproxy_milliseconds}
""" Override these default patterns to parse out what is captured in your haproxy.cfg"""
HAPROXYCAPTUREDREQUESTHEADERS %{DATA:captured_request_headers}
HAPROXYCAPTUREDRESPONSEHEADERS %{DATA:captured_response_headers}
""" Example:"""
"""  These haproxy config lines will add data to the logs that are captured"""
"""  by the patterns below. Place them in your custom patterns directory to"""
"""  override the defaults."""
"""
"""  capture request header Host len 40"""
"""  capture request header X-Forwarded-For len 50"""
"""  capture request header Accept-Language len 50"""
"""  capture request header Referer len 200"""
"""  capture request header User-Agent len 200"""
"""
"""  capture response header Content-Type len 30"""
"""  capture response header Content-Encoding len 10"""
"""  capture response header Cache-Control len 200"""
"""  capture response header Last-Modified len 200"""
"""parse a haproxy 'httplog' line"""
HAPROXYHTTPBASE %{IP:client_ip}:%{INT:client_port} \[%{HAPROXYDATE:accept_date}\] %{NOTSPAC
E:frontend_name} %{NOTSPACE:backend_name}/%{NOTSPACE:server_name} %{INT:time_request}/%{INT
:time_queue}/%{INT:time_backend_connect}/%{INT:time_backend_response}/%{NOTSPACE:time_durat
ion} %{INT:http_status_code} %{NOTSPACE:bytes_read} %{DATA:captured_request_cookie} %{DATA:
captured_response_cookie} %{NOTSPACE:termination_state} %{INT:actconn}/%{INT:feconn}/%{INT:
beconn}/%{INT:srvconn}/%{NOTSPACE:retries} %{INT:srv_queue}/%{INT:backend_queue} (\{%{HAPRO
XYCAPTUREDREQUESTHEADERS}\})?( )?( \{%{HAPROXYCAPTUREDRESPONSEHEADERS}\})?( )?"( <BADREQ>|(
%{WORD:http_verb} (%{URIPROTO:http_proto}://)?(?:%{USER:http_user}(?::[^@]*)? @)?(?:%{URIHO
ST:http_host})?(?:%{URIPATHPARAM:http_request})?( HTTP/%{NUMBER:http_version})?))?"
HAPROXYHTTP (?:%{SYSLOGTIMESTAMP:syslog_timestamp}|%{TIMESTAMP_ISO8601:timestamp8601}) %{IP
ORHOST:syslog_server} %{SYSLOGPROG}: %{HAPROXYHTTPBASE}
"""parse a haproxy 'tcplog' line"""
HAPROXYTCP (?:%{SYSLOGTIMESTAMP:syslog_timestamp}|%{TIMESTAMP_ISO8601:timestamp8601}) %{IPO
RHOST:syslog_server} %{SYSLOGPROG}: %{IP:client_ip}:%{INT:client_port} \[%{HAPROXYDATE:acce
pt_date}\] %{NOTSPACE:frontend_name} %{NOTSPACE:backend_name}/%{NOTSPACE:server_name} %{INT
:time_queue}/%{INT:time_backend_connect}/%{NOTSPACE:time_duration} %{NOTSPACE:bytes_read} %
{NOTSPACE:termination_state} %{INT:actconn}/%{INT:feconn}/%{INT:beconn}/%{INT:srvconn}/%{NO
TSPACE:retries} %{INT:srv_queue}/%{INT:backend_queue}
```

Sample patterns for Java

```
JAVACLASS (?:[a-zA-Z$_][a-zA-Z$_0-9]*\.)*[a-zA-Z$_][a-zA-Z$_0-9]*
"""Space is an allowed character to match special cases like 'Native Method' or 'Unknown Source'"""
JAVAFILE (?:[A-Za-z0-9_. -]+)
"""Allow special <init> method"""
JAVAMETHOD (?:(<init>)|[a-zA-Z$_][a-zA-Z$_0-9]*)
"""Line number is optional in special cases 'Native method' or 'Unknown source'"""
JAVASTACKTRACEPART %{SPACE}at %{JAVACLASS:class}\. %{JAVAMETHOD:method}\(%{JAVAFILE:file}(?::%{NUMBER:line})? \)
""" Java Logs"""
JAVATHREAD (?:[A-Z]{2}-Processor[\d]+)
JAVACLASS (?:[a-zA-Z0-9-]+\.) +[A-Za-z0-9$]+
JAVAFILE (?:[A-Za-z0-9_. -]+)
JAVASTACKTRACEPART at %{JAVACLASS:class}\. %{WORD:method}\(%{JAVAFILE:file}:%{NUMBER:line}\)
JAVALOGMESSAGE (. *)
""" MMM dd, yyyy HH:mm:ss eg: Jan 9, 2014 7:13:13 AM"""
CATALINA_DATESTAMP %{MONTH} %{MONTHDAY}, 20%{YEAR} %{HOUR}:? %{MINUTE}(?::? %{SECOND}) (?:AM|PM)
""" yyyy-MM-dd HH:mm:ss,SSS ZZZ eg: 2014-01-09 17:32:25,527 -0800"""
TOMCAT_DATESTAMP 20%{YEAR}-%{MONTHNUM}-%{MONTHDAY} %{HOUR}:? %{MINUTE}(?::? %{SECOND}) %{ISO8601_TIMEZONE}
CATALINALOG %{CATALINA_DATESTAMP:timestamp} %{JAVACLASS:class} %{JAVALOGMESSAGE:logmessage}
""" 2014-01-09 20:03:28,269 -0800 | ERROR | com.example.service.ExampleService - something compeletely unexpected happened..."""
TOMCATLOG %{TOMCAT_DATESTAMP:timestamp} \| %{LOGLEVEL:level} \| %{JAVACLASS:class} - %{JAVALOGMESSAGE:logmessage}
```

Sample patterns for Junos

```
"""JUNOS 11.4 RT_FLOW patterns"""
RT_FLOW_EVENT (RT_FLOW_SESSION_CREATE|RT_FLOW_SESSION_CLOSE|RT_FLOW_SESSION_DENY)
RT_FLOW1 %{RT_FLOW_EVENT:event}: %{GREEDYDATA:close-reason}: %{IP:src-ip}/%{INT:src-port}->%{IP:dst-ip}/%{INT:dst-port} %{DATA:service} %{IP:nat-src-ip}/%{INT:nat-src-port}->%{IP:nat-dst-ip}/%{INT:nat-dst-port} %{DATA:src-nat-rule-name} %{DATA:dst-nat-rule-name} %{INT:protocol-id} %{DATA:policy-name} %{DATA:from-zone} %{DATA:to-zone} %{INT:session-id} \d+\(%{DATA:sent}\) \d+\(%{DATA:received}\) %{INT:elapsed-time} . *
RT_FLOW2 %{RT_FLOW_EVENT:event}: session created %{IP:src-ip}/%{INT:src-port}->%{IP:dst-ip}/%{INT:dst-port} %{DATA:service} %{IP:nat-src-ip}/%{INT:nat-src-port}->%{IP:nat-dst-ip}/%{INT:nat-dst-port} %{DATA:src-nat-rule-name} %{DATA:dst-nat-rule-name} %{INT:protocol-id} %{DATA:policy-name} %{DATA:from-zone} %{DATA:to-zone} %{INT:session-id} . *
RT_FLOW3 %{RT_FLOW_EVENT:event}: session denied %{IP:src-ip}/%{INT:src-port}->%{IP:dst-ip}/%{INT:dst-port} %{DATA:service} %{INT:protocol-id}\(\d\) %{DATA:policy-name} %{DATA:from-zone} %{DATA:to-zone} . *
```

Sample patterns for Linux Syslog

```
SYSLOG5424PRINTASCII [! -~]+
SYSLOGBASE2 (?:%{SYSLOGTIMESTAMP:timestamp}|%{TIMESTAMP_ISO8601:timestamp8601}) (?:%{SYSLOG
FACILITY} )? %{SYSLOGHOST:logsource}+(?: %{SYSLOGPROG}:|)
SYSLOGPAMSESSION %{SYSLOGBASE} (? =%{GREEDYDATA:message})%{WORD:pam_module}\(%{DATA:pam_cal
ler}\): session %{WORD:pam_session_state} for user %{USERNAME:username}(?: by %{GREEDYDATA:
pam_by})?
CRON_ACTION [A-Z ]+
CRONLOG %{SYSLOGBASE} \(%{USER:user}\) %{CRON_ACTION:action} \(%{DATA:message}\)
SYSLOGLINE %{SYSLOGBASE2} %{GREEDYDATA:message}
"""IETF 5424 syslog(8) format (see http://www.rfc-editor.org/info/rfc5424)"""
SYSLOG5424PRI <%{NONNEGINT:syslog5424_pri}>
SYSLOG5424SD \[%{DATA}\]+
SYSLOG5424BASE %{SYSLOG5424PRI}%{NONNEGINT:syslog5424_ver} +(?:%{TIMESTAMP_ISO8601:syslog54
24_ts}|-) +(?:%{HOSTNAME:syslog5424_host}|-) +(-|%{SYSLOG5424PRINTASCII:syslog5424_app}) +(
-|%{SYSLOG5424PRINTASCII:syslog5424_proc}) +(-|%{SYSLOG5424PRINTASCII:syslog5424_msgid}) +(
?:%{SYSLOG5424SD:syslog5424_sd}|-|)
SYSLOG5424LINE %{SYSLOG5424BASE} +%{GREEDYDATA:syslog5424_msg}
```

Sample patterns for Mcollective

```
"""Remember, these can be multi-line events."""
MCOLLECTIVE ., \[%{TIMESTAMP_ISO8601:timestamp} #%{POSINT:pid}\]%{SPACE}%{LOGLEVEL:event_le
vel}
MCOLLECTIVEAUDIT %{TIMESTAMP_ISO8601:timestamp}:
```

Sample patterns for MongoDB

```
MONGO_LOG %{SYSLOGTIMESTAMP:timestamp} \[%{WORD:component}\] %{GREEDYDATA:message}
MONGO_QUERY \{ (? <={ ). *(? = } ntoreturn:) \}
MONGO_SLOWQUERY %{WORD} %{MONGO_WORDDASH:database}\. %{MONGO_WORDDASH:collection} %{WORD}:
%{MONGO_QUERY:query} %{WORD}:%{NONNEGINT:ntoreturn} %{WORD}:%{NONNEGINT:ntoskip} %{WORD}:%{
NONNEGINT:nscanned}.*nreturned:%{NONNEGINT:nreturned}.. + (? <duration>[0-9]+)ms
MONGO_WORDDASH \b[\w-]+\b
MONGO3_SEVERITY \w
MONGO3_COMPONENT %{WORD}|-
MONGO3_LOG %{TIMESTAMP_ISO8601:timestamp} %{MONGO3_SEVERITY:severity} %{MONGO3_COMPONENT:co
mponent}%{SPACE}(?:\[%{DATA:context}\])? %{GREEDYDATA:message}
```

Sample patterns for Nagios

```
NAGIOSTIME \[%{NUMBER:nagios_epoch}\]
"""nagios log types"""
NAGIOS_TYPE_CURRENT_SERVICE_STATE CURRENT SERVICE STATE
NAGIOS_TYPE_CURRENT_HOST_STATE CURRENT HOST STATE
NAGIOS_TYPE_SERVICE_NOTIFICATION SERVICE NOTIFICATION
NAGIOS_TYPE_HOST_NOTIFICATION HOST NOTIFICATION
NAGIOS_TYPE_SERVICE_ALERT SERVICE ALERT
NAGIOS_TYPE_HOST_ALERT HOST ALERT
NAGIOS_TYPE_SERVICE_FLAPPING_ALERT SERVICE FLAPPING ALERT
NAGIOS_TYPE_HOST_FLAPPING_ALERT HOST FLAPPING ALERT
NAGIOS_TYPE_SERVICE_DOWNTIME_ALERT SERVICE DOWNTIME ALERT
```

```
NAGIOS_TYPE_HOST_DOWNTIME_ALERT HOST DOWNTIME ALERT
NAGIOS_TYPE_PASSIVE_SERVICE_CHECK PASSIVE SERVICE CHECK
NAGIOS_TYPE_PASSIVE_HOST_CHECK PASSIVE HOST CHECK
NAGIOS_TYPE_SERVICE_EVENT_HANDLER SERVICE EVENT HANDLER
NAGIOS_TYPE_HOST_EVENT_HANDLER HOST EVENT HANDLER
NAGIOS_TYPE_EXTERNAL_COMMAND EXTERNAL COMMAND
NAGIOS_TYPE_TIMEPERIOD_TRANSITION TIMEPERIOD TRANSITION
"""external check types"""
NAGIOS_EC_DISABLE_SVC_CHECK DISABLE_SVC_CHECK
NAGIOS_EC_ENABLE_SVC_CHECK ENABLE_SVC_CHECK
NAGIOS_EC_DISABLE_HOST_CHECK DISABLE_HOST_CHECK
NAGIOS_EC_ENABLE_HOST_CHECK ENABLE_HOST_CHECK
NAGIOS_EC_PROCESS_SERVICE_CHECK_RESULT PROCESS_SERVICE_CHECK_RESULT
NAGIOS_EC_PROCESS_HOST_CHECK_RESULT PROCESS_HOST_CHECK_RESULT
NAGIOS_EC_SCHEDULE_SERVICE_DOWNTIME SCHEDULE_SERVICE_DOWNTIME
NAGIOS_EC_SCHEDULE_HOST_DOWNTIME SCHEDULE_HOST_DOWNTIME
NAGIOS_EC_DISABLE_HOST_SVC_NOTIFICATIONS DISABLE_HOST_SVC_NOTIFICATIONS
NAGIOS_EC_ENABLE_HOST_SVC_NOTIFICATIONS ENABLE_HOST_SVC_NOTIFICATIONS
NAGIOS_EC_DISABLE_HOST_NOTIFICATIONS DISABLE_HOST_NOTIFICATIONS
NAGIOS_EC_ENABLE_HOST_NOTIFICATIONS ENABLE_HOST_NOTIFICATIONS
NAGIOS_EC_DISABLE_SVC_NOTIFICATIONS DISABLE_SVC_NOTIFICATIONS
NAGIOS_EC_ENABLE_SVC_NOTIFICATIONS ENABLE_SVC_NOTIFICATIONS
NAGIOS_WARNING Warning:%{SPACE}%{GREEDYDATA:nagios_message}
NAGIOS_CURRENT_SERVICE_STATE %{NAGIOS_TYPE_CURRENT_SERVICE_STATE:nagios_type}: %{DATA:nagios_hostname};%{DATA:nagios_service};%{DATA:nagios_state};%{DATA:nagios_statetype};%{DATA:nagios_statecode};%{GREEDYDATA:nagios_message}
NAGIOS_CURRENT_HOST_STATE %{NAGIOS_TYPE_CURRENT_HOST_STATE:nagios_type}: %{DATA:nagios_hostname};%{DATA:nagios_state};%{DATA:nagios_statetype};%{DATA:nagios_statecode};%{GREEDYDATA:nagios_message}
NAGIOS_SERVICE_NOTIFICATION %{NAGIOS_TYPE_SERVICE_NOTIFICATION:nagios_type}: %{DATA:nagios_notifyname};%{DATA:nagios_hostname};%{DATA:nagios_service};%{DATA:nagios_state};%{DATA:nagios_contact};%{GREEDYDATA:nagios_message}
NAGIOS_HOST_NOTIFICATION %{NAGIOS_TYPE_HOST_NOTIFICATION:nagios_type}: %{DATA:nagios_notifyname};%{DATA:nagios_hostname};%{DATA:nagios_state};%{DATA:nagios_contact};%{GREEDYDATA:nagios_message}
NAGIOS_SERVICE_ALERT %{NAGIOS_TYPE_SERVICE_ALERT:nagios_type}: %{DATA:nagios_hostname};%{DATA:nagios_service};%{DATA:nagios_state};%{DATA:nagios_statelevel};%{NUMBER:nagios_attempt};%{GREEDYDATA:nagios_message}
NAGIOS_HOST_ALERT %{NAGIOS_TYPE_HOST_ALERT:nagios_type}: %{DATA:nagios_hostname};%{DATA:nagios_state};%{DATA:nagios_statelevel};%{NUMBER:nagios_attempt};%{GREEDYDATA:nagios_message}
NAGIOS_SERVICE_FLAPPING_ALERT %{NAGIOS_TYPE_SERVICE_FLAPPING_ALERT:nagios_type}: %{DATA:nagios_hostname};%{DATA:nagios_service};%{DATA:nagios_state};%{GREEDYDATA:nagios_message}
NAGIOS_HOST_FLAPPING_ALERT %{NAGIOS_TYPE_HOST_FLAPPING_ALERT:nagios_type}: %{DATA:nagios_hostname};%{DATA:nagios_state};%{GREEDYDATA:nagios_message}
NAGIOS_SERVICE_DOWNTIME_ALERT %{NAGIOS_TYPE_SERVICE_DOWNTIME_ALERT:nagios_type}: %{DATA:nagios_hostname};%{DATA:nagios_service};%{DATA:nagios_state};%{GREEDYDATA:nagios_comment}
NAGIOS_HOST_DOWNTIME_ALERT %{NAGIOS_TYPE_HOST_DOWNTIME_ALERT:nagios_type}: %{DATA:nagios_hostname};%{DATA:nagios_state};%{GREEDYDATA:nagios_comment}
NAGIOS_PASSIVE_SERVICE_CHECK %{NAGIOS_TYPE_PASSIVE_SERVICE_CHECK:nagios_type}: %{DATA:nagios_hostname};%{DATA:nagios_service};%{DATA:nagios_state};%{GREEDYDATA:nagios_comment}
NAGIOS_PASSIVE_HOST_CHECK %{NAGIOS_TYPE_PASSIVE_HOST_CHECK:nagios_type}: %{DATA:nagios_hostname};%{DATA:nagios_state};%{GREEDYDATA:nagios_comment}
NAGIOS_SERVICE_EVENT_HANDLER %{NAGIOS_TYPE_SERVICE_EVENT_HANDLER:nagios_type}: %{DATA:nagios_hostname};%{DATA:nagios_service};%{DATA:nagios_state};%{DATA:nagios_statelevel};%{DATA:nagios_event_handler_name}
```

```
NAGIOS_HOST_EVENT_HANDLER %{NAGIOS_TYPE_HOST_EVENT_HANDLER:nagios_type}: %{DATA:nagios_hostname};%{DATA:nagios_state};%{DATA:nagios_statelevel};%{DATA:nagios_event_handler_name}
NAGIOS_TIMEPERIOD_TRANSITION %{NAGIOS_TYPE_TIMEPERIOD_TRANSITION:nagios_type}: %{DATA:nagios_service};%{DATA:nagios_unknown1};%{DATA:nagios_unknown2}
"""Disable host & service check"""
NAGIOS_EC_LINE_DISABLE_SVC_CHECK %{NAGIOS_TYPE_EXTERNAL_COMMAND:nagios_type}: %{NAGIOS_EC_DISABLE_SVC_CHECK:nagios_command};%{DATA:nagios_hostname};%{DATA:nagios_service}
NAGIOS_EC_LINE_DISABLE_HOST_CHECK %{NAGIOS_TYPE_EXTERNAL_COMMAND:nagios_type}: %{NAGIOS_EC_DISABLE_HOST_CHECK:nagios_command};%{DATA:nagios_hostname}
"""Enable host & service check"""
NAGIOS_EC_LINE_ENABLE_SVC_CHECK %{NAGIOS_TYPE_EXTERNAL_COMMAND:nagios_type}: %{NAGIOS_EC_ENABLE_SVC_CHECK:nagios_command};%{DATA:nagios_hostname};%{DATA:nagios_service}
NAGIOS_EC_LINE_ENABLE_HOST_CHECK %{NAGIOS_TYPE_EXTERNAL_COMMAND:nagios_type}: %{NAGIOS_EC_ENABLE_HOST_CHECK:nagios_command};%{DATA:nagios_hostname}
"""Process host & service check"""
NAGIOS_EC_LINE_PROCESS_SERVICE_CHECK_RESULT %{NAGIOS_TYPE_EXTERNAL_COMMAND:nagios_type}: %{NAGIOS_EC_PROCESS_SERVICE_CHECK_RESULT:nagios_command};%{DATA:nagios_hostname};%{DATA:nagios_service};%{DATA:nagios_state};%{GREEDYDATA:nagios_check_result}
NAGIOS_EC_LINE_PROCESS_HOST_CHECK_RESULT %{NAGIOS_TYPE_EXTERNAL_COMMAND:nagios_type}: %{NAGIOS_EC_PROCESS_HOST_CHECK_RESULT:nagios_command};%{DATA:nagios_hostname};%{DATA:nagios_state};%{GREEDYDATA:nagios_check_result}
"""Disable host & service notifications"""
NAGIOS_EC_LINE_DISABLE_HOST_SVC_NOTIFICATIONS %{NAGIOS_TYPE_EXTERNAL_COMMAND:nagios_type}: %{NAGIOS_EC_DISABLE_HOST_SVC_NOTIFICATIONS:nagios_command};%{GREEDYDATA:nagios_hostname}
NAGIOS_EC_LINE_DISABLE_HOST_NOTIFICATIONS %{NAGIOS_TYPE_EXTERNAL_COMMAND:nagios_type}: %{NAGIOS_EC_DISABLE_HOST_NOTIFICATIONS:nagios_command};%{GREEDYDATA:nagios_hostname}
NAGIOS_EC_LINE_DISABLE_SVC_NOTIFICATIONS %{NAGIOS_TYPE_EXTERNAL_COMMAND:nagios_type}: %{NAGIOS_EC_DISABLE_SVC_NOTIFICATIONS:nagios_command};%{DATA:nagios_hostname};%{GREEDYDATA:nagios_service}
"""Enable host & service notifications"""
NAGIOS_EC_LINE_ENABLE_HOST_SVC_NOTIFICATIONS %{NAGIOS_TYPE_EXTERNAL_COMMAND:nagios_type}: %{NAGIOS_EC_ENABLE_HOST_SVC_NOTIFICATIONS:nagios_command};%{GREEDYDATA:nagios_hostname}
NAGIOS_EC_LINE_ENABLE_HOST_NOTIFICATIONS %{NAGIOS_TYPE_EXTERNAL_COMMAND:nagios_type}: %{NAGIOS_EC_ENABLE_HOST_NOTIFICATIONS:nagios_command};%{GREEDYDATA:nagios_hostname}
NAGIOS_EC_LINE_ENABLE_SVC_NOTIFICATIONS %{NAGIOS_TYPE_EXTERNAL_COMMAND:nagios_type}: %{NAGIOS_EC_ENABLE_SVC_NOTIFICATIONS:nagios_command};%{DATA:nagios_hostname};%{GREEDYDATA:nagios_service}
"""Schedule host & service downtime"""
NAGIOS_EC_LINE_SCHEDULE_HOST_DOWNTIME %{NAGIOS_TYPE_EXTERNAL_COMMAND:nagios_type}: %{NAGIOS_EC_SCHEDULE_HOST_DOWNTIME:nagios_command};%{DATA:nagios_hostname};%{NUMBER:nagios_start_time};%{NUMBER:nagios_end_time};%{NUMBER:nagios_fixed};%{NUMBER:nagios_trigger_id};%{NUMBER:nagios_duration};%{DATA:author};%{DATA:comment}
"""End matching line"""
  NAGIOSLOGLINE %{NAGIOSTIME} (?:%{NAGIOS_WARNING}|%{NAGIOS_CURRENT_SERVICE_STATE}|%{NAGIOS_CURRENT_HOST_STATE}|%{NAGIOS_SERVICE_NOTIFICATION}|%{NAGIOS_HOST_NOTIFICATION}|%{NAGIOS_SERVICE_ALERT}|%{NAGIOS_HOST_ALERT}|%{NAGIOS_SERVICE_FLAPPING_ALERT}|%{NAGIOS_HOST_FLAPPING_ALERT}|%{NAGIOS_SERVICE_DOWNTIME_ALERT}|%{NAGIOS_HOST_DOWNTIME_ALERT}|%{NAGIOS_PASSIVE_SERVICE_CHECK}|%{NAGIOS_PASSIVE_HOST_CHECK}|%{NAGIOS_SERVICE_EVENT_HANDLER}|%{NAGIOS_HOST_EVENT_HANDLER}|%{NAGIOS_TIMEPERIOD_TRANSITION}|%{NAGIOS_EC_LINE_DISABLE_SVC_CHECK}|%{NAGIOS_EC_LINE_ENABLE_SVC_CHECK}|%{NAGIOS_EC_LINE_DISABLE_HOST_CHECK}|%{NAGIOS_EC_LINE_ENABLE_HOST_CHECK}|%{NAGIOS_EC_LINE_PROCESS_HOST_CHECK_RESULT}|%{NAGIOS_EC_LINE_PROCESS_SERVICE_CHECK_RESULT}|%{NAGIOS_EC_LINE_SCHEDULE_HOST_DOWNTIME}|%{NAGIOS_EC_LINE_DISABLE_HOST_SVC_NOTIFICATIONS}|%{NAGIOS_EC_LINE_ENABLE_HOST_SVC_NOTIFICATIONS}|%{NAGIOS_EC_LINE_DISABLE_HOST_NOTIFICATIONS}|%{NAGIOS_EC_LINE_ENABLE_HOST_NOTIFICATIONS}|%{NAGIOS_EC_LINE_DISABLE_SVC_NOTIFICATIONS}
```

```
|%{NAGIOS_EC_LINE_ENABLE_SVC_NOTIFICATIONS})
```

Sample patterns for PostgreSQL

```
"""Default postgresql pg_log format pattern"""
POSTGRESQL %{DATESTAMP:timestamp} %{TZ} %{DATA:user_id} %{GREEDYDATA:connection_id} %{POSIN
T:pid}
```

Sample patterns for Rails

```
RUUID \h{32}
"""rails controller with action"""
RCONTROLLER (? <controller>[^#]+)#(? <action>\w+)
"""this will often be the only line:"""
RAILS3HEAD (? m)Started %{WORD:verb} "%{URIPATHPARAM:request}" for %{IPORHOST:clientip} at
(? <timestamp>%{YEAR}-%{MONTHNUM}-%{MONTHDAY} %{HOUR}:%{MINUTE}:%{SECOND} %{ISO8601_TIMEZON
E})
"""for some a strange reason, params are stripped of {} - not sure that's a good idea."""
RPROCESSING \W*Processing by %{RCONTROLLER} as (? <format>\S+)(?:\W*Parameters: {%{DATA:par
ams}}\W*)?
RAILS3FOOT Completed %{NUMBER:response}%{DATA} in %{NUMBER:totalms}ms %{RAILS3PROFILE}%{GRE
EDYDATA}
RAILS3PROFILE (?:\(Views: %{NUMBER:viewms}ms \| ActiveRecord: %{NUMBER:activerecordms}ms|\(
ActiveRecord: %{NUMBER:activerecordms}ms)?
"""putting it all together"""
RAILS3 %{RAILS3HEAD}(?:%{RPROCESSING})?(? <context>(?:%{DATA}\n)*)(?:%{RAILS3FOOT})?
```

Sample patterns for Redis

```
REDISTIMESTAMP %{MONTHDAY} %{MONTH} %{TIME}
REDISLOG \[%{POSINT:pid}\] %{REDISTIMESTAMP:timestamp} \*
```

Sample patterns for Ruby

```
RUBY_LOGLEVEL (?:DEBUG|FATAL|ERROR|WARN|INFO)
RUBY_LOGGER [DFEWI], \[%{TIMESTAMP_ISO8601:timestamp} #%{POSINT:pid}\] *%{RUBY_LOGLEVEL:log
level} -- +%{DATA:progname}: %{GREEDYDATA:message}
```

12.7.6. JMESPath syntax

This topic describes the common JMESPath syntax.

JMESPath is an enhanced query and compute language for JSON. It can be used to extract, compute, and convert JSON data. For more information, see JMESPath Tutorial.

The data transformation feature provides the `json_select`, `e_json`, and `e_split` functions. You can use JMESPath in these functions to extract values of fields or JSON expressions and compute specific values. Example:

```
json_select(Value, "JMESPath expression", ...)
e_json(Field, jmes="JMESPath expression", ...)
e_split(Field, ... jmes="JMESPath expression", ...)
```

For more information about how to use these functions, see json_select, e_json, and e_split.

Extract a value by using a key

- Raw log entry:

```
"json_data":{
   "a": "foo",
   "b": "bar",
   "c": "baz"
 }
```

- JMESPath syntax:

```
json_select(v("json_data"), "a") # Return value: foo.
json_select(v("json_data"), "b") # Return value: bar.
json_select(v("json_data"), "c") # Return value: baz.
```

Extract a nested value

- Raw log entry:

```
"json_data":{"a":
               {"b":
                 {"c":
                   {"d":"value"}
               }
             }
         }
```

- JMESPath syntax:

```
json_select(v("json_data"), "a.b.c.d") # Return value: value.
```

Extract values by using slicing data

- Raw log entry:

```
"json_data":{
   "a": ["b", "c", "d", "e", "f"]
 }
```

- JMESPath syntax:

```
json_select(v("json_data"), "a[2: ]") # Return value: ["d", "e", "f"].
```

Extract a value by combining the preceding methods

- Raw log entry:

```
"json_data":{
  "a": {
    "b": {
      "c": [{"d": [0, [1, 2]]}, {"d": [3, 4]}]
    }
  }
}
```

- JMESPath syntax:

```
json_select(v("json_data"), "a.b.c[0].d[1][0]") # Return value: 1.
```

Extract values by using projection

- Raw log entry 1:

```
"json_data":{
    "people": [
      {"first": "James", "last": "d"},
      {"first": "Jacob", "last": "e"},
      {"first": "Jayden", "last": "f"},
      {"missing": "different"}
    ],
    "foo": {"bar": "baz"}
}
```

- JMESPath syntax 1:

```
json_select(v("json_data"), "people[*].first") # Return value: ["James","Jacob","Jayden"]
.
```

- Raw log entry 2:

```
"json_data":{
    "ops": {
    "functionA": {"numArgs": 2},
    "functionB": {"numArgs": 3},
    "functionC": {"variadic": true}
    }
  }
```

- JMESPath syntax 2:

```
json_select(v("json_data"), "ops. *.numArgs") # Return value: [2, 3].
```

- Raw log entry 3:

```
"json_data":{
  "machines": [
    {"name": "a", "state": "running"},
    {"name": "b", "state": "stopped"},
    {"name": "c", "state": "running"}
  ]
}
```

- JMESPath syntax 3:

```
json_select(v("json_data"), "machines[? state=='running'].name") # Return value: ["a", "c
"].
```

Extract values from a list

- Raw log entry:

```
"json_data":{
  "people": [
    {
    "name": "a",
    "state": {"name": "up"}
    },
    {
    "name": "b",
    "state": {"name": "down"}
    }
  ]
}
```

- JMESPath syntax:

```
json_select(v("json_data"), "people[].[name, state.name]") # Return values: [["a","up"],[
"b","down"]]
```

Compute the length of an array

- Raw log entry:

```
"json_data":{
   "a": ["b", "c", "d", "e", "f"]
 }
```

- JMESPath syntax:

```
json_select(v("json_data"), "length(a)") # Return value: 5.
```

```
# Set the no-empty field to true if the length of array a is greater than 0.
e_if(json_select(v("json_data"), "length(a)", default=0), e_set("no-empty", true))
```

12.7.7. Date and time formatting directives

The ANSI C standard defines a set of directives used to parse and format date and time strings.

Log Service supports all the directives defined in the ANSI C (1989 version) standard. The following table describes these directives and provides specific examples and notes.

Directive	Description	Example	Note
%a	The abbreviation of the weekday.	`Sun, ..., Mon`	Currently, date and time strings are displayed in the en-US locale. Other locales are not supported.

Directive	Description	Example	Note
%A	The full name of the weekday.	`Sunday, ..., Monday`	Currently, date and time strings are displayed in the en-US locale. Other locales are not supported.
%w	The weekday represented as a decimal number. 0 indicates Sunday, and 6 indicates Saturday.	`0, 1, 2, 3, 4, 5, 6`	None.
%d	The day of the month represented as a zero-padded decimal number.	`01, 02, ..., 31`	The leading zero is optional for formatting directives `%d, %m, %H, %I, %M, %S, %J, %U, %W, %V, %y` when they are used to parse date and time strings.
%b	The abbreviation of the month.	`Jan, Feb, ..., Dec`	Currently, date and time strings are displayed in the en-US locale. Other locales are not supported.
%B	The full name of the month.	`January, February, ..., December`	The leading zero is optional for formatting directives `%d, %m, %H, %I, %M, %S, %J, %U, %W, %V, %y` when they are used to parse date and time strings.
%m	The month represented as a zero-padded decimal number.	`01, 02, ..., 12`	The leading zero is optional for formatting directives `%d, %m, %H, %I, %M, %S, %J, %U, %W, %V, %y` when they are used to parse date and time strings.
%y	The year without the century part, represented as a zero-padded decimal number.	`00, 01, ..., 99`	The leading zero is optional for formatting directives `%d, %m, %H, %I, %M, %S, %J, %U, %W, %V, %y` when they are used to parse date and time strings.
%Y	The year with the century part, represented as a zero-padded decimal number.	`0001, 0002, ..., 2013, 2014, ..., 9998, 9999`	A year can be parsed from a number that ranges from 1 to 9999. If the year is earlier than 1000, it must be zero-filled to 4-digit width. For example, 0180 indicates the year of 180 AD.

Directive	Description	Example	Note
%H	The hour in the 24-hour format, represented as a zero-padded decimal number.	`00, 01, ..., 23`	The leading zero is optional for formatting directives `%d, %m, %H, %I, %M, %S, %J, %U, %W, %V, %y` when they are used to parse date and time strings.
%I	The hour in the 12-hour format, represented as a zero-padded decimal number.	`01, 02, ..., 12`	The leading zero is optional for formatting directives `%d, %m, %H, %I, %M, %S, %J, %U, %W, %V, %y` when they are used to parse date and time strings.
%p	The period in the 12-hour format.	`AM, PM`	• Currently, date and time strings are displayed in the en-US locale. Other locales are not supported. • The `%p` directive affects the hour field in the parsing result when the `%I` directive is used to parse the hour.
%M	The minute represented as a zero-padded decimal number.	`00, 01, ..., 59`	The leading zero is optional for formatting directives `%d, %m, %H, %I, %M, %S, %J, %U, %W, %V, %y` when they are used to parse date and time strings.
%S	The second represented as a zero-padded decimal number.	`00, 01, ..., 59`	• Leap seconds are not supported. • The leading zero is optional for formatting directives `%d, %m, %H, %I, %M, %S, %J, %U, %W, %V, %y` when they are used to parse date and time strings.
%f	The microsecond represented as a zero-padded decimal number.	`000000, 000001, ..., 999999`	The `%f` directive can be used to parse the microsecond from a numeric string consisting of 0 to 6 characters.

Directive	Description	Example	Note
%z	The UTC offset in the ±HHMM[SS[.ffffff]] format. An empty string is generated for this directive during parsing if the date and time string does not contain the time zone information.	`(empty), +0000, -0400, +1030, +063415, -030712.345216`	The `%z` and `%Z` directives are replaced by empty strings during parsing if the date and time string does not contain the time zone information. The minute information is optional in the string to be parsed by the `%z` directive to the `±HHMM[SS[.ffffff]]` format. Strings separated by colons (:) are supported during parsing. For example, `+01:00:00` is parsed as an offset of one hour. In addition, `Z` is identical to `+00:00`.
%Z	The name of the time zone. An empty string is generated for this directive during parsing if the date and time string does not contain the time zone information.	`(empty), UTC, EST, CST`	None.
%j	The day of the year.	`001, 002, ..., 366`	The leading zero is optional for formatting directives `%d, %m, %H, %I, %M, %S, %J, %U, %W, %V, %y` when they are used to parse date and time strings.
%U	The week number of the year, where Sunday is the first day of a week. A day before the first Sunday of the year is regarded as a day in week 0.	`00, 01, ..., 53`	• When used to parse data and time strings, the values obtained through the `%U` and `%W` directives can only be used in calculation. • The leading zero is optional for formatting directives `%d, %m, %H, %I, %M, %S, %J, %U, %W, %V, %y` when they are used to parse date and time strings.

Directive	Description	Example	Note
%W	The week number of the year, where Monday is the first day of a week. A day before the first Monday of the year is regarded as a day in week 0.	`00, 01, ..., 53`	• When used to parse data and time strings, the values obtained through the `%U` and `%W` directives can only be used in calculation. • The leading zero is optional for formatting directives `%d, %m, %H, %I, %M, %S, %J, %U, %W, %V, %y` when they are used to parse date and time strings.
%c	The date and time representation in the current locale.	`Tue Aug 16 21:30:00 1988`	Currently, date and time strings are displayed in the en-US locale. Other locales are not supported.
%x	The date representation in the current locale.	`08/16/88`	Currently, date and time strings are displayed in the en-US locale. Other locales are not supported.
%X	The time representation in the current locale.	`21:30:00`	Currently, date and time strings are displayed in the en-US locale. Other locales are not supported.
%%	The literal `%` character.	`%`	None.

Several additional directives not defined in the C (1989 version) standard are included for convenience.

Directive	Description	Example	Note
%G	The ISO 8601 week-based year.	`0001, 0002, ..., 2013, 2014, ..., 9998, 9999`	When used to parse data and time strings, the values obtained through the `%V` directive can only be used in calculation.
%u	The ISO 8601 weekday, where Monday is the first day of a week.	`1, 2, ..., 7`	None.

Directive	Description	Example	Note
%V	The ISO 8601 week number, where Monday is the first day of a week.	`01, 02, ..., 53`	• When used to parse data and time strings, the values obtained through the `%V` directive can only be used in calculation. • The leading zero is optional for formatting directives `%d, %m, %H, %I, %M, %S, %J, %U, %W, %V, %y` when they are used to parse date and time strings.

12.7.8. Time zone list

This topic lists the values of the time zone parameter in the date and time functions.

Many date and time functions in the domain specific language (DSL) syntax use the parameter `tz=time zone string`. Examples:

```
dt_parse(v("__time__"), tz="Asia/Shanghai") # Parse the value of the __time__ field to the time in the time zone of Shanghai.
dt_now(tz="Asia/Tokyo") # Obtain the current time in the time zone of Tokyo.
```

The following section lists all the valid values of the `tz` parameter. Each row displays four time zone strings separated with commas (,).

```
Africa/Abidjan, Africa/Accra, Africa/Addis_Ababa, Africa/Algiers
Africa/Asmara, Africa/Asmera, Africa/Bamako, Africa/Bangui
Africa/Banjul, Africa/Bissau, Africa/Blantyre, Africa/Brazzaville
Africa/Bujumbura, Africa/Cairo, Africa/Casablanca, Africa/Ceuta
Africa/Conakry, Africa/Dakar, Africa/Dar_es_Salaam, Africa/Djibouti
Africa/Douala, Africa/El_Aaiun, Africa/Freetown, Africa/Gaborone
Africa/Harare, Africa/Johannesburg, Africa/Juba, Africa/Kampala
Africa/Khartoum, Africa/Kigali, Africa/Kinshasa, Africa/Lagos
Africa/Libreville, Africa/Lome, Africa/Luanda, Africa/Lubumbashi
Africa/Lusaka, Africa/Malabo, Africa/Maputo, Africa/Maseru
Africa/Mbabane, Africa/Mogadishu, Africa/Monrovia, Africa/Nairobi
Africa/Ndjamena, Africa/Niamey, Africa/Nouakchott, Africa/Ouagadougou
Africa/Porto-Novo, Africa/Sao_Tome, Africa/Timbuktu, Africa/Tripoli
Africa/Tunis, Africa/Windhoek, America/Adak, America/Anchorage
America/Anguilla, America/Antigua, America/Araguaina, America/Argentina/Buenos_Aires
America/Argentina/Catamarca, America/Argentina/ComodRivadavia, America/Argentina/Cordoba, America/Argentina/Jujuy
America/Argentina/La_Rioja, America/Argentina/Mendoza, America/Argentina/Rio_Gallegos, America/Argentina/Salta
America/Argentina/San_Juan, America/Argentina/San_Luis, America/Argentina/Tucuman, America/Argentina/Ushuaia
America/Aruba, America/Asuncion, America/Atikokan, America/Atka
```

```
America/Bahia, America/Bahia_Banderas, America/Barbados, America/Belem
America/Belize, America/Blanc-Sablon, America/Boa_Vista, America/Bogota
America/Boise, America/Buenos_Aires, America/Cambridge_Bay, America/Campo_Grande
America/Cancun, America/Caracas, America/Catamarca, America/Cayenne
America/Cayman, America/Chicago, America/Chihuahua, America/Coral_Harbour
America/Cordoba, America/Costa_Rica, America/Creston, America/Cuiaba
America/Curacao, America/Danmarkshavn, America/Dawson, America/Dawson_Creek
America/Denver, America/Detroit, America/Dominica, America/Edmonton
America/Eirunepe, America/El_Salvador, America/Ensenada, America/Fort_Nelson
America/Fort_Wayne, America/Fortaleza, America/Glace_Bay, America/Godthab
America/Goose_Bay, America/Grand_Turk, America/Grenada, America/Guadeloupe
America/Guatemala, America/Guayaquil, America/Guyana, America/Halifax
America/Havana, America/Hermosillo, America/Indiana/Indianapolis, America/Indiana/Knox
America/Indiana/Marengo, America/Indiana/Petersburg, America/Indiana/Tell_City, America/Ind
iana/Vevay
America/Indiana/Vincennes, America/Indiana/Winamac, America/Indianapolis, America/Inuvik
America/Iqaluit, America/Jamaica, America/Jujuy, America/Juneau
America/Kentucky/Louisville, America/Kentucky/Monticello, America/Knox_IN, America/Kralendi
jk
America/La_Paz, America/Lima, America/Los_Angeles, America/Louisville
America/Lower_Princes, America/Maceio, America/Managua, America/Manaus
America/Marigot, America/Martinique, America/Matamoros, America/Mazatlan
America/Mendoza, America/Menominee, America/Merida, America/Metlakatla
America/Mexico_City, America/Miquelon, America/Moncton, America/Monterrey
America/Montevideo, America/Montreal, America/Montserrat, America/Nassau
America/New_York, America/Nipigon, America/Nome, America/Noronha
America/North_Dakota/Beulah, America/North_Dakota/Center, America/North_Dakota/New_Salem, A
merica/Ojinaga
America/Panama, America/Pangnirtung, America/Paramaribo, America/Phoenix
America/Port-au-Prince, America/Port_of_Spain, America/Porto_Acre, America/Porto_Velho
America/Puerto_Rico, America/Punta_Arenas, America/Rainy_River, America/Rankin_Inlet
America/Recife, America/Regina, America/Resolute, America/Rio_Branco
America/Rosario, America/Santa_Isabel, America/Santarem, America/Santiago
America/Santo_Domingo, America/Sao_Paulo, America/Scoresbysund, America/Shiprock
America/Sitka, America/St_Barthelemy, America/St_Johns, America/St_Kitts
America/St_Lucia, America/St_Thomas, America/St_Vincent, America/Swift_Current
America/Tegucigalpa, America/Thule, America/Thunder_Bay, America/Tijuana
America/Toronto, America/Tortola, America/Vancouver, America/Virgin
America/Whitehorse, America/Winnipeg, America/Yakutat, America/Yellowknife
Antarctica/Casey, Antarctica/Davis, Antarctica/DumontDUrville, Antarctica/Macquarie
Antarctica/Mawson, Antarctica/McMurdo, Antarctica/Palmer, Antarctica/Rothera
Antarctica/South_Pole, Antarctica/Syowa, Antarctica/Troll, Antarctica/Vostok
Arctic/Longyearbyen, Asia/Aden, Asia/Almaty, Asia/Amman
Asia/Anadyr, Asia/Aqtau, Asia/Aqtobe, Asia/Ashgabat
Asia/Ashkhabad, Asia/Atyrau, Asia/Baghdad, Asia/Bahrain
Asia/Baku, Asia/Bangkok, Asia/Barnaul, Asia/Beirut
Asia/Bishkek, Asia/Brunei, Asia/Calcutta, Asia/Chita
Asia/Choibalsan, Asia/Chongqing, Asia/Chungking, Asia/Colombo
Asia/Dacca, Asia/Damascus, Asia/Dhaka, Asia/Dili
Asia/Dubai, Asia/Dushanbe, Asia/Famagusta, Asia/Gaza
Asia/Harbin, Asia/Hebron, Asia/Ho_Chi_Minh, Asia/Hong_Kong
Asia/Hovd, Asia/Irkutsk, Asia/Istanbul, Asia/Jakarta
Asia/Jayapura, Asia/Jerusalem, Asia/Kabul, Asia/Kamchatka
Asia/Karachi, Asia/Kashgar, Asia/Kathmandu, Asia/Katmandu
```

```
Asia/Khandyga, Asia/Kolkata, Asia/Krasnoyarsk, Asia/Kuala_Lumpur
Asia/Kuching, Asia/Kuwait, Asia/Macao, Asia/Macau
Asia/Magadan, Asia/Makassar, Asia/Manila, Asia/Muscat
Asia/Nicosia, Asia/Novokuznetsk, Asia/Novosibirsk, Asia/Omsk
Asia/Oral, Asia/Phnom_Penh, Asia/Pontianak, Asia/Pyongyang
Asia/Qatar, Asia/Qostanay, Asia/Qyzylorda, Asia/Rangoon
Asia/Riyadh, Asia/Saigon, Asia/Sakhalin, Asia/Samarkand
Asia/Seoul, Asia/Shanghai, Asia/Singapore, Asia/Srednekolymsk
Asia/Taipei, Asia/Tashkent, Asia/Tbilisi, Asia/Tehran
Asia/Tel_Aviv, Asia/Thimbu, Asia/Thimphu, Asia/Tokyo
Asia/Tomsk, Asia/Ujung_Pandang, Asia/Ulaanbaatar, Asia/Ulan_Bator
Asia/Urumqi, Asia/Ust-Nera, Asia/Vientiane, Asia/Vladivostok
Asia/Yakutsk, Asia/Yangon, Asia/Yekaterinburg, Asia/Yerevan
Atlantic/Azores, Atlantic/Bermuda, Atlantic/Canary, Atlantic/Cape_Verde
Atlantic/Faeroe, Atlantic/Faroe, Atlantic/Jan_Mayen, Atlantic/Madeira
Atlantic/Reykjavik, Atlantic/South_Georgia, Atlantic/St_Helena, Atlantic/Stanley
Australia/ACT, Australia/Adelaide, Australia/Brisbane, Australia/Broken_Hill
Australia/Canberra, Australia/Currie, Australia/Darwin, Australia/Eucla
Australia/Hobart, Australia/LHI, Australia/Lindeman, Australia/Lord_Howe
Australia/Melbourne, Australia/NSW, Australia/North, Australia/Perth
Australia/Queensland, Australia/South, Australia/Sydney, Australia/Tasmania
Australia/Victoria, Australia/West, Australia/Yancowinna, Brazil/Acre
Brazil/DeNoronha, Brazil/East, Brazil/West, CET
CST6CDT, Canada/Atlantic, Canada/Central, Canada/Eastern
Canada/Mountain, Canada/Newfoundland, Canada/Pacific, Canada/Saskatchewan
Canada/Yukon, Chile/Continental, Chile/EasterIsland, Cuba
EET, EST, EST5EDT, Egypt
Eire, Etc/GMT, Etc/GMT+0, Etc/GMT+1
Etc/GMT+10, Etc/GMT+11, Etc/GMT+12, Etc/GMT+2
Etc/GMT+3, Etc/GMT+4, Etc/GMT+5, Etc/GMT+6
Etc/GMT+7, Etc/GMT+8, Etc/GMT+9, Etc/GMT-0
Etc/GMT-1, Etc/GMT-10, Etc/GMT-11, Etc/GMT-12
Etc/GMT-13, Etc/GMT-14, Etc/GMT-2, Etc/GMT-3
Etc/GMT-4, Etc/GMT-5, Etc/GMT-6, Etc/GMT-7
Etc/GMT-8, Etc/GMT-9, Etc/GMT0, Etc/Greenwich
Etc/UCT, Etc/UTC, Etc/Universal, Etc/Zulu
Europe/Amsterdam, Europe/Andorra, Europe/Astrakhan, Europe/Athens
Europe/Belfast, Europe/Belgrade, Europe/Berlin, Europe/Bratislava
Europe/Brussels, Europe/Bucharest, Europe/Budapest, Europe/Busingen
Europe/Chisinau, Europe/Copenhagen, Europe/Dublin, Europe/Gibraltar
Europe/Guernsey, Europe/Helsinki, Europe/Isle_of_Man, Europe/Istanbul
Europe/Jersey, Europe/Kaliningrad, Europe/Kiev, Europe/Kirov
Europe/Lisbon, Europe/Ljubljana, Europe/London, Europe/Luxembourg
Europe/Madrid, Europe/Malta, Europe/Mariehamn, Europe/Minsk
Europe/Monaco, Europe/Moscow, Europe/Nicosia, Europe/Oslo
Europe/Paris, Europe/Podgorica, Europe/Prague, Europe/Riga
Europe/Rome, Europe/Samara, Europe/San_Marino, Europe/Sarajevo
Europe/Saratov, Europe/Simferopol, Europe/Skopje, Europe/Sofia
Europe/Stockholm, Europe/Tallinn, Europe/Tirane, Europe/Tiraspol
Europe/Ulyanovsk, Europe/Uzhgorod, Europe/Vaduz, Europe/Vatican
Europe/Vienna, Europe/Vilnius, Europe/Volgograd, Europe/Warsaw
Europe/Zagreb, Europe/Zaporozhye, Europe/Zurich, GB
GB-Eire, GMT, GMT+0, GMT-0
GMT0, Greenwich, HST, Hongkong
Iceland, Indian/Antananarivo, Indian/Chagos, Indian/Christmas
```

```
Iceland, Indian/Antananarivo, Indian/Chagos, Indian/Christmas
Indian/Cocos, Indian/Comoro, Indian/Kerguelen, Indian/Mahe
Indian/Maldives, Indian/Mauritius, Indian/Mayotte, Indian/Reunion
Iran, Israel, Jamaica, Japan
Kwajalein, Libya, MET, MST
MST7MDT, Mexico/BajaNorte, Mexico/BajaSur, Mexico/General
NZ, NZ-CHAT, Navajo, PRC
PST8PDT, Pacific/Apia, Pacific/Auckland, Pacific/Bougainville
Pacific/Chatham, Pacific/Chuuk, Pacific/Easter, Pacific/Efate
Pacific/Enderbury, Pacific/Fakaofo, Pacific/Fiji, Pacific/Funafuti
Pacific/Galapagos, Pacific/Gambier, Pacific/Guadalcanal, Pacific/Guam
Pacific/Honolulu, Pacific/Johnston, Pacific/Kiritimati, Pacific/Kosrae
Pacific/Kwajalein, Pacific/Majuro, Pacific/Marquesas, Pacific/Midway
Pacific/Nauru, Pacific/Niue, Pacific/Norfolk, Pacific/Noumea
Pacific/Pago_Pago, Pacific/Palau, Pacific/Pitcairn, Pacific/Pohnpei
Pacific/Ponape, Pacific/Port_Moresby, Pacific/Rarotonga, Pacific/Saipan
Pacific/Samoa, Pacific/Tahiti, Pacific/Tarawa, Pacific/Tongatapu
Pacific/Truk, Pacific/Wake, Pacific/Wallis, Pacific/Yap
Poland, Portugal, ROC, ROK
Singapore, Turkey, UCT, US/Alaska
US/Aleutian, US/Arizona, US/Central, US/East-Indiana
US/Eastern, US/Hawaii, US/Indiana-Starke, US/Michigan
US/Mountain, US/Pacific, US/Samoa, UTC
Universal, W-SU, WET, Zulu
```

13.Performance guide

This topic describes the factors that may affect data transformation performance.

The speed of a data transformation task depends on the number of source shards and the logic and complexity of the transformation rule. For more information, see Data transformation basics. In most cases, one shard is required for a transformation speed of 1 MB uncompressed data per second (85 GB/day). For example, if data is written to the source Logstore at a speed of 1 TB per day, the number of shards in the source Logstore must be 12 (1024 GB/85 = 12). For more information, see Split a shard.

Data transformation performance

The data transformation speed is related to the following factors of a transformation rule:

- Output log entries
  - Log size. The larger the output log size, the slower the transformation speed. This is because the output of larger data packets requires more computing and network resources. The smaller the output log size, the faster the transformation speed. The output log size is measured by the number of output log entries (a log entry is split if the size is large), number of fields contained in output log entries, or the content of output log entries.
  - Log groups. Output log entries are tagged and packaged into groups. More log groups require more network resources and lead to a slower transformation speed. The fewer the output log groups, the faster the transformation speed.
- Transformation logic

  The more complex the transformation logic, the slower the transformation speed. This is because more complex transformation logic results in more searches, computations, and external resource synchronizations. These processes consume more computing and network resources. The less complex the transformation logic, the faster the transformation speed.
- Third-party data sources

  If you use a third-party source to enrich your data, a larger size of pulled data indicates a slower transformation speed. In addition, if the pulled data such as OSS objects reside in another region, the transformation speed will also be reduced.

Improve the data transformation performance of the source Logstore

- Improve the performance of real-time data transformation.

  You can increase the number of shards to improve the performance of real-time data transformation. For more information about the billing methods of shards, see Pay-as-you-go.
- Improve the performance of historical data transformation.

  Shard splitting applies only to new data. If the size of historical data is large and the number of shards is insufficient, you can create multiple data transformation tasks for the source Logstore and configure non-overlapping transformation periods for the tasks. For example, if you need to transform historical log data that is generated from September 1 to September 10, you can create nine tasks to transform data that is generated in the following periods: `[September 1, September 2), [September 2, September 3) ... [September 9, September 10]`.

> **Note** The transformation period is calculated based on the log receiving time. For more information, see Create a data transformation job.

Improve the data transformation performance of the destination Logstore

The number of shards in the destination Logstore depends on the following two factors:

- The write speed during data transformation. The maximum write speed of a shard is 5 MB/s. You can estimate the specific write speed based on the number of shards in the source Logstore and the number of concurrent data transformation tasks.

  For example, if the source Logstore has 20 shards, the destination Logstore must have at least 4 shards.

- Whether you need to create indexes to query data in the destination Logstore. If you need to create indexes to query data in the destination Logstore, we recommend that you plan 50 million log entries for a shard and specify the number of shards based on the plan.

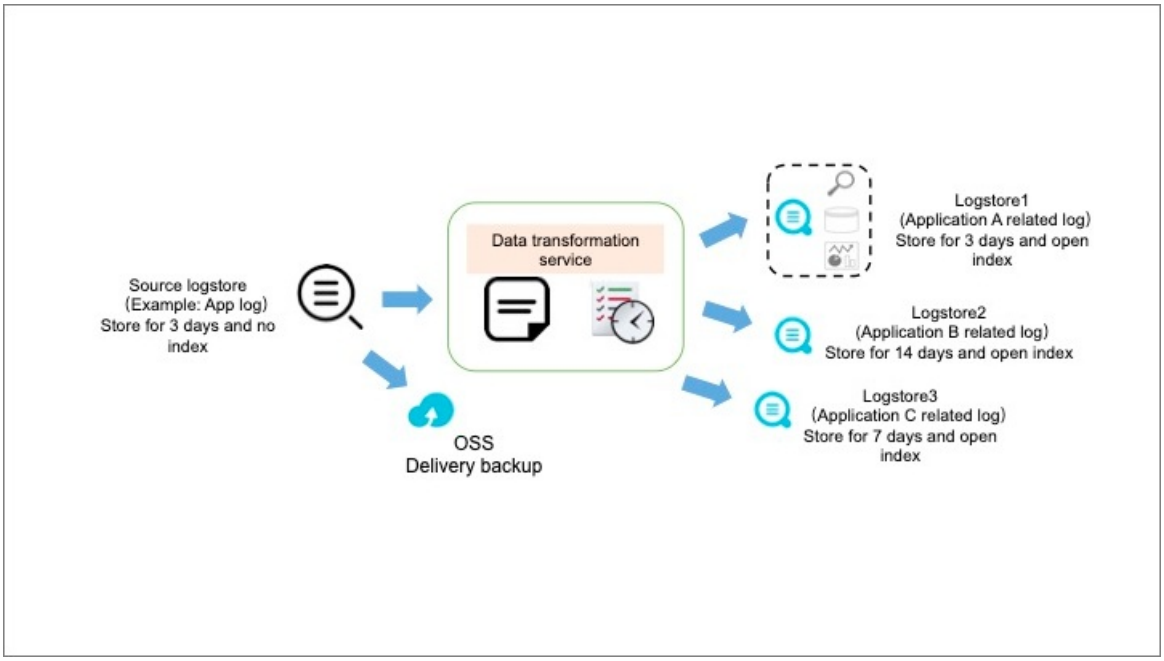
  For example, if each log entry is 1 KB and the size of the log entries that you need to transform and write every day is 10 GB, the number of log entries is 10 million. If you need to query data that covers 30 days (about 300 million log entries), we recommend that you create 6 shards in the destination Logstore.

14.Cost optimization guide

The data transformation feature helps reduce your time and labor costs to tidy data and boost your business. This topic describes how to configure rules to transform data at an optimal cost.

Typical configurations

We recommend that you import log data into one or more Logstores and use the data transformation feature to dispatch transformed data to destination Logstores. In addition, we recommend that you configure retention periods and indexes for data in different destination Logstores. For more information, see Data transformation basics and Performance guide.



Cost factors

When you use Log Service, your costs depend on the following factors. For more information, see Billing methods.

- The amount of data imported per day
- The data retention period
- The number of indexes that you create

The following examples describe how to optimize costs.

Optimize imported logs

Assume that you collect logs from an application and import 100 GB of log data into a source Logstore per day. You also create full-text indexes for the log data and set a data retention period of 30 days. In this case, you are billed about USD 562 per month.

However, you want to filter the logs from pods of a certain type, such as operations logs and error logs. These logs account for 20% of the total amount of the raw logs. You also want to retain these logs for 30 days and retain other logs for seven days. In this case, we recommend that you use the following method to optimize costs:

- Create a source Logstore that retains logs for three days. Do not create indexes for the log data.

- Create a destination Logstore to store operations logs and error logs for 30 days and create indexes for the log data.
- Create another destination Logstore to store other logs for seven days and create indexes for the log data.

In this case, you are billed about USD 421 per month. This method can reduce your costs by 25%.

You can use the data transformation feature to retain important logs for 60 days and the other logs for seven days. If you want to retain 20% of your logs, your costs are reduced by 12% and the retention period of those logs is doubled.

Optimize log entries in a log

Assume that you collect logs from an application and import 100 GB of log data into a source Logstore per day. You also create full-text indexes for the log data and set a data retention period of 30 days. In this case, you are billed about USD 562 per month.

The following is a raw log entry with a size of 1021 bytes.

```
__source__:  192.0.2.0
__topic__:  ddos_access_log
body_bytes_sent:  3866
cc_action:  none
cc_blocks:
cc_phase:
content_type:  text/x-flv
host:  www.example.com
http_cookie:  i1=w1;x2=q2
http_referer:  http://www.example.com
http_user_agent:  Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko
) Chrome/192.0.2.1 Safari/537.36
http_x_forwarded_for:  192.0.2.2
https:  true
isp_line:  BGP
matched_host:  www.example.com
method:  GET
real_client_ip:  192.0.2.3
remote_addr:  192.0.2.4
remote_port:  48196
request_length:  2946
request_method:  GET
request_time_msec:  78920
request_uri:  /request/nvwlvvkhw
server_name:  www.example.com
status:  502
time:  2019-07-22T17:40:26+08:00
ua_browser:  mozilla
ua_browser_family:
ua_browser_type:
ua_browser_version:  9.0
ua_device_type:
ua_os:  windows_7
ua_os_family:
upstream_addr:  192.0.2.4:80
upstream_ip:  192.0.2.5
upstream_response_time:  0.858
upstream_status:  200
user_id:  st0s2b5
```

If you require only some fields in the log entry, you can retain the destination fields for 30 days and create indexes for these fields. You can retain other fields for only three days. In this case, we recommend that you use the following method:

- Create a source Logstore that retains logs for three days. Do not create indexes for the log data.
- Create a destination Logstore to store operations logs and error logs for 30 days and create indexes for the log data.

If the size of a transformed log entry is 60% the size of the raw log entry, you are billed about USD 393 per month. This method can reduce your costs by 30%.

The following is the 618-byte log entry transformed from the 1021-byte raw log entry.

```
__source__:  192.0.2.0
__topic__:  ddos_access_log
body_bytes_sent:  3866
content_type:  text/x-flv
host:  www.example.com
http_referer:  http://www.example.com
ua_browser:  mozilla
ua_browser_family:
ua_browser_type:
ua_browser_version:  9.0
ua_device_type:
ua_os:  windows_7
http_x_forwarded_for:  192.0.2.2
matched_host:  www.example.com
method:  GET
real_client_ip:  192.0.2.3
request_length:  2946
request_uri:  /request/nvwlvvkhw
status:  502
upstream_addr:  192.0.2.4:80
upstream_ip:  192.0.2.5
upstream_response_time:  0.858
upstream_status:  200
user_id:  st0s2b5
```

15.Best practices

15.1. Cleanse data by using functions

The data transformation feature of Log Service allows you to cleanse raw data. You can use one or more functions to cleanse a large amount of data. This way, the logs collected to Log Service can be converted to a standard format. This topic describes how to use functions to cleanse data in various scenarios.

Scenario 1: Filter logs by using the e_keep function and e_drop function

You can use the e_drop or e_keep function to filter logs. You can also specify the DROP parameter and use the e_if or e_if_else function to filter logs.

The following common transformation rules can be used:

- `e_keep(e_search(...))` : The log entries that meet the conditions are retained. Otherwise, the log entries are dropped.
- `e_drop(e_search(...))` : The log entries that meet the conditions are dropped. Otherwise, the log entries are retained.
- `e_if_else(e_search("..."), KEEP, DROP)` : The log entries that meet the conditions are retained. Otherwise, the log entries are dropped.
- `e_if(e_search("not ..."), DROP)` : The log entries that meet the conditions are dropped. Otherwise, the log entries are retained.
- `e_if(e_search("..."), KEEP)` . This transformation rule is invalid.

Example:

- Raw log entries

```
# Log entry 1
__source__:  192.168.0.1
__tag__:__client_ip__:  192.168.0.2
__tag__:__receive_time__:  1597214851
__topic__: app
class:  test_case
id:  7992
test_string:  <function test1 at 0x1027401e0>
# Log entry 2
__source__:  192.168.0.1
class:  produce_case
id:  7990
test_string:  <function test1 at 0x1020401e0>
```

- Transformation rule

  Drop the log entries whose __topic__ and __tag__:__receive_time__ fields are empty.

```
e_if(e_not_has("__topic__"),e_drop())
e_if(e_not_has("__tag__:__receive_time__"),e_drop())
```

- Result

```
__source__:  192.168.0.1
__tag__:__client_ip__:  192.168.0.2
__tag__:__receive_time__:  1597214851
__topic__: app
class:  test_case
id:  7992
test_string:  <function test1 at 0x1027401e0>
```

Scenario 2: Assign values to empty fields in a log entry by using the e_set function

You can use the e_set function to assign values to empty fields in a log entry.

- Sub-scenario 1: Assign a value to a field if the field does not exist or is empty.

  ```
  e_set("result", "......value......", mode="fill")
  ```

  For information about the mode parameter, see Field check and overwrite modes.

  Example:

  - Raw log entry

    ```
    name:
    ```

  - Transformation rule

    ```
    e_set("name", "aspara2.0", mode="fill")
    ```

  - Result

    ```
    name:  aspara2.0
    ```

- Sub-scenario 2: Use Grok function to simplify a regular expression and extract field values.

  Example:

  - Raw log entry

    ```
    content: "ip address: 192.168.1.1"
    ```

  - Transformation rule

    Use the Grok function to extract the IP address in the content field.

    ```
    e_regex("content", grok(r"(%{IP})"),"addr")
    ```

  - Result

    ```
    addr:  192.168.1.1
    content: "ip address: 192.168.1.1"
    ```

- Sub-scenario 3: Assign values to multiple fields.

  ```
  e_set("k1", "v1", "k2", "v2", "k3", "v3", ......)
  ```

  Example:

- Raw log entry

```
__source__:  192.168.0.1
__topic__:
__tag__:
__receive_time__:
id:  7990
test_string:  <function test1 at 0x1020401e0>
```

- Transformation rule

  Assign values to the __topic__, __tag__, and __receive_time__ fields.

```
e_set("__topic__","app", "__tag__","stu","__receive_time__","1597214851")
```

- Result

```
__source__:  192.168.0.1
__topic__:  app
__tag__:  stu
__receive_time__:  1597214851
id:  7990
test_string:  <function test1 at 0x1020401e0>
```

Scenario 3: Delete a field and rename a field by using the e_search, e_rename, and e_compose functions

We recommend that you use the e_compose function to check whether the data meets the conditions and then perform operations based on the check result.

Example:

- Raw log entry

```
content: 123
age: 23
name: twiss
```

- Transformation rule

  If the value of the content field is *123*, delete the age and name fields. Then, rename the content field to ctx.

```
e_if(e_search("content==123"),e_compose(e_drop_fields("age|name"), e_rename("content", "ctx")))
```

- Result

```
ctx: 123
```

Scenario 4: Convert the type of fields in a log entry by using the v, cn_int, and dt_totimestamp functions

The fields and values in log entries are processed as strings during the data transformation process. Data of a non-string type is automatically converted to data of the string type. Therefore, you must be familiar with the types of fields that a function can receive when you invoke the function. For more information, see Syntax introduction.

- Sub-scenario 1: Use the op_add function to concatenate strings or add numbers.

  The op_add function can receive data of both the string and numeric types. Therefore, no field type needs to be converted.

  Example:

  - Raw log entry

    ```
    a : 1
    b : 2
    ```

  - Transformation rule

    ```
    e_set("d",op_add(v("a"), v("b")))
    e_set("e",op_add(ct_int(v("a")), ct_int(v("b"))))
    ```

  - Result

    ```
    a:1
    b:2
    d:12
    e:3
    ```

- Sub-scenario 2: Use the Event check functions function and ct_int function to convert data types and use the op_mul function to multiply data.

  Example:

  - Raw log entry

    ```
    a:2
    b:5
    ```

  - Transformation rule

    The values of both v("a") and v("b") are of the string type. However, the second field of the op_mul function can receive only numeric values. Therefore, you must use the ct_int function to convert a string to an integer, and then pass the value to the op_mul function.

    ```
    e_set("c",op_mul(ct_int(v("a")), ct_int(v("b"))))
    e_set("d",op_mul(v("a"), ct_int(v("b"))))
    ```

  - Result

    ```
    a: 2
    b: 5
    c: 10
    d: 22222
    ```

- Sub-scenario 3: Use the dt_parse function or dt_parsetimestamp function to convert a string or datetime object to standard time.

  The dt_totimestamp function receives only datetime objects. Therefore, you must use the dt_parse function to convert the string value of time1 to a datetime object. You can also use the dt_parsetimestamp function because it can receive both datetime objects and strings. For more information, see Date and time functions.

  Example:

- Raw log entry

```
time1: 2020-09-17 9:00:00
```

- Transformation rule

  Convert the time indicated by time1 to a UNIX timestamp.

```
e_set("time1", "2019-06-03 2:41:26")
e_set("time2", dt_totimestamp(dt_parse(v("time1")))) or e_set("time2", dt_parsetimestam
p(v("time1")))
```

- Result:

```
time1:  2019-06-03 2:41:26
time2:  1559529686
```

Scenario 5: Fill the default values in log fields that do not exist by specifying the default parameter

Some expression functions that are used to transform data in Log Service have specific requirements for input parameters. If the input parameters do not meet the requirements, the data transformation rule returns the default values or an error. If a necessary log field is incomplete, you can fill the default value in the log field by using the op_len function.

> **Notice** If default values are passed to subsequent functions, errors may occur. We recommend that you resolve the exceptions returned by the data transformation rules at the earliest opportunity.

- Raw log entry

```
data_len: 1024
```

- Transformation rule

```
e_set("data_len", op_len(v("data", default="")))
```

- Result

```
data: 0
data_len: 0
```

Scenario 6: Add one or more fields based on conditions by using the e_if and e_switch functions

We recommend that you use the e_if or e_switch function to add one or more fields to log entries based on specified conditions. For more information, see Flow control functions.

- e_if function syntax

```
e_if(condition 1, operation 1, condition 2, operation 2, condition 3, operation 3, ...)
```

- e_switch function syntax

When you use the e_switch function, you must specify condition-operation pairs. The conditions are checked in sequence. An operation is performed only after its paired condition is met. After a condition is met, the corresponding operation result is returned and no more conditions are checked. If a condition is not met, its paired operation is not performed and the next condition is checked. If no conditions are met and the default field is specified, the operation that is specified by default is performed and the corresponding result is returned.

```
e_switch(condition 1, operation 1, condition 2, operation 2, condition 3, operation 3, ..
., default=None)
```

Example:

- Raw log entry

```
status1: 200
status2: 404
```

- e_if function
  - Transformation rule

```
e_if(e_match("status1", "200"), e_set("status1_info", "normal"),
     e_match("status2", "404"), e_set("status2_info", "error"))
```

  - Result

```
status1: 200
status2: 404
status1_info: normal
status2_info: error
```

- e_switch function
  - Transformation rule

```
e_switch(e_match("status1", "200"), e_set("status1_info", "normal"),
         e_match("status2", "404"), e_set("status2_info", "error"))
```

  - Result

    The e_switch function checks the conditions in sequence. After a condition is met, the corresponding operation result is returned and no more conditions are checked.

```
status1: 200
status2: 404
status1_info: normal
```

15.2. Check data by using functions

You can check and process data based on specific conditions by using the functions. This topic describes how to use functions to check data in various scenarios.

Scenario 1: Check whether a field exists

- Raw log entry

```
a: a_value
b:       // Empty string
```

- Domain-specific language (DSL) orchestration
  - Solution 1: Use the `e_has` and `e_not_has` functions.

```
e_if(e_has("a"),e_set("has_a", true))
e_if(e_has("b"),e_set("has_b", true))
e_if(e_has("c"),e_set("has_c", true))
e_if(e_not_has("a"),e_set("not_has_a", true))
e_if(e_not_has("b"),e_set("not_has_b", true))
e_if(e_not_has("c"),e_set("not_has_c", true))
```

  - Solution 2: Use the `e_search` function.

```
e_if(e_search('a: *'),e_set("has_a", true))
e_if(e_search('b: *'), e_set("has_b", true))
e_if(e_search('c: *'), e_set("has_c", true))
e_if(e_search('not a: *'), e_set("not_has_a", true))
e_if(e_search('not b: *'), e_set("not_has_b", true))
e_if(e_search('not c: *'), e_set("not_has_c", true))
```

> **Note** In the preceding example, an `e_if` function is written for each condition to better illustrate the solution. You can simplify the function by including all conditions and the corresponding operations as `e_if(condition 1, operation 1, condition 2, operation 2)`.

- Result

```
a:a_value
b:    // Empty string
has_a: true
has_b: true
has_c: false
not_has_a: false
not_has_b: false
not_has_c: true
```

Scenario 2: Check whether a field value exists and is not empty

- Raw log entry

```
a: a_value
b:     // Empty string
```

- DSL orchestration

- Solution 1 (recommended): Use the `v` function that returns a field value.

```
e_if(v("a"), e_set("not_empty_a", true))
e_if(v("b"), e_set("not_empty_b", true))
e_if(v("c"), e_set("not_empty_c", true))
```

> **Note** If the field value extracted by the `v` function exists and is not empty, the `Bool` value *true* is returned. Otherwise, *false* is returned.

- Solution 2: Use the `e_search` function.

```
# The field value contains at least one character.
e_if(e_search('a: "?"'), e_set("not_empty_a", true))
e_if(e_search('b: "?"'), e_set("not_empty_b", true))
e_if(e_search('c: "?"'), e_set("not_empty_c", true))
# Regular expression
e_if(e_search('a~=".+"'), e_set("not_empty_a", true))
e_if(e_search('b~=".+"'), e_set("not_empty_b", true))
e_if(e_search('c~=".+"'), e_set("not_empty_c", true))
# The field value exists and is not empty.
e_if(e_search('a: * and not a==""'), e_set("not_empty_a", true))
e_if(e_search('b: * and not b==""'), e_set("not_empty_b", true))
e_if(e_search('c: * and not c==""'), e_set("not_empty_b", true))
```

- Result

```
a: a_value
b:     // Empty string
not_empty_a: true
not_empty_b: false
not_empty_c: false
```

Scenario 3: Check whether a field value exists and is empty

- Raw log entry

```
a: a_value
b:     // Empty string
```

- DSL orchestration

- Solution 1 (recommended): Use the `v` function that returns a field value.

```
e_if(op_and(e_has("a"), op_not(v("a"))), e_set("empty_a", true))
e_if(op_and(e_has("b"), op_not(v("b"))), e_set("empty_b", true))
e_if(op_and(e_has("c"), op_not(v("c"))), e_set("empty_c", true))
# Invalid syntax
e_if(op_not(v("a")), e_set("empty_a", true))
e_if(op_not(v("b")), e_set("empty_b", true))
e_if(op_not(v("c")), e_set("empty_c", true))
```

> **Note** If the field value extracted by the `v` function exists and is not empty, the `Bool` value *true* is returned. Otherwise, *false* is returned. The *true* value is returned if the field value does not exist or if the field value is `None`.

- Solution 2: Use the `e_search` function.

```
e_if(e_search('a=""'), e_set("empty_a", true))
e_if(e_search('b=""'), e_set("empty_b", true))
e_if(e_search('c=""'), e_set("empty_c", true))
# Invalid syntax
e_if(e_search('a:""'), e_set("empty_a", true))
e_if(e_search('b:""'), e_set("empty_b", true))
```

> **Note** In the preceding example of the invalid syntax, the `e_search` function is used for partial query. In this case, true is returned if the value of the `a: ""` field exists, regardless of whether the value is empty.

- Result

```
a: a_value
b:       // Empty string
empty_a: false
empty_b: true
empty_b: false
```

Scenario 4: Perform actions based on the logical relationships between field values

- Raw log entries

```
Log entry 1
http_host: example
status: 200
request_method: GET
scheme: https
header_length: 700
body_length: 1200
Log entry 2
http_host: example.org
status: 200
request_method: POST
scheme: https
header_length: 100
body_length: 800
Log entry 3
http_host: example.net
status: 200
request_method: GET
scheme:  http
header_length: 700
body_length: 800
Log entry 4
http_host: aliyundoc.com
status: 404
request_method: GET
scheme: https
header_length: 100
body_length: 300
```

- Requirement 1

  Add the `type` field to all log entries in which the value of the `status` field is *200*. The value of the type field is *normal*.

  - DSL orchestration

    ```
    e_if(e_match("status", "200"), e_set("type", "normal"))
    Or
    e_if(e_search('status==200'), e_set("type", "normal"))
    ```

    Note

    - You can use one of these solutions in scenarios where the requirements are simple.
    - In this case, `status:200` can be used to check whether the value of the status field contains 200. To be more precise, we recommend that you use `status==200`.

- Result

```
Log entry 1
type: normal
http_host: example
status: 200
request_method: GET
scheme: https
header_length: 700
body_length: 1200
Log entry 2
type: normal
http_host: example.org
status: 200
request_method: POST
scheme: https
header_length: 100
body_length: 800
Log entry 3
type: normal
http_host: example.net
status: 200
request_method: GET
scheme: http
header_length: 700
body_length: 800
Log entry 4
http_host: aliyundoc.com
status: 404
request_method: GET
scheme: https
header_length: 100
body_length: 300
```

- Requirement 2

  Add the `type` field to all log entries that meet the following conditions: the value of the `status` field is *200*, the value of the `request_method` field is *GET*, and the value of the `scheme` field is *https*. The value of the type field is *normal*.

- DSL orchestration

```
e_if(e_search('status==200 and request_method==GET and scheme==https'), e_set("type", "
normal"))
Or
e_if(e_match_all("status", "200", "request_method", "GET", "scheme", "https"), e_set("t
ype", "normal"))
```

> **Note**
>
> - You can use the `e_search` or `e_match_all` function to match multiple fields. The `e_search` function is simpler.
> - In this case, `status:200` can be used to check whether the value of the status field contains 200. To be more precise, we recommend that you use `status==200`.

- Result

```
Log entry 1
type: normal
http_host: example
status: 200
request_method: GET
scheme: https
header_length: 700
body_length: 1200
Log entry 2
http_host: example.org
status: 200
request_method: POST
scheme: https
header_length: 100
body_length: 800
Log entry 3
http_host: example.net
status: 200
request_method: GET
scheme: http
header_length: 700
body_length: 800
Log entry 4
http_host: aliyundoc.com
status: 404
request_method: GET
scheme: https
header_length: 100
body_length: 300
```

- Requirement 3

  Add the `type` field to all log entries that meet one or more of the following conditions: the value of the `status` field is *200*, the value of the `request_method` field is *GET*, or the value of the `scheme` field is *https*. The value of the type field is *normal*.

- DSL orchestration

```
e_if(e_search('status==200 or request_method==GET or scheme==https'), e_set("type", "no
rmal"))
Or
e_if(e_match_any("status", "200", "request_method", "GET", "scheme", "https"), e_set("t
ype", "normal"))
```

- Result

```
Log entry 1
type: normal
http_host: example
status: 200
request_method: GET
scheme: https
header_length: 700
body_length: 100
Log entry 2
type: normal
http_host: example.org
status: 200
request_method: POST
scheme: https
header_length: 100
body_length: 800
Log entry 3
type: normal
http_host: example.net
status: 200
request_method: GET
scheme: http
header_length: 700
body_length: 800
Log entry 4
type: normal
http_host: aliyundoc.com
status: 404
request_method: GET
scheme: https
header_length: 100
body_length: 1300
```

- Requirement 4

  Add the `type` field to all log entries that meet the following conditions: the value of the `status` field is *200*, the value of the `request_method` field is *GET*, and the sum of the values of the `header_length` and `body_length` fields is less than or equal to *1000*. The value of the type field is *normal*.

- DSL orchestration

```
e_if(op_and(e_search('status: 200 and request_method: GET'), op_le(op_sum(v("header_length"), v("body_length")), 1000)), e_set("type", "normal"))
```

> **Note** You can use the `e_search` function and other expression functions for multiple logical operations.

- Result

```
Log entry 1
type: normal
http_host: example
status: 200
request_method: GET
scheme: https
header_length: 700
body_length: 100
Log entry 2
http_host: example.org
status: 200
request_method: POST
scheme: https
header_length: 100
body_length: 800
Log entry 3
http_host: example.net
status: 200
request_method: GET
scheme: http
header_length: 700
body_length: 800
Log entry 4
http_host: aliyundoc.com
status: 404
request_method: GET
scheme: https
header_length: 100
body_length: 1300
```

15.3. Convert datetime

You can convert time and date to improve the efficiency of log query and analysis. This topic describes how to convert and offset the datetime by using functions.

Terms

Domain-specific language (DSL) supports three data types: datetime string, datetime object, and UNIX timestamp.

- Datetime string

Datetime strings are used to convert time and date data into readable strings. Datetime strings are divided into two types in DSL syntax:

- Datetime strings with a time zone, for example, `2019-06-02 18:41:26+08:00`.
- Datetime strings without a time zone, for example, `2019-06-02 10:41:26`.

In a datetime string with a time zone, the time difference is appended to the datetime to indicate the time zone. Examples:

- `2019-06-02 18:41:26+08:00` indicates that the datetime `2019-06-02 18:41:26` is in the `UTC+8` time zone.
- `2019-06-02 18:41:26-07:00` indicates that the datetime `2019-06-02 18:41:26` is in the `UTC-7` time zone.

- Datetime object

  Datetime objects are instantiated to show data and time. Datetime objects are used to convert time and date data into readable strings.

- UNIX timestamp

  A UNIX timestamp indicates the number of seconds that have elapsed since 00:00:00 Thursday, 1 January 1970. UNIX timestamps can be used in the following scenarios:

  - Show system time.

    In event logs, the metadata field `__time__` indicates the time when logs are generated and the field `__receive_time__` indicates the time when Log Service receives logs. The values of these fields use UNIX timestamps to indicate the system time.

    ```
    __source__:  192.0.2.1
    __tag__:__receive_time__:  1562741899
    __topic__:
    __time__: 1562731122
    ```

  - Perform time-related calculations.

    A UNIX timestamp indicates the number of seconds that have elapsed since 00:00:00 Thursday, 1 January 1970. You can use UNIX timestamps to perform time-related calculations in multiple scenarios. For example:

    - Raw log entries

      ```
      time1: 1562741899
      time2: 1562731122
      ```

    - DSL orchestration

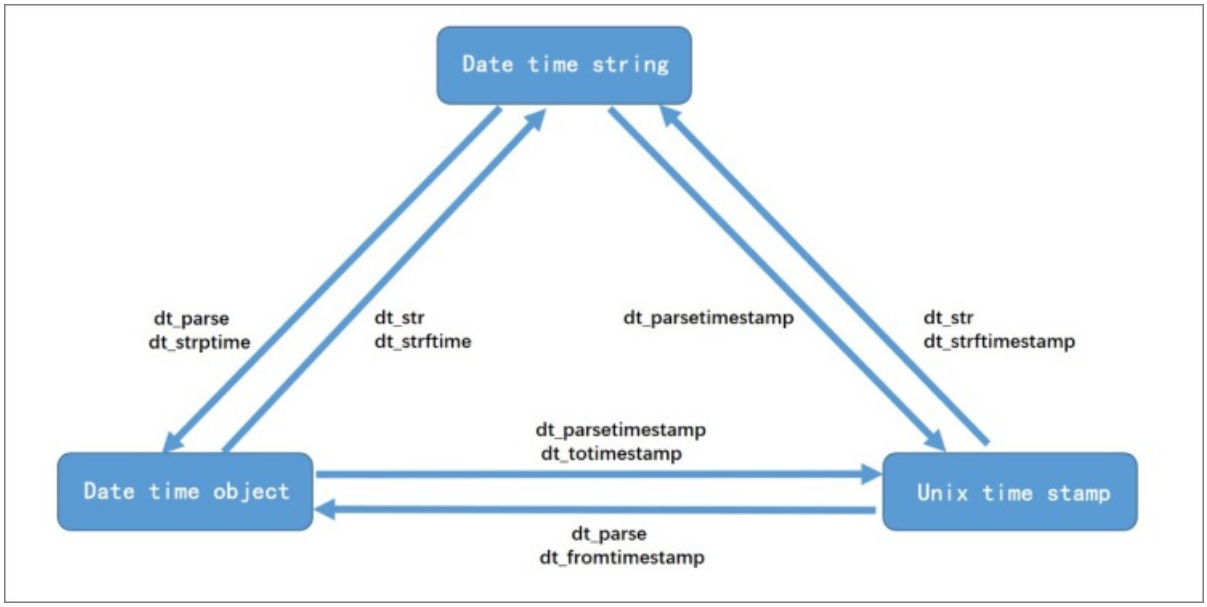
      ```
      e_set("time_diff", op_sub(v("time1"), v("time2")))
      ```

    - Result

      ```
      time1: 1562741899
      time2: 1562731122
      time_diff: 10777
      ```

Use functions to convert data types

The following figure shows how to use functions to convert the three data types: datetime string, datetime object, and UNIX timestamp.



The following table describes the conversion scenarios and conversion functions.

Conversion scenario		Conversion function
Perform conversions between a datetime object and a UNIX timestamp.	Convert a datetime object to a UNIX timestamp.	• `dt_parsetimestamp`: converts a datetime object or a datetime string to a UNIX timestamp. • `dt_totimestamp`: converts a datetime object to a UNIX timestamp.
	Convert a UNIX timestamp to a datetime object.	• `dt_parse`: converts a UNIX timestamp or a datetime string to a datetime object. • `dt_fromtimestamp`: converts a UNIX timestamp to a datetime object.
Perform conversions between a datetime object and a datetime string.	Convert a datetime object to a datetime string.	• `dt_str`: converts a datetime object, a UNIX timestamp, or a datetime string to a datetime string in a specified format. • `dt_strftime`: converts a datetime object to a datetime string.

Conversion scenario		Conversion function
Perform conversions between a datetime string and a UNIX timestamp.	Convert a datetime string to a datetime object.	• `dt_parse`: converts a datetime string or a UNIX timestamp to a datetime object. • `dt_strptime`: converts a datetime string to a datetime object.
	Convert a datetime string to a UNIX timestamp.	`dt_parsetimestamp`: converts a datetime string or a datetime object to a UNIX timestamp.
	Convert a UNIX timestamp to a datetime string.	• `dt_str`: converts a UNIX timestamp, a datetime object, or a datetime string to a datetime string in a specified format. • `dt_strftimestamp`: converts a UNIX timestamp to a datetime string.

Three conversion scenarios and six conversion functions are described in the preceding table. These conversion functions are divided into the following two types:

- Automatic conversion functions

  Automatic conversion functions such as `dt_parse` automatically convert different data types, such as UNIX timestamps, datetime objects, and datetime strings.

- Dedicated functions

  Automatic conversion functions cannot meet your requirements in some scenarios. For example, automatic conversion functions such as `dt_parse` cannot parse date types in some custom formats. In this case, you must use the `dt_strptime` function.

> **Note** For more information, see Date and time functions and Conversion functions.

Perform conversions between a datetime object and a UNIX timestamp

- Conversion functions
  - `dt_parsetimestamp`: Recommended. This automatic conversion function converts a datetime object or a datetime string to a UNIX timestamp.
  - `e_set`: You can set the tz parameter in this function to add a time zone to a datetime object. You can also set the tz parameter to convert a source time zone to a destination time zone.
- Convert a UNIX timestamp to a datetime string with a time zone.

- Raw log entries

```
time: 1562741899
```

- DSL orchestration

```
e_set("new_time", dt_parse(v("time"), tz="Asia/Shanghai"))
```

- Result

```
time: 1562741899
new_time: 2019-07-10 06:58:19+08:00
```

Perform conversions between a datetime string and a UNIX timestamp

- Conversion functions
  - `dt_str`: converts a UNIX timestamp, a datetime object, or a datetime string to a datetime string in a specified format.
  - `dt_strftimestamp`: converts a UNIX timestamp to a datetime string.
  - `dt_parsetimestamp`: converts a datetime string or a datetime object to a UNIX timestamp.
- Scenario 1: Convert a datetime string without a time zone to a UNIX timestamp.

  For example, to convert `2019-06-02 18:41:26` to a UNIX timestamp, you must specify a time zone for the datetime string. The converted UNIX timestamp varies with time zones.
  - Raw log entries

```
time: 2019-06-02 18:41:26
```

  - DSL orchestration

```
e_set("Shanghai_timestamp", dt_parsetimestamp(v("time"), tz="Asia/Shanghai"))
e_set("Los_Angeles_timestamp", dt_parsetimestamp(v("time"), tz="America/Los_Angeles"))
e_set("UTC_timestamp", dt_parsetimestamp(v("time")))
```

  - Result

```
Shanghai_timestamp: 1559472086
Los_Angeles_timestamp: 1559526086
UTC_timestamp: 1559500886
```

  > **Note**
  > - `tz="Asia/Shanghai"` indicates that the time indicated by the `time` field is in the time zone of Shanghai.
  > - If no time zone is specified, the UTC+0 time zone is used as the default time zone.
  > - For more information about the values of the `tz=time zone string` parameter, see Time zone list.
- Scenario 2: Convert a datetime string with a time zone to a UNIX timestamp.

You do not need to specify the time zone parameter if a datetime string contains a time zone, for example `2019-06-02 18:41:26+08:00`.

- Raw log entries

```
China_time : 2019-06-02 18:41:26+08:00
America_time: 2019-06-02 3:41:26-07:00
UTC_time : 2019-06-02 10:41:26+00:00
```

- DSL orchestration

```
e_set("timestamp1", dt_parsetimestamp(v("China_time")))
e_set("timestamp2", dt_parsetimestamp(v("America_time")))
e_set("timestamp3", dt_parsetimestamp(v("UTC_time")))
```

- Result

```
America_time:  2019-06-02 3:41:26-07:00
China_time:  2019-06-02 18:41:26+08:00
UTC_time:  2019-06-02 10:41:26+00:00
timestamp1: 1559472086
timestamp2: 1559472086
timestamp3: 1559472086
```

- Scenario 3: Convert a custom datetime without a time zone to a UNIX timestamp.
  - Raw log entries

```
time1: 2019-07-10 06:58:19
time2: 2019/07/10 06-58-19
```

  - DSL orchestration

```
e_set("time3", dt_parsetimestamp(v("time1")))
e_set("time4", dt_parsetimestamp(dt_strptime(v("time2"), "%Y/%m/%d %H-%M-%S")))
```

  - Result

```
time1: 2019-07-10 06:58:19
time2: 2019/07/10 06-58-19
time3: 1562741899
time4: 1562741899
```

Perform conversions between a datetime object and a datetime string.

- Conversion functions
  - `dt_parse`: converts a datetime string or a UNIX timestamp to a datetime object.
  - `dt_astimezone`: returns a datetime object that contains a time zone.
- Scenario 1: Convert a datetime string without a time zone to a datetime object in a specified time zone.

For example, you can convert the `2019-06-02 18:41:26` datetime string to a UNIX timestamp and then convert the UNIX timestamp to a datetime string in another time zone. The following example shows how to convert the datetime in the time zone of Los Angeles to the datetime in the time zone of Shanghai.

- Raw log entries

```
# Assume that the datetime in the time field is the time zone of Los Angeles.
time : 2019-06-04 2:41:26
```

- DSL orchestration

```
e_set("timestamp", dt_parsetimestamp(v("time"), tz="America/Los_Angeles"))
e_set("Shanghai_time", dt_parse(v("timestamp"), tz="Asia/Shanghai"))
```

- Result

```
Shanghai_time : 2019-06-04 17:41:26+08:00
time : 2019-06-04 2:41:26
timestamp:  1559641286
```

- Scenario 2: Convert a datetime string without a time zone to a datetime object with a time zone.
  - Raw log entries

```
time : 2019-07-10 06:58:19
```

  - DSL orchestration

```
e_set("new_time", dt_parse(v("time"), tz="Asia/Shanghai"))
```

  - Result

```
time: 2019-07-10 06:58:19
new_time: 2019-07-10 06:58:19+08:00
```

- Scenario 3: Convert a datetime string with a time zone to a datetime object in the destination time zone.
  - Raw log entries

```
time : 2019-06-04 2:41:26+08:00
```

  - DSL orchestration

```
e_set("new_time", dt_astimezone(v("time"), tz="America/Los_Angeles"))
```

  - Result

```
new_time : 2019-06-03 11:41:26-07:00
time : 2019-06-04 2:41:26+08:00
```

Offset datetimes

- Conversion functions

- `dt_add` function: The following syntax shows the parameters of this function.

  ```
  dt_add (field name, dt1 = None, dt2 = None, year (s) = None, month (s) = None, day (s) = None, hour (s) = None, minute (s) = None, second (s) = None, microsecond (s) = None, weeks (s) = None, weekday = None)
  ```

  The parameters that end with (s), such as `year (s)`, `month (s)`, and `day (s)`, have two patterns. For example, year (s) can be `year` and `years`, and month (s) can be `month` and `months`. Take `year` and `years` as examples. If `year` is used in the syntax, the value of `year` replaces the value of year in raw log entries. If `years` is used in the syntax, the value of `years` is added to the value of year in raw log entries. You must use the `dt_add` function at the same time. This function allows you to add a value, subtract a value, or overwrite the value of a datetime.
- The `weekday` parameter in the `dt_add` function is used together with the `dt_MO` and `dt_TU` parameters to offset a specified weekday. For more information, see dt_MO.

- Scenario 1: Offset a datetime by year and month.

  The following example shows how to offset a datetime by year and month.

  - Raw log entries

    ```
    time1 : 2019-06-04 2:41:26
    ```

  - DSL orchestration 1

    ```
    e_set("time2", dt_add(v("time1"), year=2018))
    ```

  - Result 1

    ```
    time1 : 2019-06-04 2:41:26
    time2 : 2018-06-04 02:41:26
    ```

  - DSL orchestration 2

    ```
    e_set("time2", dt_add(v("time1"), years=2018))
    ```

  - Result 2

    ```
    time1 : 2019-06-04 2:41:26
    time2 : 4037-06-04 02:41:26
    ```

- Scenario 2: Offset a datetime by week

  The following example shows how to offset a datetime by week.

  - Raw log entries

    ```
    # June 4, 2019 is a Tuesday.
     time1 : 2019-06-04 2:41:26
    ```

- DSL orchestration

```
# Return the datetime of the next Monday after time1.
e_set("nex_Monday", dt_add(v("time1"), weekday=dt_MO(1)))
# Return the datetime of the last Tuesday before time1.
e_set("previous_Tuesday", dt_add(v("time1"), weekday=dt_TU(op_neg(1))))
# Return the datetime of the second Saturday after time1.
e_set("nex_next_Saturday", dt_add(v("time1"), weekday=dt_SA(2)))
# Return the datetime of the second to last Sunday before time1.
e_set("previous_previous_Sunday", dt_add(v("time1"), weekday=dt_SU(op_neg(2))))
```

- Result

```
next_Monday : 2019-06-10 02:41:26
next_next_Saturday : 2019-06-15 02:41:26
previous_Tuesday : 2019-06-04 2:41:26
previous_previous_Sunday : 2019-05-26 02:41:26
time1 : 2019-06-04 2:41:26
```

> **Note** If `time 1` is a Tuesday, the last Tuesday and the next Tuesday are the days that are one week before or after `time 1`.

15.4. Mask sensitive data

When you transform, ship, or use data, you can configure data masking rules to reduce the exposure of sensitive data. This way, you can mitigate the risk of data leaks in an efficient manner. This topic describes how to use functions to mask sensitive data in various scenarios.

Background information

Data masking is commonly used to mask sensitive data such as mobile phone numbers, bank card numbers, email addresses, IP addresses, AccessKey pairs, ID numbers, URLs, order numbers, and strings. When you transform data in the Log Service console, you can use one of the following solutions to mask data: replacement by regular expressions (key function *regex_replace*), Base64 transcoding (key function *base64_encoding*), MD5 encoding (key function *md5_encoding*), str_translate mapping (key function *str_translate*), Grok capture (key function *grok*). For more information, see Regular expression functions, Grok function, and Encoding and decoding functions.

Scenario 1: Mask mobile phone numbers

- Solution

  To mask the mobile phone numbers in log entries, you can use the *regex_replace* function.

- Example
  - Raw log entry

```
iphone: 13900001234
```

- Domain-specific language (DSL) orchestration

```
e_set(
    "sec_iphone",
    regex_replace(v("iphone"), r"(\d{0,3})\d{4}(\d{4})", replace=r"\1****\2"),
)
```

- Result:

```
iphone: 13900001234
sec_iphone: 139****1234
```

Scenario 2: Mask bank card information

- Solution

  To mask the bank card information in log entries, for example, bank card numbers, you can use the *regex_replace* function.

- Example
  - Raw log entry

```
content: bank number is 491648411333978312 and credit card number is 4916484113339780
```

  - DSL orchestration

```
e_set(
    "bank_number",
    regex_replace(
        v("content"), r"([1-9]{1})(\d{14}|\d{13}|\d{11})(\d{4})", replace=r"****\3"
    ),
)
```

  - Result

```
content: bank number is 491648411333978312 and credit card number is 4916484113339780
bank_number: bank number is ****8312 and credit card number is ****9780
```

Scenario 3: Mask email addresses

- Solution

  To mask the email addresses in log entries, you can use the *regex_replace* function.

- Example
  - Raw log entry

```
content: email is twiss2345@aliyun.com
```

- DSL orchestration

```
e_set(
    "email_encrypt",
    regex_replace(
        v("content"),
        r"[A-Za-z\d]+([-_.][A-Za-z\d]+)*(@([A-Za-z\d]+[-.])+[A-Za-z\d]{2,4})",
        replace=r"****\2",
    ),
)
```

- Result

```
content: email is twiss2345@aliyun.com
email_encrypt: email is ****@aliyun.com
```

Scenario 4: Mask AccessKey pairs

- Solution

  To mask the AccessKey pairs in log entries, you can use the *regex_replace* function.

- Example
  - Raw log entry

```
content: ak id is rDhc9qxjhIhlBiyphP7buo5yg5h6Eq and ak key is XQr1EPtfnlZLYlQc
```

  - DSL orchestration

```
e_set(
    "akid_encrypt",
    regex_replace(
        v("content"),
        r"([a-zA-Z0-9]{4})(([a-zA-Z0-9]{26})|([a-zA-Z0-9]{12}))",
        replace=r"\1****",
    ),
)
```

  - Result

```
content: ak id is rDhc9qxjhIhlBiyphP7buo5yg5h6Eq and ak key is XQr1EPtfnlZLYlQc
akid_encrypt: ak id is rDhc**** and ak key is XQr1****
```

Scenario 5: Mask IP addresses

- Solution

  To extract and mask the IP addresses in log entries, you can use the *regex_replace* function and the *grok* function.

- Example
  - Raw log entry

```
content: ip is 192.168.1.1
```

- DSL orchestration

```
e_set("ip_encrypt",regex_replace(v('content'), grok('(%{IP})'), replace=r"****"))
```

- Result

```
content: ip is 192.168.1.1
ip_encrypt: ip is ****
```

Scenario 6: Mask ID card numbers

- Solution

  To mask the ID card numbers in log entries, you can use the *regex_replace* function and the *grok* function.

- Example
  - Raw log entry

```
content: Id card is 11010519491231002X
```

  - DSL orchestration

```
e_set(
    "id_encrypt", regex_replace(v("content"), grok("(%{CHINAID})"), replace=r"\1****")
)
```

  - Result

```
content: Id card is 11010519491231002X
id_encrypt: Id card is 110105****
```

Scenario 7: Mask URLs

- Solution

  To mask the URLs in log entries, you can convert the URLs to plaintext and then use the Base64 encoding and decoding functions to transcode the URLs.

- Example
  - Raw log entry

```
url: https://www.aliyun.com/sls?logstore
```

  - DSL orchestration

```
e_set("base64_url",base64_encoding(v("url")))
```

  - Result

```
url: https://www.aliyun.com/sls?logstore
base64_url: aHR0cHM6Ly93d3cuYWxpeXVuLmNvbS9zbHM/bG9nc3RvcmU=
```

> **Note** To decode the value of the `base64_url` field, you can use the `base64_decoding(v("base64_url"))` function.

Scenario 8: Mask order numbers

- Solution

  To mask the order numbers in log entries and prevent other users from decoding the order numbers, you can use the MD5 encoding function to encode the order numbers.

- Example
  - Raw log entry

    ```
    orderId: 15121412314
    ```

  - DSL orchestration

    ```
    e_set("md5_orderId",md5_encoding(v("orderId")))
    ```

  - Result

    ```
    orderId: 15121412314
    md5_orderId: 852751f9aa48303a5691b0d020e52a0a
    ```

Scenario 9: Mask strings

- Solution

  To mask the strings in log entries, you can use the *str_translate* function to configure mapping rules for the strings.

- Example
  - Raw log entry

    ```
    data: message level is info_
    ```

  - DSL orchestration

    ```
    e_set("data_translate", str_translate(v("data"),"aeiou","12345"))
    ```

  - Result

    ```
    data: message level is info
    data_translate: m2ss1g2 l2v2l 3s 3nf4
    ```

15.5. Text parsing

15.5.1. Parse Syslog messages in standard formats

Syslog is an industry-standard protocol that can be used to record device logs. Syslog is commonly used in network management tools, security management systems, and log audit systems. This topic describes how to use the Grok function in the Domain Specific Language (DSL) of Log Service to parse Syslog messages in different formats.

Overview

Syslog is widely used for message logging in UNIX-like operating systems. Syslog messages can be recorded in local files or sent to Syslog servers over the Internet. Each server can store and parse Syslog messages of multiple devices.

Background information

Traditional Syslog messages are in random formats due to a lack of Syslog standards. In some cases, Syslog messages may fail to be parsed because they can be considered only as strings. Therefore, correctness is the first priority when you parse Syslog messages in different formats.

Syslog protocols

Two Syslog protocols are commonly used in the industry: RFC 5424 issued in 2009 and RFC 3164 issued in 2001. This section describes the differences between the two protocols to help you better use the Grok function to parse Syslog messages.

- RFC 5424

  Syslog messages that use the RFC 5424 protocol contain the following fields. For more information, see RFC 5424 - The Syslog Protocol.

  ```
  PRI VERSION SP TIMESTAMP SP HOSTNAME SP APP-NAME SP PROCID SP MSGID
  ```

  The following examples describe these fields:

```
"""
Example1:
<34>1 2019-07-11T22:14:15.003Z aliyun.example.com ali - ID47 - BOM'su root' failed for lo
nvick on /dev/pts/8
"""
PRI -- 34
VERSION -- 1
TIMESTAMP -- 2019-07-11T22:14:15.003Z
HOSTNAME -- aliyun.example.com
APP-NAME -- ali
PROCID -- None
MSGID -- ID47
MESSAGE -- 'su root' failed for lonvick on /dev/pts/8
"""
Example2:
<165>1 2019-07-11T22:14:15.000003-07:00 192.0.2.1 myproc 8710 - - %% It's time to make th
e do-nuts.
"""
PRI -- 165
VERSION -- 1
TIMESTAMP -- 2019-07-11T05:14:15.000003-07:00
HOSTNAME -- 192.0.2.1
APP-NAME -- myproc
PROCID -- 8710
STRUCTURED-DATA -- "-"
MSGID -- "-"
MESSAGE -- "%% It's time to make the do-nuts."
"""
Example3: - with STRUCTURED-DATA
<165>1 2019-07-11T22:14:15.003Z aliyun.example.com
          evntslog - ID47 [exampleSDID@32473 iut="3" eventSource=
          "Application" eventID="1011"] BOMAn application
          event log entry...
"""
PRI -- 165
VERSION -- 1
TIMESTAMP -- 2019-07-11T22:14:15.003Z
HOSTNAME -- aliyun.example.com
APP-NAME -- evntslog
PROCID -- "-"
MSGID -- ID47
STRUCTURED-DATA -- [exampleSDID@32473 iut="3" eventSource="Application" eventID="1011"]
MESSAGE -- An application event log entry...
```

- RFC 3164

  Syslog messages that use the RFC 3164 protocol contain the following fields. For more information, see RFC 3164 - The BSD Syslog Protocol.

  ```
  PRI HEADER[TIME HOSTNAME] MSG
  ```

  The following example describes these fields:

```
"""
<30>Oct 9 22:33:20 hlfedora auditd[1787]: The audit daemon is exiting.
"""
PRI -- 30
HEADER
- TIME -- Oct 9 22:33:20
- HOSTNAME -- hlfedora
MSG
- TAG -- auditd[1787]
- Content --The audit daemon is exiting.
```

Parse Syslog messages in common formats by using the Grok function

This section describes how to use the Grok function to parse Syslog messages in common formats. For more information about Grok rules, see Grok patterns.

- TraditionalFormat
  - Raw log entry

    ```
      receive_time: 1558663265
      __topic__:
      content: May  5 10:20:57 iZbp1a65x3r1vhpe94fi2qZ systemd: Started System Logging Serv
    ice.
    ```

  - DSL orchestration

    ```
    e_regex('content', grok('%{SYSLOGBASE} %{GREEDYDATA:message}'))
    ```

  - Result

    ```
      receive_time: 1558663265
      __topic__:
      content: May  5 10:20:57 iZbp1a65x3r1vhpe94fi2qZ systemd: Started System Logging Serv
    ice.
      timestamp: May  5 10:20:57
      logsource: iZbp1a65x3r1vhpe94fi2qZ
      program: systemd
      message: Started System Logging Service.
    ```

- FileFormat
  - Raw log entry

    ```
      receive_time: 1558663265
      __topic__:
      content: 2019-05-06T09:26:07.874593+08:00 iZbp1a65x3r1vhpe94fi2qZ root: 834753
    ```

  - DSL orchestration

    ```
    e_regex('content',grok('%{TIMESTAMP_ISO8601:timestamp} %{SYSLOGHOST:hostname} %{SYSLOGP
    ROG} %{GREEDYDATA:message}'))
    ```

- Result

```
  receive_time: 1558663265
  __topic__:
  content: 2019-05-06T09:26:07.874593+08:00 iZbp1a65x3r1vhpe94fi2qZ root: 834753
  timestamp: 2019-05-06T09:26:07.874593+08:00
  hostname: iZbp1a65x3r1vhpe94fi2qZ
  program: root
  message: 834753
```

- RSYSLOG_SyslogProtocol23Format
  - Raw log entry

```
  receive_time: 1558663265
  __topic__:
  content: <13>1 2019-05-06T11:50:16.015554+08:00 iZbp1a65x3r1vhpe94fi2qZ root - - - tw
ish
```

  - DSL orchestration

```
e_regex('content',grok('%{POSINT:priority}>%{NUMBER:version} %{TIMESTAMP_ISO8601:timest
amp} %{syslogHOST:hostname} %{PROG:program} - - - %{GREEDYDATA:message}'))
```

  - Result

```
  receive_time: 1558663265
  __topic__:
  content: <13>1 2019-05-06T11:50:16.015554+08:00 iZbp1a65x3r1vhpe94fi2qZ root - - - tw
ish
  priority: 13
  version: 1
  timestamp: 2019-05-06T11:50:16.015554+08:00
  hostname: iZbp1a65x3r1vhpe94fi2qZ
  program: root
  message: twish
```

- RSYSLOG_DebugFormat
  - Log content

```
  receive_time: 1558663265
  __topic__:
  content: 2019-05-06T14:29:37.558854+08:00 iZbp1a65x3r1vhpe94fi2qZ root: environment
```

  - DSL orchestration

```
e_regex('content',grok('%{TIMESTAMP_ISO8601:timestamp} %{SYSLOGHOST:hostname} %{SYSLOGP
ROG} %{GREEDYDATA:message}'))
```

- Result

```
  receive_time: 1558663265
  __topic__:
  content: 2019-05-06T14:29:37.558854+08:00 iZbp1a65x3r1vhpe94fi2qZ root: environment
  timestamp: 2019-05-06T14:29:37.558854+08:00
  hostname: iZbp1a65x3r1vhpe94fi2qZ
  program: root
  message: environment
```

Parse Syslog messages in uncommon formats by using the Grok function

This section describes how to use the Grok function to parse Syslog messages in two uncommon formats: Fluent RFC5424 and Fluent RFC3164. These messages are collected by using the Fluent software.

- Fluent RFC5424
  - Log content

```
  receive_time: 1558663265
  __topic__:
  content: <16>1 2019-02-28T12:00:00.003Z 192.168.0.1 aliyun 11111 ID24224 [exampleSDID
@20224 iut='3' eventSource='Application' eventID='11211] Hi, from Fluentd!
```

  - DSL orchestration

```
e_regex('content',grok('%{POSINT:priority}>%{NUMBER:version} %{TIMESTAMP_ISO8601:timest
amp} %{SYSLOGHOST:hostname} %{WORD:ident} %{USER:pid} %{USERNAME:msgid} (? P<extradata>
(\[(. *)\]|[^ ])) %{GREEDYDATA:message}'))
```

  - Result

```
  receive_time: 1558663265
  __topic__:
  content: <16>1 2019-02-28T12:00:00.003Z 192.168.0.1 aliyun 11111 ID24224 [exampleSDID
@20224 iut='3' eventSource='Application' eventID='11211] Hi, from aliyun!
  priority: 16
  version: 1
  timestamp: 2019-02-28T12:00:00.003Z
  hostname: 192.168.0.1
  ident: aliyun
  pid: 1111
  msgid: ID24224
  extradata: [exampleSDID@20224 iut='3' eventSource='Application' eventID='11211]
  message: Hi, from aliyun!
```

- Fluent RFC3164
  - Log content

```
  receive_time: 1558663265
  __topic__:
  content: <6>Feb 28 12:00:00 192.168.0.1 aliyun[11111]: [error] Syslog test
```

- DSL orchestration

```
e_regex('content', grok('%{POSINT:priority}>%{SYSLOGTIMESTAMP:timestamp} %{SYSLOGHOST:h
ostname} %{WORD:ident}(? P<pid>(\[[a-zA-Z0-9. _-]+\]|[^:])): (? P<level>(\[(\w+)\]|[^ ]
)) %{GREEDYDATA:message}'))
```

- Result

```
 receive_time: 1558663265
  __topic__:
  content: <6>Feb 28 12:00:00 192.168.0.1 aliyun[11111]: [error] Syslog test
  priority: 6
  timestamp: Feb 28 12:00:00
  hostname: 192.168.0.1
  ident: aliyun
  pid: [1111]
  level: [error]
  message: Syslog test
```

- Parse the priority field in a log entry

  After you parse the Syslog messages in the FluentRFC5424 and FluentRFC3164 formats, you can further parse the priority field to obtain information about facility and severity. For more information about how to use RFC5424, see e_syslogrfc. Example

  - Raw log entry

```
receive_time: 1558663265
  __topic__:
  content: <13>1 2019-05-06T11:50:16.015554+08:00 iZbp1a65x3r1vhpe94fi2qZ root - - - tw
ish
  priority: 13
  version: 1
  timestamp: 2019-05-06T11:50:16.015554+08:00
  hostname: iZbp1a65x3r1vhpe94fi2qZ
  program: root
  message: twish
```

  - DSL orchestration

```
e_syslogrfc("priority","SYSLOGRFC5424")
```

- Result

```
  receive_time: 1558663265
  __topic__:
  content: <13>1 2019-05-06T11:50:16.015554+08:00 iZbp1a65x3r1vhpe94fi2qZ root - - - tw
ish
  priority: 13
  version: 1
  timestamp: 2019-05-06T11:50:16.015554+08:00
  hostname: iZbp1a65x3r1vhpe94fi2qZ
  program: root
  message: twish
  _facility_: 1
  _severity_: 5
  _severitylabel_: Notice: normal but significant condition
  _facilitylabel_: user-level messages
```

15.5.2. Parse NGINX logs

NGINX access logs record the detailed information of user access requests. You can parse NGINX access logs to monitor and analyze your business. This topic describes how to use regular expressions or the Grok function to parse NGNIX access logs.

Parsing methods

Log Service allows you to parse NGNIX logs by using regular expressions or the Grok function.

- Use regular expressions.

  If you are unfamiliar with regular expressions, using regular expressions to parse logs may be difficult, inefficient, and time consuming. Therefore, we recommend that you use the Grok function instead of regular expressions to parse logs. For more information about regular expressions, see Regular expressions.

- (Recommended) Use the Grok function.

  Compared with regular expressions, the Grok function is easier to learn. You can use this function to parse logs if you are familiar with the field types in different Grok patterns. The Grok function is superior to regular expressions in terms of flexibility, efficiency, cost effectiveness, and learning curves. Log Service supports 400 Grok patterns for data transformation. We recommend that you use this function to parse logs. For more information about Grok patterns, see Grok patterns.

> **Note**
>
> - You can combine regular expressions and the Grok function to parse logs.
> - You can customize regular expressions or the Grok function to parse NGINX logs that are in a custom format.

Use regular expressions to parse NGINX access logs that contain a success status code

The following example shows how to use regular expressions to parse NGINX access logs that contain a success status code.

- Raw log entry

```
__source__:  192.168.0.1
__tag__:__client_ip__:  192.168.254.254
__tag__:__receive_time__:  1563443076
content: 192.168.0.2 - - [04/Jan/2019:16:06:38 +0800] "GET http://example.aliyundoc.com/_
astats?application=&inf.name=eth0 HTTP/1.1" 200 273932 "-" "Mozilla/5.0 (compatible; Goog
lebot/2.1; +http://www.example.com/bot.html)"
```

- Requirements
  - Requirement 1: Extract the code, ip, datetime, protocol, request, sendbytes, refere, useragent, and verb fields from the NGINX logs.
  - Requirement 2: Extract the uri_proto, uri_domain, and uri_param fields from the request field.
  - Requirement 3: Extract the uri_path and uri_query fields from the uri_param field.
- DSL orchestration
  - General orchestration

```
"""Step 1: Parse the NGINX logs."""
e_regex("content",r'(? P<ip>\d+\.\d+\.\d+\.\d+)( - - \[)(? P<datetime>[\s\S]+)\] \"(? P
<verb>[A-Z]+) (? P<request>[\S]*) (? P<protocol>[\S]+)["] (? P<code>\d+) (? P<sendbytes
>\d+) ["](? P<refere>[\S]*)["] ["](? P<useragent>[\S\s]+)["]')
"""Step 2: Parse the request field obtained in Step 1."""
e_regex('request',r'(? P<uri_proto>(\w+)):\/\/(? P<uri_domain>[a-z0-9.] *[^\/])(? P<uri
_param>(. +)$)')
"""Step 3: Parse the uri_param field obtained in Step 2."""
e_regex('uri_param',r'(? P<uri_path>\/\_[a-z]+[^?]) \?(? <uri_query>(. +)$)')
```

- Specific orchestration and the transformation results
  - Orchestration specific to Requirement 1:

```
e_regex("content",r'(? P<ip>\d+\.\d+\.\d+\.\d+)( - - \[)(? P<datetime>[\s\S]+)\] \"(?
P<verb>[A-Z]+) (? P<request>[\S]*) (? P<protocol>[\S]+)["] (? P<code>\d+) (? P<sendby
tes>\d+) ["](? P<refere>[\S]*)["] ["](? P<useragent>[\S\s]+)["]')
```

    Sub-result

```
__source__:  192.168.0.1
__tag__:  __receive_time__:  1563443076
code:  200
content:  192.168.0.2 - - [04/Jan/2019:16:06:38 +0800] "GET http://example.aliyundoc.
com/_astats?application=&amp;inf.name=eth0 HTTP/1.1" 200 273932 "-" "Mozilla/5.0 (com
patible; Googlebot/2.1; +http://www.example.com/bot.html)"httpversion:  1.1
datetime:  04/Jan/2019:16:06:38 +0800
ip:  192.168.0.2
protocol:  HTTP/1.1
refere:  -
request:  http://example.aliyundoc.com/_astats?application=&amp;inf.name=eth0
sendbytes:  273932
useragent:  Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.example.com/bot.html)
verb:  GET
```

  - Orchestration specific to Requirement 2 (Parse the request field).

```
e_regex('request',r'(? P<uri_proto>(\w+)):\/\/(? P<uri_domain>[a-z0-9.] *[^\/])(? P<u
ri_param>(. +)$)')
```

    Sub-result

```
uri_param: /_astats? application=&inf.name=eth0
uri_domain: example.aliyundoc.com
uri_proto: http
```

  - Orchestration specific to Requirement 3 (Parse the uri_param field).

```
e_regex('uri_param',r'(? P<uri_path>\/\_[a-z]+[^?]) \?(? <uri_query>(. +)$)')
```

    Sub-result

```
uri_path: /_astats
uri_query: application=&inf.name=eth0
```

- Result

```
__source__:  192.168.0.1
__tag__:  __receive_time__:  1563443076
code:  200
content:  192.168.0.2 - - [04/Jan/2019:16:06:38 +0800] "GET http://example.aliyundoc.com/
_astats?application=&amp;inf.name=eth0 HTTP/1.1" 200 273932 "-" "Mozilla/5.0 (compatible;
Googlebot/2.1; +http://www.example.com/bot.html)"httpversion:  1.1
datetime:  04/Jan/2019:16:06:38 +0800
ip:  192.168.0.2
protocol:  HTTP/1.1
refere:  -
request:  http://example.aliyundoc.com/_astats?application=&amp;inf.name=eth0
sendbytes:  273932
uri_domain:  example.aliyundoc.com
uri_proto:  http
uri_param: /_astats? application=&inf.name=eth0
uri_path: /_astats
uri_query: application=&inf.name=eth0
useragent:  Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.example.com/bot.html)
verb:  GET
```

Use the Grok function to parse NGINX access logs that contain a success status code.

The following example shows how to use the Grok function to parse NGINX access logs that contain a success status code.

- Raw log entry

```
__source__:  192.168.0.1
__tag__:__client_ip__:  192.168.254.254
__tag__:__receive_time__:  1563443076
content: 192.168.0.2 - - [04/Jan/2019:16:06:38 +0800] "GET http://example.aliyundoc.com/_
astats?application=&inf.name=eth0 HTTP/1.1" 200 273932 "-" "Mozilla/5.0 (compatible; Goog
lebot/2.1; +http://www.example.com/bot.html)"
```

- Requirements
  - Requirement 1: Extract the clientip, bytes, agent, auth, verb, request, ident, timestamp, httpversion, response, and referrer fields from the NGINX logs.
  - Requirement 2: Extract the uri_proto, uri_domain, and uri_param fields from the request field.
  - Requirement 3: Extract uri_path and uri_query fields from the uri_param field.
- DSL orchestration

- General orchestration

```
"""Step 1: Parse the NGINX logs."""
e_regex('content',grok('%{COMBINEDAPACHELOG}'))
"""Step 2: Parse the request field obtained in Step 1."""
e_regex('request',grok("%{URIPROTO:uri_proto}://(?:%{USER:user}(?::[^@]*)? @)?(?:%{URIH
OST:uri_domain})?(?:%{URIPATHPARAM:uri_param})?"))
"""Step 3: Parse the uri_param field obtained in Step 2."""
e_regex('uri_param',grok("%{GREEDYDATA:uri_path}\? %{GREEDYDATA:uri_query}"))
```

To use the Grok function to parse the NGINX logs, you only need to use the `COMBINEDAPACHELOG` pattern.

Pattern	Rule	Description
COMMONAPACHELOG	`%{IPORHOST:clientip} %{HTTPDUSER:ident} %{USER:auth} \[%{HTTPDATE:timestamp}\] "(?:%{WORD:verb} %{NOTSPACE:request}(?: HTTP/%{NUMBER:httpversion})? |%{DATA:rawrequest})" %{NUMBER:response} (?:%{NUMBER:bytes}|-)`	Parses the clientip, ident, auth, timestamp, verb, request, httpversion, response, and bytes fields.
COMBINEDAPACHELOG	`%{COMMONAPACHELOG} %{QS:referrer} %{QS:agent}`	Parses all the fields in the COMMONAPACHELOG pattern, and parses the referrer and agent fields.

- Specific orchestration and the transformation results

- Orchestration specific to Requirement 1:

```
e_regex('content',grok('%{COMBINEDAPACHELOG}'))
```

Sub-result

```
clientip: 192.168.0.1
__tag__:  __receive_time__:  1563443076
agent:  "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.example.com/bot.html)"
auth:  -
bytes:  273932
clientip:  192.168.0.2
content:  192.168.0.2 - - [04/Jan/2019:16:06:38 +0800] "GET http://example.aliyundoc.
com/_astats?application=&amp;inf.name=eth0 HTTP/1.1" 200 273932 "-" "Mozilla/5.0 (com
patible; Googlebot/2.1; +http://www.example.com/bot.html)"
httpversion:  1.1
ident:  -
referrer:  "-"
request:  http://example.aliyundoc.com/_astats?application=&amp;inf.name=eth0
response:  200
timestamp:  04/Jan/2019:16:06:38 +0800
verb:  GET
```

- Orchestration specific to Requirement 2 (Parse the request field).

```
e_regex('request',grok("%{URIPROTO:uri_proto}://(?:%{USER:user}(?::[^@]*)? @)?(?:%{UR
IHOST:uri_domain})?(?:%{URIPATHPARAM:uri_param})?"))
```

Sub-result

```
uri_proto: http
uri_domain: example.aliyundoc.com
uri_param: /_astats? application=&inf.name=eth0
```

You can use the Grok patterns to parse the request field. The following table describes the patterns.

Pattern	Rule	Description
URIPROTO	`[A-Za-z]+(\+[A-Za-z+]+)?`	Matches URI schemes. For example, in `http://hostname.domain.tld/_astats?application=&inf.name=eth0`, the matched content is http.
USER	`[a-zA-Z0-9._-]+`	Matches content that contains letters, digits, and `._-`.
URIHOST	`%{IPORHOST}(?::%`	Matches IP addresses, hostnames, or positive integers.
URIPATHPARAM	`%{URIPATH}(?:%{URIPARAM})?`	Matches the uri_param field.

- Orchestration specific to Requirement 3 (Parse the uri_param field).

```
e_regex('uri_param',grok("%{GREEDYDATA:uri_path}\? %{GREEDYDATA:uri_query}"))
```

Sub-result

```
uri_path: /_astats
uri_query: application=&inf.name=eth0
```

The following table describes the Grok pattern that is used to parse the uri_param field.

Pattern	Rule	Description
GREEDYDATA	`.*`	Matches zero or multiple characters that are not line breaks.

- Result

```
__source__:  192.168.0.1
__tag__:__receive_time__:  1563443076
agent:  "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.example.com/bot.html)"
auth:  -
bytes:  273932
clientip:  192.168.0.2
content:  192.168.0.2 - - [04/Jan/2019:16:06:38 +0800] "GET http://example.aliyundoc.com/
_astats?application=&amp;inf.name=eth0 HTTP/1.1" 200 273932 "-" "Mozilla/5.0 (compatible;
Googlebot/2.1; +http://www.example.com/bot.html)"
httpversion:  1.1
ident:  -
referrer:  "-"
request:  http://example.aliyundoc.com/_astats?application=&amp;inf.name=eth0
response:  200
timestamp:  04/Jan/2019:16:06:38 +0800
uri_domain:  example.aliyundoc.com
uri_param:  /_astats? application=&amp;inf.name=eth0
uri_path:  /_astats
uri_proto:  http
uri_query:  application=&amp;inf.name=eth0
verb:  GET
```

Use the Grok function to parse NGINX access logs that contain an error status code

The following example shows how to use the Grok function to parse NGINX access logs that contain an error status code.

- Raw log entry

```
__source__:  192.168.0.1
__tag__:__client_ip__:  192.168.254.254
__tag__:__receive_time__:  1563443076
content: 2019/08/07 16:05:17 [error] 1234#1234: *1234567 attempt to send data on a closed
socket: u:111111ddd, c:0000000000000000, ft:0 eof:0, client: 1.2.3.4, server: sls.aliyun.
com, request: "GET /favicon.ico HTTP/1.1", host: "sls.aliyun.com", referrer: "https://sls
.aliyun.com/question/answer/123.html?from=singlemessage"
```

- Requirement:

  Parse the host, http_version, log_level, pid, referrer, request, request_time, server, and verb fields from the content field.

- DSL orchestration:

```
e_regex('content',grok('%{DATESTAMP:request_time} \[%{LOGLEVEL:log_level}\] %{POSINT:pid}
#%{NUMBER}: %{GREEDYDATA:errormessage}(?:, client: (? <client>%{IP}|%{HOSTNAME}))(?:, ser
ver: %{IPORHOST:server})(?:, request: "%{WORD:verb} %{NOTSPACE:request}( HTTP/%{NUMBER:ht
tp_version})")(?:, host: "%{HOSTNAME:host}")?(?:, referrer: "%{NOTSPACE:referrer}")?'))
```

- Result

```
___source__:  192.168.0.1
__tag__:__client_ip__:  192.168.254.254
__tag__:__receive_time__:  1563443076
content:  2019/08/07 16:05:17 [error] 1234#1234: *1234567 attempt to send data on a close
d socket: u:111111ddd, c:0000000000000000, ft:0 eof:0, client: 1.2.3.4, server: sls.aliyu
n.com, request: "GET /favicon.ico HTTP/1.1", host: "sls.aliyun.com", referrer: "https://s
ls.aliyun.com/question/answer/123.html?
host: sls.aliyun.com
http_version: 1.1
log_level: error
pid: 1234
referrer: https://sls.aliyun.com/question/answer/123.html?from=singlemessage
request: /favicon.ico
request_time:  19/08/07 16:05:17
server: sls.aliyun.com
verb: GET
```

15.5.3. Parse Java error logs

In big data scenarios that require high concurrency, effective analysis of Java error logs can reduce the O&M costs of Java applications. You can use Log Service to collect Java error logs from Alibaba Cloud services and use the data transformation feature to parse the collected logs.

Prerequisites

The Java error logs of Log Service, Object Storage Service (OSS), Server Load Balancer (SLB), and ApsaraDB RDS are collected and stored in a Logstore named cloud_product_error_log. For more information, see Use Logtail to collect logs.

Scenarios

For example, you have developed Java Application A by using multiple Alibaba Cloud services, such as OSS and Log Service. You have created a Logstore named cloud_product_erro_log in the China (Hangzhou) region to store Java error logs that are generated when you call the API operations of the Alibaba Cloud services. To fix Java errors in an efficient manner, you need to use Log Service to analyze the Java error logs at regular intervals.

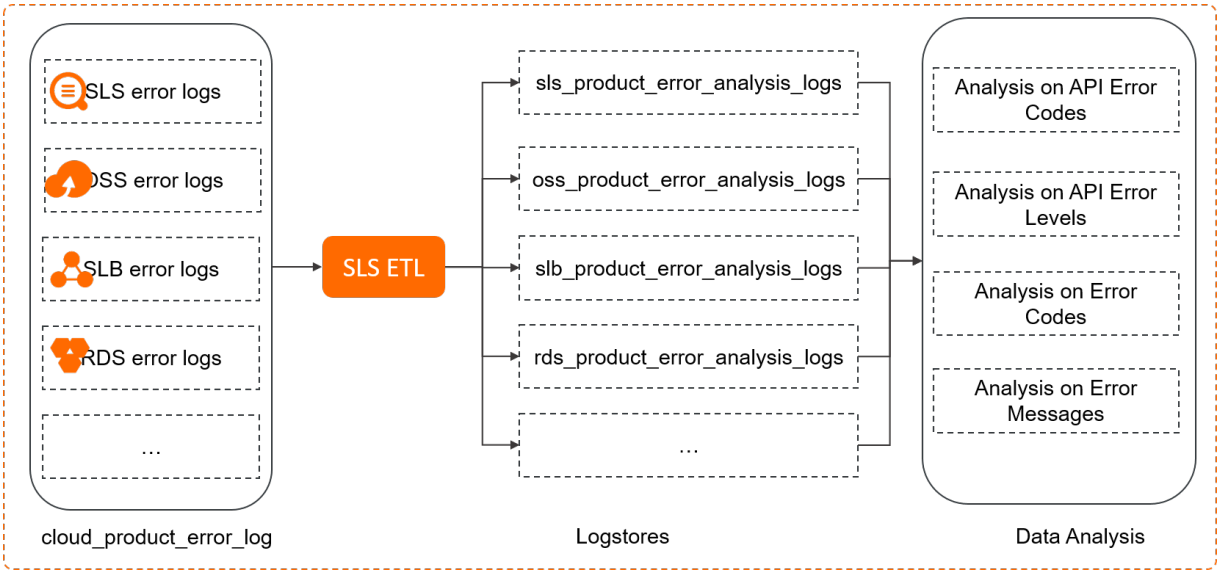
To meet the preceding requirements, you must parse the log time, error code, status code, service name, error message, request method, and error line number from the collected logs, and then send the parsed logs to the Logstore of each cloud service for error analysis.

The following example shows a raw log:

```
__source__:192.0.2.10
__tag__:__client_ip__:203.0.113.10
__tag__:__receive_time__:1591957901
__topic__:
message: 2021-05-15 16:43:35 ParameterInvalid 400
com.aliyun.openservices.log.exception.LogException:The body is not valid json string.
   at com.aliyun.openservice.log.Client.ErrorCheck(Client.java:2161)
   at com.aliyun.openservice.log.Client.SendData(Client.java:2312)
   at com.aliyun.openservice.log.Client.PullLogsk(Client.java:1397)
   at com.aliyun.openservice.log.Client.SendData(Client.java:2265)
   at com.aliyun.openservice.log.Client.GetCursor(Client.java:1123)
   at com.aliyun.openservice.log.Client.PullLogs(Client.java:2161)
   at com.aliyun.openservice.log.Client.ErrorCheck(Client.java:2426)
   at transformEvent.main(transformEvent.java:2559)
```

Procedure

The error logs of Application A are collected by using Logtail and are stored in the cloud_product_error_log Logstore. Then, the error logs are transformed and the transformed logs are sent to the Logstore of each cloud service for error analysis. The procedure consists of the following steps:



1. Design a data transformation statement: In this step, analyze the transformation logic and write a transformation statement.
2. Create a data transformation task: In this step, send logs to different Logstores of cloud services

for error analysis.

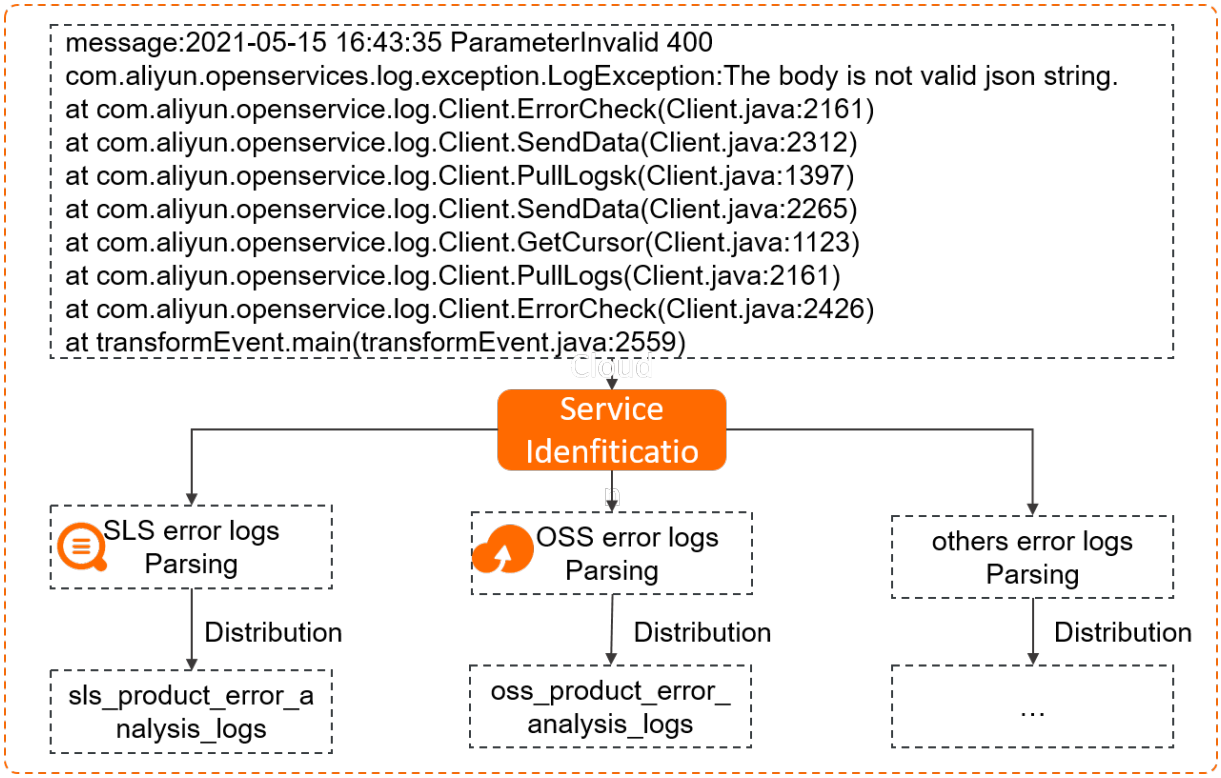
3. Query and analyze data: In this step, analyze error logs in the Logstore of each cloud service.

Step 1: Design a data transformation statement

Transformation procedure

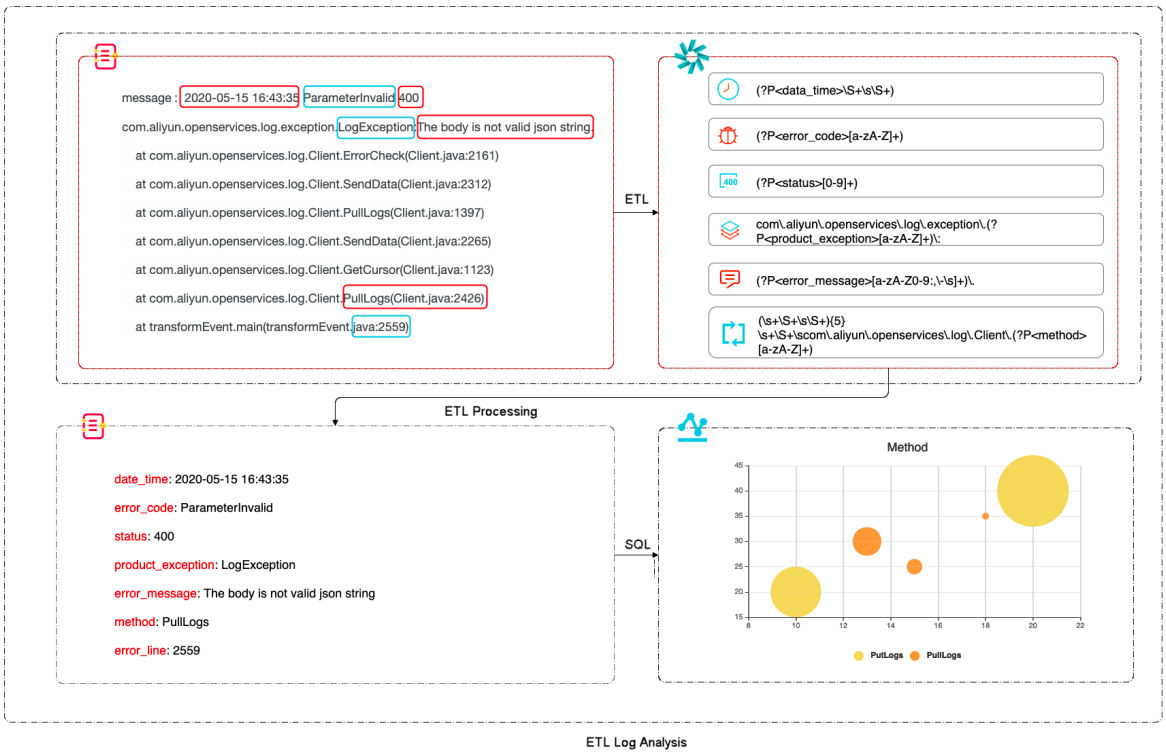
To analyze error logs in a convenient manner, you must complete the following operations:

1. Extract the log time, error code, status code, service name, error message, request method, and error line number from the message field.
2. Send error logs to the Logstore of each cloud service.



Transformation logic

In this case, you must analyze the log time, error code, status code, service name, error message, request method, and error line number in the raw log field, and then design regular expressions for each field that you want to extract.



1. Use the regex_match function to match logs that contain LogException. For more information, see regex_match.
2. If a log contains LogException, the log is transformed based on the transformation rule of Log Service error logs. If a log contains OSSException, the log is transformed based on the transformation rule of OSS error logs. For more information, see e_switch.
3. Use the e_regex function to parse error logs for each cloud service. For more information, see e_regex.
4. Delete the message field and send error logs to the Logstore of the corresponding cloud service. For more information, see e_drop_fields and e_output and e_coutput.
5. For more information, see the Group section in Regular expressions.

Syntax description

Transformation statement syntax

The following example shows the specific syntax of a data transformation statement:

```
e_switch(
    regex_match(v("message"), r"LogException"),
    e_compose(
        e_regex(
            "message",
            "(?P<data_time>\S+\s\S+)\s(?P<error_code>[a-zA-Z]+)\s(?P<status>[0-9]+)\scom\.a
liyun\.openservices\.log\.exception\.(?P<product_exception>[a-zA-Z]+)\:(?P<error_message>[a
-zA-Z0-9:,\-\s]+)\.(\s+\S+\s\S+){5}\s+\S+\scom\.aliyun\.openservices\.log\.Client\.(?P<meth
od>[a-zA-Z]+)\S+\s+\S+\stransformEvent\.main\(transformEvent\.java\:(?P<error_line>[0-9]+)\
)",
        ),
        e_drop_fields("message"),
        e_output("sls-error"),
    ),
    regex_match(v("message"), r"OSSException"),
    e_compose(
        e_regex(
            "message",
            "(?P<data_time>\S+\s\S+)\scom\.aliyun\.oss\.(?P<product_exception>[a-zA-Z]+)\:(
?P<error_message>[a-zA-Z0-9,\s]+)\.\n\[ErrorCode\]\:\s(?P<error_code>[a-zA-Z]+)\n\[RequestI
d\]\:\s(?P<request_id>[a-zA-Z0-9]+)\n\[HostId\]\:\s(?P<host_id>[a-zA-Z-.]+)\n\S+\n\S+(\s\S+
){3}\n\s+\S+\s+(.+)(\s+\S+){24}\scom\.aliyun\.oss\.OSSClient\.(?P<method>[a-zA-Z]+)\S+\s+\S
+\stransformEvent\.main\(transformEvent\.java:(?P<error_line>[0-9]+)\)",
        ),
        e_drop_fields("message"),
        e_output("oss-error"),
    ),
)
```

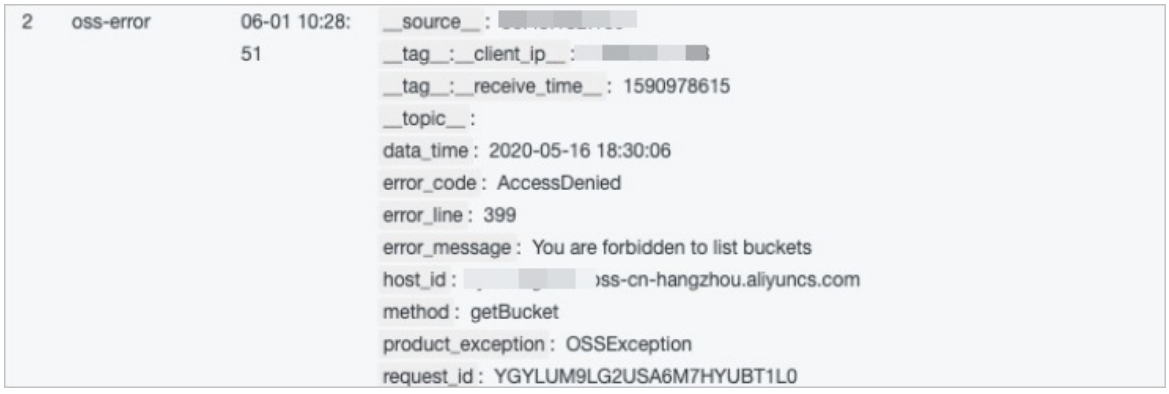
Step 2: Create a data transformation task

1. Go to the data transformation page.
   i.
   ii.
   iii. On the Search & Analysis page, click **Data Transformation**.
2. In the upper-right corner of the page, specify a time range for the required log data.

   Make sure that log data exists on the **Raw Logs** tab.
3. In the edit box, enter the following data transformation statement:

```
e_switch(
    regex_match(v("message"), r"LogException"),
    e_compose(
        e_regex(
            "message",
            "(?P<data_time>\S+\s\S+)\s(?P<error_code>[a-zA-Z]+)\s(?P<status>[0-9]+)\scom\.aliyun\.openservices\.log\.exception\.(?P<product_exception>[a-zA-Z]+)\:(?P<error_message>[a-zA-Z0-9:,\-\s]+)\.(\s+\S+\s\S+){5}\s+\S+\scom\.aliyun\.openservices\.log\.Client\.(?P<method>[a-zA-Z]+)\S+\s+\S+\stransformEvent\.main\(transformEvent\.java\:(?P<error_line>[0-9]+)\)",
        ),
        e_drop_fields("message"),
        e_output("sls-error"),
    ),
    regex_match(v("message"), r"OSSException"),
    e_compose(
        e_regex(
            "message",
            "(?P<data_time>\S+\s\S+)\scom\.aliyun\.oss\.(?P<product_exception>[a-zA-Z]+)\:(?P<error_message>[a-zA-Z0-9,\s]+)\.\n\[ErrorCode\]\:\s(?P<error_code>[a-zA-Z]+)\n\[RequestId\]\:\s(?P<request_id>[a-zA-Z0-9]+)\n\[HostId\]\:\s(?P<host_id>[a-zA-Z-.]+)\n\S+\n\S+(\s\S+){3}\n\s+\S+\s+(.+)(\s+\S+){24}\scom\.aliyun\.oss\.OSSClient\.(?P<method>[a-zA-Z]+)\S+\s+\S+\stransformEvent\.main\(transformEvent\.java:(?P<error_line>[0-9]+)\)",
        ),
        e_drop_fields("message"),
        e_output("oss-error"),
    ),
)
```

4. Click **Preview Data**.



5. Create a data transformation task.
   i. Click **Save as Transformation Rule**.

ii. In the **Create Data Transformation Rule** panel, set the parameters and click **OK**. The following table describes the parameters.

Parameter	Description
Rule Name	The name of the transformation rule, for example, test.
Authorization Method	Select **Default Role** to read data from the source Logstore.
Storage Target	
Target Name	The name of the storage target, for example, sls-error or oss-error.
Target Region	The region where the destination project resides, for example, China (Hangzhou).
Target Project	The name of the destination project to which transformed data is saved.
Target Logstore	The name of the destination Logstore to which transformed data is saved, for example, sls-error or oss-error.
Authorization Method	Select **Default Role** to write transformation results to the destination Logstore.
Processing Range	
Time Range	Select **All**.

After you create a data transformation task, a dashboard is automatically created for the task. You can view the metrics of the task on the dashboard.

On the **Exception detail** chart, you can view the logs that failed to be parsed, and then modify the regular expression.

- If a log fails to be parsed, you can specify the severity of the log as WARNING to report the log. The data transformation task continues running.
- If you specify the severity of the log as ERROR to report the log, the data transformation task stops running. In this case, you must identify the cause of the error and modify the regular expression until the data transformation task can parse all required types of error logs.

Step 3: Analyze error logs

After raw error logs are transformed, you can analyze the error logs. In this example, only the Java error logs of Log Service are analyzed.

1.
2.
3. Enter a query statement in the search box.
   - To calculate the number of errors for each request method, execute the following query statement:

     ```
     * | SELECT COUNT(method) as m_ct, method GROUP BY method
     ```

   - To calculate the number of occurrences of each error message for the PutLogs API operation,

execute the following query statement:

```
* | SELECT error_message,COUNT(error_message) as ct_msg, method WHERE method LIKE 'PutLogs' GROUP BY error_message,method
```

- To calculate the number of occurrences for each error code, execute the following query statement:

```
* | SELECT error_code,COUNT(error_code) as count_code GROUP BY error_code
```

- To query the error information of each request method by log time, execute the following query statement:

```
* | SELECT date_format(data_time, '%Y-%m-%d %H:%m:%s') as date_time,status,product_exception,error_line, error_message,method ORDER BY date_time desc
```

4. On the Search & Analysis page, click **15Minutes(Relative)** to specify a time range.

   You can select a relative time or a time frame. You can also specify a custom time range.

   > **Note** The query results may contain logs that are generated 1 minute earlier or later than the specified time range.

5. Click **Search & Analyze** to view the query and analysis result.

15.5.4. Extract dynamic key-value pairs from a string

This topic describes how to extract dynamic key-value pairs from a string by using different functions.

Functions

Extracting dynamic key-value pairs is a process that extracts and transforms keywords and values. You can use the e_kv function, e_kv_delimit function, and e_regex function to extract dynamic key-value pairs. The following table describes the functions that you can use in different scenarios.

Function	Keyword extraction	Value extraction	Keyword transformation	Value transformation
e_kv	Uses specific regular expressions.	Supports the default character set and specific delimiters such as (,) and (").	Supports prefixes and suffixes.	Supports text escape.
e_kv_delimit	Uses specific regular expressions.	Uses delimiters.	Supports prefixes and suffixes.	None.
e_regex	Uses custom regular expressions and the default character set.	Custom.	Custom.	Custom.

In most cases, you can use the `e_kv` function to extract key-value pairs, especially when you need to extract and escape enclosed characters or backslashes (\). In complicated scenarios, you can use the `e_regex` function to extract key-value pairs. In specific scenarios, you need to extract key-value pairs by using the `e_kv_delemit` function.

Extract keywords

- Method

  When you use the `e_kv` function, `e_kv_delimit` function, or `e_regex` function to extract keywords, the functions must comply with the extraction constraints. For more information, see Constraints on field name extraction.

- Example 1

  The following example describes three methods that you can use to extract keywords and values from the `k1: q=asd&a=1&b=2&__1__=3` log entry.

  - Use the e_kv function.
    - Raw log entry

      ```
      k1: q=asd&a=1&b=2&__1__=3
      ```

    - Transformation rule

      ```
      # By default, keywords are extracted by using a specified character set.
      e_kv("k1")
      ```

    - Result

      ```
      k1: q=asd&a=1&b=2
      q: asd
      a: 1
      b: 2
      ```

      Note The keyword `__1__` is not extracted because it does not comply with the extraction constraints. For more information, see Constraints on field name extraction.

  - Use the e_kv_delimit function.
    - Raw log entry

      ```
      k1: q=asd&a=1&b=2&__1__=3
      ```

    - Transformation rule

      ```
      # After the key-value pair is separated by an ampersand (&), extract the keywords by using the ampersand (&).
      e_kv_delimit("k1", pair_sep=r"&")
      ```

    - Result

      ```
      k1: q=asd&a=1&b=2
      q: asd
      a: 1
      b: 2
      ```

- Use the e_regex function.
  - Raw log entry

    ```
    k1: q=asd&a=1&b=2&__1__=3
    ```

  - Transformation rule

    ```
    # Keywords and values are extracted by using a custom character set.
    e_regex("k1",r"(\w+)=([a-zA-Z0-9]+)",{r"\1": r"\2"})
    ```

  - Result

    ```
    k1: q=asd&a=1&b=2
    q: asd
    a: 1
    b: 2
    ```

- Example 2

  The following example describes three methods that you can use to extract keywords from the `content:k1=v1&k2=v2? k3:v3` log entry by using regular expressions:

  - Use the e_kv function.
    - Raw log entry

      ```
      content:k1=v1&k2=v2? k3:v3
      ```

    - Transformation rule

      ```
      e_kv("content",sep="(?:=|:)")
      ```

    - Result

      ```
      content:k1=v1&k2=v2? k3:v3
      k1: v1
      k2: v2
      k3: v3
      ```

      Note When the character set is passed to the `pari_sep`, `kv_sep`, or `sep` field, regular expressions that include a non-capturing group are used in the format of `(?:character set)`.

- Use the e_kv_delimit function.
  - Raw log entry

    ```
    content:k1=v1&k2=v2? k3:v3
    ```

  - Transformation rule

    ```
    e_kv_delimit("content",pair_sep=r"&?",kv_sep="(?:=|:)")
    ```

  - Result

    ```
    content:k1=v1&k2=v2? k3:v3
    k1: v1
    k2: v2
    k3: v3
    ```

- Use the e_regex function.
  - Raw log entry

    ```
    content:k1=v1&k2=v2? k3:v3
    ```

  - Transformation rule

    ```
    e_regex("content",r"([a-zA-Z0-9]+)[=|:]([a-zA-Z0-9]+)",{r"\1": r"\2"})
    ```

  - Result

    ```
    content:k1=v1&k2=v2? k3:v3
    k1: v1
    k2: v2
    k3: v3
    ```

- Example 3

  The following example shows how to use the `e_regex` function to extract keywords from complex strings.

  - Raw log entry

    ```
    content :"ak_id:"LTAiscW,"ak_key:"rsd7r8f
    ```

  - Transformation rule

    If double quotation marks (") exist in front of the keywords, you can use the `e_regex` function.

    ```
    e_regex("str",r'(\w+):(\"\w+)',{r"\1":r"\2"})
    ```

  - Result

    The log format after DSL orchestration:

    ```
    content :"ak_id:"LTAiscW,"ak_key:"rsd7r8f
    ak_id: LTAiscW
    ak_key: rsd7r8f
    ```

Extract values

- Use the `e_kv` function to extract values if clear identifiers exist between dynamic key-value pairs or between keywords and values, such as `a=b`, or `a="cxxx"`. Example:
  - Raw log entry

    ```
    content1:  k="helloworld",the change world, k2="good"
    ```

  - Transformation rule

    In this case, `the change world` is not extracted.

    ```
    e_kv("content1")
    # The syntax of the e_kv_delimit function: A space character is required before k2. The
    refore, k2 can be parsed only when the pair_sep parameter of the e_kv_delimit function
    is set to ",\s".
    e_kv_delimit("content1",kv_sep="=", pair_sep=",\s")
    # The syntax of the e_regex function.
    e_regex("str",r"(\w+)=(\"\w+)",{r"\1": r"\2"})
    ```

  - Result

    The extracted log entry:

    ```
    content1:  k="helloworld",the change world, k2="good"
    k1: helloworld
    k2: good
    ```

- To extract values from log entries that contain the `"` character in the `content:k1="v1=1"&k2=v2? k3=v3` format, we recommend that you use the `e_kv` function.
  - Raw log entry

    ```
    content:k1="v1=1"&k2=v2? k3=v3
    ```

  - Transformation rule

    ```
    e_kv("content",sep="=", quote="'")
    ```

  - Result

    The extracted log entry:

    ```
    content: k1='v1=1'&k2=v2? k3=v3
    k1: v1=1
    k2:v2
    k3:v3
    ```

  If you use the `e_kv_delimit` function to extract values and the syntax is `e_kv_delimit("ctx", pair_sep=r"&?", kv_sep="=")`, only `k2: v2` and `k3: v3` can be parsed. The keyword `k1="v1` in the first key-value pair is dropped because the keyword does not comply with the extraction constraints. For more information, see Constraints on field name extraction.

- Some key-value pairs separated by delimiters contain special characters but they are not enclosed in specific characters. We recommend that you use the e_kv_delimit function to extract values from such key-value pairs. Example:

- Raw log entry

```
content:  rats eat rice, oil|chicks eat bugs, rice|kittens eat fish, mice|
```

- Transformation rule (recommended)

  Use the e_kv_delimit function.

```
e_kv_delimit("content", pair_sep="|", kv_sep=" eat ")
```

- Result (recommended)

  The parsed log entry:

```
content:  rats eat rice, oil|chicks eat bugs, rice|kittens eat fish, mice|
kittens:  fish, mice
chicks:  bugs, rice
rats:  rice, oil
```

- Transformation rule (not recommended)

  If you use the `e_kv` function, some log fields cannot be parsed.

```
e_kv("f1", sep="eat")
```

- Result (not recommended)

  The parsed log entry:

```
content:  rats eat rice, oil|chicks eat bugs, rice|kittens eat fish, mice|
kittens:  fish
chicks:  bugs
rats:  rice
```

Transform keywords

- You can use the `e_kv` and `e_kv_delimit` functions to transform keywords and values by setting the prefix and suffix parameters in the format of `prefix="", suffix=""`.
  - Raw log entry

```
k1: q=asd&a=1&b=2
```

  - Transformation rule

```
e_kv("k1", sep="=", quote='"', prefix="start_", suffix="_end")
e_kv_delimit("k1", pair_sep=r"&", kv_sep="=", prefix="start_", suffix="_end")
e_regex("k1",r"(\w+)=([a-zA-Z0-9]+)",{r"start_\1_end": r"\2"})
```

  - Result

    Log data is transformed into keywords in the following format:

```
k1: q=asd&a=1&b=2
start_q_end: asd
start_a_end: 1
start_b_end: 2
```

- You can also use the `e_regex` function to transform the log entry. Example:
  - Transformation rule

    ```
    e_regex("k1",r"(\w+)=([a-zA-Z0-9]+)",{r"\1_\1": r"\2"})
    ```

  - Result

    Log data is transformed into keywords in the following format:

    ```
    k1: q=asd&a=1&b=2
    q_q: asd
    a_a: 1
    a_a: 2
    ```

Transform values

- Use the `e_kv` function if the log format is `k1:"v1\"abc"`, or double quotation marks exist in the log content. Example:
  - Raw log entry

    ```
    """
    In this example, the backlash (\) character is not an escape character.
    """
    content2:  k1:"v1\"abc", k2:"v2", k3: "v3"
    ```

  - Transformation rule 1

    ```
    e_kv("content2",sep=":", quote='"')
    ```

  - Result 1

    The extracted log entry:

    ```
    content2:  k1:"v1\"abc", k2:"v2", k3: "v3"
    k1: v1\
    k2: v2
    k3: v3
    ```

  - Transformation rule 2

    You can use the `e_kv` function to escape the `\` character by using the `escape` parameter. Example:

    ```
    e_kv("content2",sep=":", quote='"',escape=True)
    ```

  - Result 2

    The extracted log entry:

    ```
    content2:  k1:"v1\"abc", k2:"v2", k3: "v3"
    k1: v1"abc
    k2: v2
    k3: v3
    ```

- Use the `e_kv` function to extract key-value pairs if the log format is `a='k1=k2\';k2=k3'`. For example:

- Raw log entry

```
data: i=c10 a='k1=k2\';k2=k3'
```

- Transformation rule 1

  In the `e_kv` function, the value of the `escape` parameter is False by default.

```
e_kv("data", quote="'")
```

- Result 1

  The extracted log entry:

```
a:  k1=k2\
i:  c10
k2:  k3
```

- Transformation rule 2

  You can use the `e_kv` function to escape the `\` character by using the `escape` parameter. Example:

```
e_kv("data", quote="'", escape=True)
```

- Result 2

  The extracted log entry:

```
data: i=c10 a='k1=k2\';k2=k3'
i: c10
a: k1=k2';k2=k3
```

- Advanced transformation of key-value pairs
  - Raw log entry

```
content:  rats eat rice|chicks eat bugs|kittens eat fish|
```

  - Transformation rule

    Use the `e_regex` function:

```
e_regex("content", r"\b(\w+) eat ([^\|]+)", {r"\1": r"\2 by \1"})
```

  - Result

    The transformed log entry:

```
content:  rats eat rice|chicks eat bugs|kittens eat fish|
kittens:  fish by kittens
chicks:  bugs by chicks
rats:  rice by rats
```

Case studies

Assume that your company needs to extract the URL data from your website logs. You can customize the transformation rules based on your business requirements.

- Initial transformation
  - Requirements
    - Requirement 1: Parse the `proto`, `domain`, and `param` fields from the log entries.
    - Requirement 2: Expand the key-value pairs in the `param` field.
  - Raw log entry

```
__source__:  192.168.0.100
__tag__:__client_ip__:  192.168.0.200
__tag__:__receive_time__:  1563517113
__topic__:
request:  https://example.com/video/getlist/s?ver=3.2.3&app_type=supplier&os=Android8.1
.0
```

  - Functions
    - General orchestration

```
# Parse the request field.
e_regex('request',grok("%{URIPROTO:uri_proto}://(?:%{USER:user}(?::[^@]*)? @)?(?:%{UR
IHOST:uri_domain})?(?:%{URIPATHPARAM:uri_param})?"))
# Parse the uri_param field.
e_regex('uri_param',grok("%{GREEDYDATA:uri_path}\? %{GREEDYDATA:uri_query}"))
# Expand the key-value pairs.
e_kv("uri_query")
```

- Specific orchestration and the transformation results
  a. Use the Grok function to parse the `request` field.

     You can also use regular expressions to parse this field. For more information, see Grok function and Grok patterns.

     ```
     e_regex('request',grok("%{URIPROTO:uri_proto}://(?:%{USER:user}(?::[^@]*)? @)?(?:
     %{URIHOST:uri_domain})?(?:%{URIPATHPARAM:uri_param})?"))
     ```

     Sub-result

     ```
     uri_domain:  example.com
     uri_param:  /video/getlist/s? ver=3.2.3&app_type=supplier&os=Android8.1.0
     uri_proto:  https
     ```

  b. Use the Grok function to parse the `uri_param` field.

     ```
     e_regex('uri_param',grok("%{GREEDYDATA:uri_path}\? %{GREEDYDATA:uri_query}"))
     ```

     Sub-result

     ```
     uri_path:  /video/getlist/s
     uri_query:  ver=3.2.3&app_type=supplier&os=Android8.1.0
     ```

  c. Extract the `uri_param` field.

     ```
     e_kv("uri_query")
     ```

     Sub-result

     ```
     app_type:  supplier
     os:  Android8.1.0
     ver:  3.2.3
     ```

- Result

  Preview the transformed log entry:

  ```
  __source__:  192.168.0.100
  __tag__:__client_ip__:  192.168.0.200
  __tag__:__receive_time__:  1563517113
  __topic__:
  request:  https://example.com/video/getlist/s?ver=3.2.3&app_type=supplier&os=Android8.1.0
  uri_domain:  example.com
  uri_path:  /video/getlist/s
  uri_proto:  https
  uri_query:  ver=3.2.3&app_type=supplier&os=Android8.1.0
  app_type:  supplier
  os:  Android8.1.0
  ver:  3.2.3
  ```

  If you only need to parse the `request` field, you can use the e_kv function. For example:

  ```
  e_kv("request")
  ```

  Preview the transformed log entry:

```
__source__:  192.168.0.100
__tag__:__client_ip__:  192.168.0.200
__tag__:__receive_time__:  1563517113
__topic__:
request:  https://example.com/video/getlist/s?ver=3.2.3&app_type=supplier&os=Android8.1.0
app_type:  supplier
os:  Android8.1.0
ver:  3.2.3
```

- Advanced transformation

  If you want to extract the dynamic fields, such as the `ver`, `app_type`, and `os` fields, you can use regular expressions or the e_kv_delimit function. Example:

  - Use regular expressions.

    ```
    e_regex("url", r"\b(\w+)=([^=&]+)", {r"\1": r"\2"})
    ```

  - Use the `e_kv_delmit` function.

    ```
    e_kv_delimit("url", pair_sep=r"? &")
    ```

- Conclusion

  Most URLs can be parsed by using the preceding functions. We recommend that you use the `e_kv` function to parse URLs from raw log entries.

15.5.5. Transform logs in specific text formats

The practice cases in this topic are based on the actual data transformation requirements submitted in tickets during daily work. This topic describes how to use LOG domain specific language (DSL) orchestration to transform logs to meet the requirements.

Scenario: convert non-standard JSON objects to JSON data and spread the objects

Assume that you want to perform secondary nesting on the collected dictionary data and spread the data. You can convert the dictionary data to JSON data and then use the `e_json` function to spread the data.

- Raw log

```
content: {
  'referer': '-',
  'request': 'GET /phpMyAdmin',
  'status': 404,
  'data-1': {
    'aaa': 'Mozilla',
    'bbb': 'asde'
  },
  'data-2': {
    'up_adde': '-',
    'up_host': '-'
  }
}
```

- LOG DSL orchestration
  i. Convert data in the `content` field to the JSON format.

```
e_set("content_json",str_replace(ct_str(v("content")),"'",'"'))
```

The log after processing is as follows:

```
content: {
  'referer': '-',
  'request': 'GET /phpMyAdmin',
  'status': 404,
  'data-1': {
    'aaa': 'Mozilla',
    'bbb': 'asde'
  },
  'data-2': {
    'up_adde': '-',
    'up_host': '-'
  }
}
content_json:  {
  "referer": "-",
  "request": "GET /phpMyAdmin",
  "status": 404,
  "data-1": {
    "aaa": "Mozilla",
    "bbb": "asde"
  },
  "data-2": {
    "up_adde": "-",
    "up_host": "-"
  }
}
```

  ii. Spread the standardized `content_json` data generated after the preceding processing. For example, set `depth` in the JSON data to *1* to spread data at the first layer.

```
e_json("content_json",depth=1,fmt='full')
```

The log after spreading is as follows:

```
content_json.data-1.data-1:  {"aaa": "Mozilla", "bbb": "asde"}
content_json.data-2.data-2:  {"up_adde": "-", "up_host": "-"}
content_json.referer:  -
content_json.request:  GET /phpMyAdmin
content_json.status:  404
```

If you set `depth` to *2*, the log after spreading is as follows:

```
content_json.data-1.aaa:  Mozilla
content_json.data-1.bbb:  asde
content_json.data-2.up_adde:  -
content_json.data-2.up_host:  -
content_json.referer:  -
content_json.request:  GET /phpMyAdmin
content_json.status:  404
```

iii. To sum up, use the following LOG DSL rules:

```
e_set("content_json",str_replace(ct_str(v("content")),"'",'"'))
e_json("content_json",depth=2,fmt='full')
```

- Log after transformation

  After the log is transformed by setting `depth` to *2*, the following log is generated:

```
content:  {
  'referer': '-',
  'request': 'GET /phpMyAdmin',
  'status': 404,
  'data-1': {
    'aaa': 'Mozilla',
    'bbb': 'asde'
  },
  'data-2': {
    'up_adde': '-',
    'up_host': '-'
  }
}
content_json:  {
  "referer": "-",
  "request": "GET /phpMyAdmin",
  "status": 404,
  "data-1": {
    "aaa": "Mozilla",
    "bbb": "asde"
  },
  "data-2": {
    "up_adde": "-",
    "up_host": "-"
  }
}
content_json.data-1.aaa:  Mozilla
content_json.data-1.bbb:  asde
content_json.data-2.up_adde:  -
content_json.data-2.up_host:  -
content_json.referer:  -
content_json.request:  GET /phpMyAdmin
content_json.status:  404
```

Convert logs in other text formats to JSON and spread the data

To spread non-standard JSON data, you can flexibly combine rules.

- Raw log

```
content : {
  "pod" => {
    "name" => "crm-learning-follow-7bc48f8b6b-m6kgb"
  }, "node" => {
    "name" => "tw5"
  }, "labels" => {
    "pod-template-hash" => "7bc48f8b6b", "app" => "crm-learning-follow"
  }, "container" => {
    "name" => "crm-learning-follow"
  }, "namespace" => "testing1"
}
```

- LOG DSL orchestration
  i. Convert the log to the JSON format by using the `str_logtash_config_normalize` function.

  ```
  e_set("normalize_data",str_logtash_config_normalize(v("content")))
  ```

  ii. Use a JSON function to spread the data.

  ```
  e_json("normalize_data",depth=1,fmt='full')
  ```

  iii. To sum up, use the following LOG DSL rules:

  ```
  e_set("normalize_data",str_logtash_config_normalize(v("content")))
  e_json("normalize_data",depth=1,fmt='full')
  ```

- Log after transformation

```
content : {
  "pod" => {
    "name" => "crm-learning-follow-7bc48f8b6b-m6kgb"
  }, "node" => {
    "name" => "tw5"
  }, "labels" => {
    "pod-template-hash" => "7bc48f8b6b", "app" => "crm-learning-follow"
  }, "container" => {
    "name" => "crm-learning-follow"
  }, "namespace" => "testing1"
}
normalize_data:  {
  "pod": {
    "name": "crm-learning-follow-7bc48f8b6b-m6kgb"
  },
  "node": {
    "name": "tw5"
  },
  "labels": {
    "pod-template-hash": "7bc48f8b6b",
    "app": "crm-learning-follow"
  },
  "container": {
    "name": "crm-learning-follow"
  },
  "namespace": "testing1"
}
normalize_data.container.container:  {"name": "crm-learning-follow"}
normalize_data.labels.labels:  {"pod-template-hash": "7bc48f8b6b", "app": "crm-learning-f
ollow"}
normalize_data.namespace:  testing1
normalize_data.node.node:  {"name": "tw5"}
normalize_data.pod.pod:  {"name": "crm-learning-follow-7bc48f8b6b-m6kgb"}
```

Convert text written in special encoding formats

Hexadecimal characters that are recorded in daily work need to be decoded before they can be read. Use the `str_hex_escape_encode` function to perform the escape operation on hexadecimal characters.

- Raw log

```
content : "\xe4\xbd\xa0\xe5\xa5\xbd"
```

- LOG DSL orchestration

```
e_set("hex_encode",str_hex_escape_encode(v("content")))
```

- Log after transformation

```
content : "\xe4\xbd\xa0\xe5\xa5\xbd"
hex_encode : "Hello"
```

Spread XML fields

You may encounter various types of data during your daily work, such as XML data. To convert XML data to JSON, use the `xml_to_json` function.

- Test log

```
str : <? xmlversion="1.0"? >
<data>
    <countryname="Liechtenstein">
        <rank>1</rank>
        <year>2008</year>
        <gdppc>141100</gdppc>
        <neighborname="Austria"direction="E"/>
        <neighborname="Switzerland"direction="W"/>
    </country>
    <countryname="Singapore">
        <rank>4</rank>
        <year>2011</year>
        <gdppc>59900</gdppc>
        <neighborname="Malaysia"direction="N"/>
    </country>
    <countryname="Panama">
        <rank>68</rank>
        <year>2011</year>
        <gdppc>13600</gdppc>
        <neighborname="Costa Rica"direction="W"/>
        <neighborname="Colombia"direction="E"/>
    </country>
</data>
```

- LOG DSL orchestration

```
e_set("str_json",xml_to_json(v("str")))
```

- Log after transformation

```
str : <? xmlversion="1.0"? >
<data>
    <countryname="Liechtenstein">
        <rank>1</rank>
        <year>2008</year>
        <gdppc>141100</gdppc>
        <neighborname="Austria"direction="E"/>
        <neighborname="Switzerland"direction="W"/>
    </country>
    <countryname="Singapore">
        <rank>4</rank>
        <year>2011</year>
        <gdppc>59900</gdppc>
        <neighborname="Malaysia"direction="N"/>
    </country>
    <countryname="Panama">
        <rank>68</rank>
        <year>2011</year>
        <gdppc>13600</gdppc>
        <neighborname="Costa Rica"direction="W"/>
        <neighborname="Colombia"direction="E"/>
```

```
    </country>
</data>
str_dict :{
  "data": {
    "country": [{
      "@name": "Liechtenstein",
      "rank": "1",
      "year": "2008",
      "gdppc": "141100",
      "neighbor": [{
        "@name": "Austria",
        "@direction": "E"
      }, {
        "@name": "Switzerland",
        "@direction": "W"
      }]
    }, {
      "@name": "Singapore",
      "rank": "4",
      "year": "2011",
      "gdppc": "59900",
      "neighbor": {
        "@name": "Malaysia",
        "@direction": "N"
      }
    }, {
      "@name": "Panama",
      "rank": "68",
      "year": "2011",
      "gdppc": "13600",
      "neighbor": [{
        "@name": "Costa Rica",
        "@direction": "W"
      }, {
        "@name": "Colombia",
        "@direction": "E"
      }]
    }]
  }
}
```

15.5.6. Parse log entries in a CSV-format log file

This topic describes how to parse log entries in a CSV-format log file, for example, a syslog file.

Parse a standard log entry in a CSV-format log file

- Raw log entry

```
_program_:error
_severity_:6
_priority_:14
_facility_:1
topic:syslog-forwarder
content:198.51.100.1|10/Jun/2019:11:32:16 +0800|aliyundoc.com|GET /zf/11874.html HTTP/1.1
|200|0.077|6404|198.51.100.10:8001|200|0.060|https://example.com/s?q=%25%24%23%40%21&from
=wy878378&uc_param_str=dnntnwvepffrgibijbprsvdsei|-|Mozilla/5.0 (Linux; Android 9; HWI-AL
00 Build/HUAWEIHWI-A00) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Mobile Safari/
537.36|-|-
```

- Transformation requirements
  i. When the `_program_` field is set to *access*, a pipe-separated value (PSV) parsing is conducted to the `content` field. Then, the `content` field is dropped.
  ii. The `request: GET /css/mip-base.css HTTP/1.1` field is split into the `request_method`, `http_version`, and `request` fields.
  iii. A `URL` decoding is conducted to the `http_referer` field.
  iv. The `time` field is formatted.
- Solution
  i. If the `_program_` field is set to *access*, use the `e_psv` function to parse the `content` field, and then delete the `content` field.

```
e_if(e_search("_program_==access"), e_compose(e_psv("content", "remote_addr, time_loc
al,host,request,status,request_time,body_bytes_sent,upstream_addr,upstream_status, up
stream_response_time,http_referer,http_x_forwarded_for,http_user_agent,session_id,gui
d", restrict=True), e_drop_fields("content")))
```

The returned log entry is as follows:

```
__source__:  192.168.0.1
__tag__:__client_ip__:  192.168.0.10
__tag__:__receive_time__:  1562845168
__topic__:
_facility_:  1
_priority_:  14
_program_:  access
_severity_:  6
body_bytes_sent:  6404
guid:  -
host:  aliyundoc.com
http_referer:  https://example.com/s?q=%25%24%23%40%21&from=wy878378&uc_param_str=dnn
tnwvepffrgibijbprsvdsei
http_user_agent:  Mozilla/5.0 (Linux; Android 9; HWI-AL00 Build/HUAWEIHWI-AL00) Apple
WebKit/537.36 (KHTML, like Gecko) Version/4.0 Mobile Safari/537.36
http_x_forwarded_for:  -
remote_addr:  192.168.0.100
request:  GET /zf/11874.html HTTP/1.1
request_time:  0.077
session_id:  -
status:  200
time_local:  10/Jun/2019:11:32:16 +0800
topic:  syslog-forwarder
upstream_addr:  192.168.0.100:8001
upstream_response_time:  0.060
upstream_status:  200
```

ii. Use the `e_regex` function to parse the `request` field into the `request_method`, `request`, and `http_version` fields.

```
e_regex("request",r"^(? P<request_method>\w+) (? P<request>. +) (? P<http_version>\w+
/[\d\.] +)$")
```

The returned log entry is as follows:

```
request:  /zf/11874.html
request_method:  GET
http_version:  HTTP/1.1
```

iii. Conduct URL decoding to the `http_referer` field.

```
e_set("http",url_decoding("http_referer"))
```

The returned log entry is as follows:

```
http: https://example.com/s?q=%$#@! &from=wy878378&uc_param_str=dnntnwvepffrgibijbprs
vdsei
```

iv. Format the time.

```
e_set("time_local",dt_strptime(v("time"),"%d/%b/%Y:%H:%M:%S +0800"))
```

The returned log entry is as follows:

```
time_local:  2019-06-13 13:45:11
```

v. Run the following syntax:

```
e_if(e_search("_program_==access"), e_compose(e_psv("content", "remote_addr, time_loc
al,host,request,status,request_time,body_bytes_sent,upstream_addr,upstream_status, up
stream_response_time,http_referer,http_x_forwarded_for,http_user_agent,session_id,gui
d", restrict=True), e_drop_fields("content")))
e_regex("request",r"^(? P<request_method>\w+) (? P<request>. +) (? P<http_version>\w+
/[\d\.]+)$")
e_set("http",url_decoding("http_referer"))
e_set("time_local",dt_strptime(v("time"),"%d/%b/%Y:%H:%M:%S +0800"))
```

- Output log content

```
__source__:  192.168.0.1
__tag__:__client_ip__:  192.168.0.10
__tag__:__receive_time__:  1562840879
__topic__:
_facility_:  1
_priority_:  14
_program_:  access
_severity_:  6
body_bytes_sent:  6404
guid:  -
host:  aliyundoc.com
http_referer:  https://example.com/s?q=%E8%9B%8B%E8%8A%B1%E9%BE%99%E9%A1%BB%E9%9D%A2%E7%9
A%84%E5%81%9A%E6%B3%95&from=wy878378&uc_param_str=dnntnwvepffrgibijbprsvdsei
http_user_agent:  Mozilla/5.0 (Linux; Android 9; HWI-AL00 Build/HUAWEIHWI-AL00) AppleWebK
it/537.36 (KHTML, like Gecko) Version/4.0 Mobile Safari/537.36
http_x_forwarded_for:  -
remote_addr:  192.168.0.100
request:  GET /zf/11874.html HTTP/1.1
request_time:  0.077
session_id:  -
status:  200
time_local:  10/Jun/2019:11:32:16 +0800
topic:  syslog-forwarder
upstream_addr:  192.168.0.100:8001
upstream_response_time:  0.060
upstream_status:  200
http: https://example.com/s?q=functions&from=wy878378&uc_param_str=dnntnwvepffrgibijbprsv
dsei
```

Parse a nonstandard log entry in a CSV-format log file

The following example demonstrates how to parse a nonstandard log entry in a CSV-format log file.

- Raw log entry

```
__source__:  192.168.0.1
__tag__:__client_ip__:  192.168.0.10
__tag__:__receive_time__:  1562840879
__topic__:
content: 192.168.0.1|07/Aug/2019:11:10:37 +0800|www.learn.aliyundoc.com|GET /alyun/htsw/?
ad=5|8|6|11| HTTP/1.1|200|6.729|14559|192.168.0.1:8001|200|6.716|-|-|Mozilla/5.0 (Linux;
Android 4.1.1; Nexus 7 Build/JRO03D))||
```

- Transformation requirements

  The `content` field is parsed.

- Solution

  In the `content` field, if no correct field can be parsed from the `GET /alyun/htsw/? ad=5|8|6|11| HTTP/1.1` part by using the `e_csv` function, extract this content, and then replace it in the `content` field with null.

```
e_if(e_search("not remote_addr: *"), e_compose(e_regex("content", r"[^\|]+\|[^\|]+\|[^\|]
+\|(? P<request>(. +)HTTP/\d.\d)"), e_set("content", regex_replace(v("content"), r"([^\|]
+\|[^\|]+\|[^\|]+)\|((. +)HTTP/\d.\d)\|(.+)",replace= r"\1||\4")), e_psv("content", "remo
te_addr,time_local,host,request,status,request_time,body_bytes_sent,upstream_addr,upstrea
m_status, upstream_response_time,http_referer,http_x_forwarded_for,http_user_agent,sessio
n_id,guid", restrict=True)))
e_drop_fields("content")
```

- Output log content

```
__source__:  192.168.0.1
__tag__:__client_ip__:  192.168.0.10
__tag__:__receive_time__:  1562840879
__topic__:
body_bytes_sent:  14559
host:  www.learn.aliyundoc.com
http_referer:  -
http_user_agent:  Mozilla/5.0 (Linux; Android 4.1.1; Nexus 7 Build/JRO03D))
http_x_forwarded_for:  -
remote_addr:  192.168.0.1
request:  GET /alyun/htsw/?ad=5|8|6|11|  HTTP/1.1
request_time:  6.729
status:  200
time_local:  07/Aug/2019:11:10:37 +0800
upstream_addr:  192.168.0.1:8001
upstream_response_time:  6.716
upstream_status:  200
```

15.5.7. Transform complex JSON data

This topic describes how to use the data transformation feature of Log Service to transform complex JSON data.

Transform complex JSON data with multiple subkeys each of which is an array

Program-built logs are written in a statistical JSON format, usually containing basic information and multiple subkeys each of which is an array. For example, a server writes a log at an interval of 1 minute. The log contains the data information status and the statistical status of servers and clients generating logs.

- Sample log

```
__source__:  192.0.2.1
__topic__:
content:{
     "service": "search_service",
     "overal_status": "yellow",
     "servers": [
         {
             "host": "192.0.2.1",
             "status": "green"
         },
         {
             "host": "192.0.2.2",
             "status": "green"
         }
     ],
     "clients": [
         {
             "host": "192.0.2.3",
             "status": "green"
         },
         {
             "host": "192.0.2.4",
             "status": "red"
         }
     ]
}
```

- Data transformation requirements
  i. Split the raw log by `topic`, including `overall_type`, `client_status`, and `server_status`.
  ii. Store different information in each `topic` as follows:
    - `overall_type`: stores the server count, client count, overall status (color), and service information.
    - `client_status`: stores the host IP address, status, and service information.
    - `server_status`: stores the host IP address, status, and service information.
- Expected result

```
__source__:  192.0.2.1
__topic__:  overall_type
client_count:  2
overal_status:  yellow
server_count:  2
service:  search_service
__source__:  192.0.2.1
__topic__:  client_status
host:  192.0.2.4
status:  red
service:  search_service
__source__:  192.0.2.1
__topic__:  client_status
host:  192.0.2.3
status:  green
service:  search_service
__source__:  192.0.2.1
__topic__:  server_status
host:  192.0.2.1
status:  green
service:  search_service
__source__:  192.0.2.1
__topic__:  server_status
host:  192.0.2.2
status:  green
service:  search_service
```

- Solution
  i. Split the raw log into three logs and then further split the logs by topic. After the splitting, the three logs have the same information except for the `topic` field.

  ```
  e_set("__topic__", "server_status,client_status,overall_type")
  e_split("__topic__")
  ```

  The log after processing is as follows:

  ```
  __source__:  192.0.2.1
  __topic__:  server_status         // The topics in the other two logs are client_stat
  us and overall_type. Except for the topic field, all the other information in the thr
  ee logs is the same.
  content:  {
      ... Same as that in the raw log...
  }
  ```

  ii. Spread the JSON data in the `content` field at the first layer and delete the `content` field.

  ```
  e_json('content',depth=1)
  e_drop_fields("content")
  ```

  The log after processing is as follows:

```
__source__:  192.0.2.1
__topic__:  overall_type              // The topics in the other two logs are client_
status and server_status. Except for the topic field, all the other information in th
e three logs is the same.
clients:  [{"host": "192.0.2.3", "status": "green"}, {"host": "192.0.2.4", "status":
"red"}]
overal_status:  yellow
servers:  [{"host": "192.0.2.1", "status": "green"}, {"host": "192.0.2.2", "status":
"green"}]
service:  search_service
```

iii. For the log with the topic `overall_type`, compute the values for `client_count` and `server_count`.

```
e_if(e_search("__topic__==overall_type"),
     e_compose(
        e_set("client_count", json_select(v("clients"), "length([*])", default=0)),
        e_set("server_count", json_select(v("servers"), "length([*])", default=0))
  ))
```

The log after processing is as follows:

```
__topic__:  overall_type
server_count:  2
client_count:  2
```

iv. Delete the clients and servers fields.

```
e_if(e_search("__topic__==overall_type"), e_drop_fields("clients", "servers"))
```

v. Further split the log with the topic `server_status`.

```
e_if(e_search("__topic__==server_status"),
     e_compose(
        e_split("servers"),
        e_json("servers", depth=1)
  ))
```

The log is split into the following two logs:

```
__topic__:  server_status
servers:  {"host": "192.0.2.1", "status": "green"}
host: 192.0.2.1
status: green
```

```
__topic__:  server_status
servers:  {"host": "192.0.2.2", "status": "green"}
host: 192.0.2.2
status: green
```

vi. Delete the servers field.

```
e_if(e_search("__topic__==overall_type"), e_drop_fields("servers"))
```

vii. Further split the log with the topic `client_status` and delete the clients field.

```
e_if(e_search("__topic__==client_status"),
     e_compose(
        e_split("clients"),
        e_json("clients", depth=1),
        e_drop_fields("clients")
  ))
```

The log is split into the following two logs:

```
__topic__:  client_status
host: 192.0.2.3
status: green
```

```
__topic__:  clients
host: 192.0.2.4
status: red
```

viii. To sum up, use the following LOG domain specific language (DSL) rules:

```
# Split the raw log by topic.
e_set("__topic__", "server_status,client_status,overall_type")
e_split("__topic__")
e_json('content',depth=1)
e_drop_fields("content")
# Process the log with the topic overall_type.
e_if(e_search("__topic__==overall_type"),
     e_compose(
        e_set("client_count" json_select(v("clients"), "length([*])", default=0)),
        e_set("server_count" json_select(v("servers"), "length([*])", default=0))
  ))
# Process the log with the topic server_status.
e_if(e_search("__topic__==server_status"),
     e_compose(
        e_split("servers"),
        e_json("servers", depth=1)
  ))
e_if(e_search("__topic__==overall_type"), e_drop_fields("servers"))
# Process the log with the topic client_status.
e_if(e_search("__topic__==client_status"),
     e_compose(
        e_split("clients"),
        e_json("clients", depth=1),
        e_drop_fields("clients")
  ))
```

Solution optimization

The preceding solution does not work well if the `content.clients` or `content.servers` field is empty. Assume that the raw log is as follows:

```
__source__:  192.0.2.1
__topic__:
content:{
            "service": "search_service",
            "overal_status": "yellow",
            "servers": [ ],
            "clients": [ ]
}
```

If you split this raw log into three logs by using the preceding solution, the logs with the topics `client_status` and `server_status` are empty.

```
__source__:  192.0.2.1
__topic__:  overall_type
client_count:  0
overal_status:  yellow
server_count:  0
service:  search_service
__source__:  192.0.2.1
__topic__:  client_status
service:  search_service
__source__:  192.0.2.1
__topic__:  server_status
host:  192.0.2.1
status:  green
service:  search_service
```

- Optimized solution 1

  Check whether the logs with the topics `server_status` and `client_status` are empty after the raw log is split. If so, discard the logs.

  ```
  # Check whether the log with the topic server_status is empty. If so, discard it. If not,
  retain it.
  e_keep(op_and(e_search("__topic__==server_status"), json_select(v("servers"), "length([*]
  )")))
  # Check whether the log with the topic client_status is empty. If so, discard it. If not,
  retain it.
  e_keep(op_and(e_search("__topic__==client_status"), json_select(v("clients"), "length([*]
  )")))
  ```

  To sum up, use the following LOG DSL rules:

```
# Split the raw log by topic.
e_set("__topic__", "server_status,client_status,overall_type")
e_split("__topic__")
e_json('content',depth=1)
e_drop_fields("content")
# Process the log with the topic overall_type.
e_if(e_search("__topic__==overall_type"),
     e_compose(
        e_set("client_count" json_select(v("clients"), "length([*])", default=0)),
        e_set("server_count" json_select(v("servers"), "length([*])", default=0))
  ))
# (New) Check whether the log with the topic server_status is empty. If so, discard it. I
f not, retain it.
e_keep(op_and(e_search("__topic__==server_status"), json_select(v("servers"), "length([*]
)")))
# Process the log with the topic server_status.
e_if(e_search("__topic__==server_status"),
     e_compose(
        e_split("servers"),
        e_json("servers", depth=1)
  ))
e_if(e_search("__topic__==overall_type"), e_drop_fields("servers"))
# (New) Check whether the log with the topic client_status is empty. If so, discard it. I
f not, retain it.
e_keep(op_and(e_search("__topic__==client_status"), json_select(v("clients"), "length([*]
)")))
# Process the log with the topic client_status.
e_if(e_search("__topic__==client_status"),
     e_compose(
        e_split("clients"),
        e_json("clients", depth=1),
        e_drop_fields("clients")
  ))
```

- Optimized solution 2

  Check whether a field is empty before splitting the raw log. If the field is not empty, split the raw log based on the field.

```
# Set the initial topic.
e_set("__topic__", "server_status")
# If the content.servers field is not empty, split the raw log to obtain a log with the t
opic server_status.
e_if(json_select(v("content"), "length(servers[*])"),
   e_compse(
      e_set("__topic__", "server_status,overall_type"),
      e_split("__topic__")
   ))
# If the content.clients field is not empty, further split the raw log to obtain a log wi
th the topic client_status.
e_if(op_and(e_search("__topic__==overall_type"), json_select(v("content"), "length(client
s[*])")),
   e_compse(
      e_set("__topic__", "client_status,overall_type"),
      e_split("__topic__")
   ))
```

To sum up, use the following LOG DSL rules:

```
# Split the raw log.
e_set("__topic__", "server_status")
# If the content.servers field is not empty, split the raw log to obtain a log with the t
opic server_status.
e_if(json_select(v("content"), "length(servers[*])"),
   e_compse(
      e_set("__topic__", "server_status,overall_type"),
      e_split("__topic__")
   ))
# If the content.clients field is not empty, further split the raw log to obtain a log wi
th the topic client_status.
e_if(op_and(e_search("__topic__==overall_type"), json_select(v("content"), "length(client
s[*])")),
   e_compse(
      e_set("__topic__", "client_status,overall_type"),
      e_split("__topic__")
   ))
# Process the log with the topic overall_type.
e_if(e_search("__topic__==overall_type"),
     e_compose(
        e_set("client_count" json_select(v("clients"), "length([*])", default=0)),
        e_set("server_count" json_select(v("servers"), "length([*])", default=0))
  ))
# Process the log with the topic server_status.
e_if(e_search("__topic__==server_status"),
     e_compose(
        e_split("servers"),
        e_json("servers", depth=1)
  ))
e_if(e_search("__topic__==overall_type"), e_drop_fields("servers"))
# Process the log with the topic client_status.
e_if(e_search("__topic__==client_status"),
     e_compose(
        e_split("clients"),
        e_json("clients", depth=1),
        e_drop_fields("clients")
  ))
```

Solution comparison

- Solution 1 is redundant in logic because it deletes empty logs after obtaining them from the raw log. However, the rules are simple and easy to maintain. We recommend that you use this solution by default.
- Solution 2 has higher processing efficiency because it checks for empty fields before splitting. However, this solution uses redundant rules. We recommend that you use this solution only for specific scenarios, for example, when a large number of additional events may be produced after the raw log is split.

Transform complex JSON data with multiple layers of nested array objects

Take the following complex JSON data with multiple layers of nested arrays as an example. Assume that you want to split the logon information stored in `login_histories` of different objects in the `users` field into separate logon events.

- Raw log

```
__source__:  192.0.2.1
__topic__:
content:{
  "users": [
    {
        "name": "user1",
        "login_histories": [
          {
            "date": "2019-10-10 0:0:0",
            "login_ip": "192.0.2.6"
          },
          {
            "date": "2019-10-10 1:0:0",
            "login_ip": "192.0.2.6"
          },
      {
      ... More logon information...
      }
        ]
    },
    {
        "name": "user2",
        "login_histories": [
          {
            "date": "2019-10-11 0:0:0",
            "login_ip": "192.0.2.7"
          },
          {
            "date": "2019-10-11 1:0:0",
            "login_ip": "192.0.2.9"
          },
      {
      ... More logon information...
      }
        ]
    },
  {
    ... More users...
  }
  ]
}
```

- Expected logs after splitting

```
__source__:  192.0.2.1
name:  user1
date:  2019-10-11 1:0:0
login_ip:  192.0.2.6
__source__: 192.0.2.1
name:  user1
date:  2019-10-11 0:0:0
login_ip:  192.0.2.6
__source__:  192.0.2.1
name:  user2
date:  2019-10-11 0:0:0
login_ip:  192.0.2.7
__source__: 192.0.2.1
name:  user2
date:  2019-10-11 1:0:0
login_ip:  192.0.2.9
... More logs...
```

- Solution
  i. Split the log and spread data based on `users` in the `content` field.

  ```
  e_split("content", jmes='users[*]', output='item')
  e_json("item",depth=1)
  ```

  The log after processing is as follows:

  ```
  __source__:  192.0.2.1
  __topic__:
  content:{... Same as that in the raw log...}
  item:  {"name": "user1", "login_histories": [{"date": "2019-10-10 0:0:0", "login_ip":
  "192.0.2.6"}, {"date": "2019-10-10 1:0:0", "login_ip": "192.0.2.6"}]}
  login_histories:  [{"date": "2019-10-10 0:0:0", "login_ip": "192.0.2.6"}, {"date": "2
  019-10-10 1:0:0", "login_ip": "192.0.2.6"}]
  name:  user1
  __source__:  192.0.2.1
  __topic__:
  content:{... Same as that in the raw log...}
  item:  {"name": "user2", "login_histories": [{"date": "2019-10-11 0:0:0", "login_ip":
  "192.0.2.7"}, {"date": "2019-10-11 1:0:0", "login_ip": "192.0.2.9"}]}
  login_histories:  [{"date": "2019-10-11 0:0:0", "login_ip": "192.0.2.7"}, {"date": "2
  019-10-11 1:0:0", "login_ip": "192.0.2.9"}]
  name:  user2
  ```

  ii. Split the log and spread data based on `login_histories`.

  ```
  e_split("login_histories")
  e_json("login_histories", depth=1)
  ```

  The log after processing is as follows:

```
__source__:  192.0.2.1
__topic__:
content: {... Same as that in the raw log...}
date:  2019-10-11 0:0:0
item:  {"name": "user2", "login_histories": [{"date": "2019-10-11 0:0:0", "login_ip":
"192.0.2.7"}, {"date": "2019-10-11 1:0:0", "login_ip": "192.0.2.9"}]}
login_histories:  {"date": "2019-10-11 0:0:0", "login_ip": "192.0.2.7"}
login_ip:  192.0.2.7
name:  user2
__source__:  192.0.2.1
__topic__:
content: {... Same as that in the raw log...}
date:  2019-10-11 1:0:0
item:  {"name": "user2", "login_histories": [{"date": "2019-10-11 0:0:0", "login_ip":
"192.0.2.7"}, {"date": "2019-10-11 1:0:0", "login_ip": "192.0.2.9"}]}
login_histories:  {"date": "2019-10-11 1:0:0", "login_ip": "192.0.2.9"}
login_ip:  192.0.2.9
name:  user2
__source__: 192.0.2.1
__topic__:
content: {... Same as that in the raw log...}
date:  2019-10-10 1:0:0
item:  {"name": "user1", "login_histories": [{"date": "2019-10-10 0:0:0", "login_ip":
"192.0.2.6"}, {"date": "2019-10-10 1:0:0", "login_ip": "192.0.2.6"}]}
login_histories:  {"date": "2019-10-10 1:0:0", "login_ip": "192.0.2.6"}
login_ip:  192.0.2.6
name:  user1
__source__: 192.0.2.1
__topic__:
content: {... Same as that in the raw log...}
date:  2019-10-10 0:0:0
item:  {"name": "user1", "login_histories": [{"date": "2019-10-10 0:0:0", "login_ip":
"192.0.2.6"}, {"date": "2019-10-10 1:0:0", "login_ip": "192.0.2.6"}]}
login_histories:  {"date": "2019-10-10 0:0:0", "login_ip": "192.0.2.6"}
login_ip:  192.0.2.6
name:  user1
```

iii. Delete irrelevant fields.

```
e_drop_fields("content", "item", "login_histories")
```

The log after processing is as follows:

```
__source__: 192.0.2.1
__topic__:
name:  user1
date:  2019-10-11 1:0:0
login_ip:  192.0.2.6
__source__:  192.0.2.1
__topic__:
name:  user1
date:  2019-10-11 0:0:0
login_ip:  192.0.2.6
__source__:  192.0.2.1
__topic__:
name:  user2
date:  2019-10-11 0:0:0
login_ip:  192.0.2.7
__source__: 192.0.2.1
__topic__:
name:  user2
date:  2019-10-11 1:0:0
login_ip:  192.0.2.9
```

iv. To sum up, use the following LOG DSL rules:

```
e_split("content", jmes='users[*]', output='item')
e_json("item",depth=1)
e_split("login_histories")
e_json("login_histories", depth=1)
e_drop_fields("content", "item", "login_histories")
```

Conclusion: If you have requirements similar to the above, split the log, spread data based on specified fields, and then delete irrelevant fields.

15.5.8. Convert logs to metrics

If you want to monitor the metric change trend of a log field, you can use the e_to_metric function to convert the log field to a metric. Then, you can view the change trend of the metric in a Metricstore. This topic describes how to convert logs to metrics. NGINX access logs are used as an example.

Prerequisites

Log data is collected. For more information, see Log collection methods .

Context

The operational data of applications includes log data, trace data, and metric data. A log indicates a discrete event. A trace indicates an event that contains a call chain. A metric indicates an event that contains numeric measurements. Logs, traces, and metrics are events. A system that can store events can also store the three types of data. Log Service provides the following two types of stores for data storage:

- Logstore

  A Logstore in Log Service is a unit that is used to collect, store, and query logs. For more information, see Logstore.

- Metricstore

  A Metricstore in Log Service is a unit that is used to collect, store, and query metrics. Metricstores are optimized based on metrics. You can use the PromQL syntax that is provided by Prometheus to query data. For more information, see Metricstore.

For most applications, a log contains more information than a metric. Metrics can be considered as logs in a specific format. Log Service allows you to convert logs to metrics. You can use one of the following methods to convert logs to metrics:

- Use the e_to_metric function to convert logs to metrics.

Scenarios

For example, you have created a Logstore named nginx-demo in the China (Hangzhou) region to store NGINX access logs.

You need to monitor the changes of the request time (request_time) and response time (upstream_response_time) of each backend server (host), and then visualize the change trend on a dashboard.

```
body_bytes_sent:1750
host:www.example.com
http_referer:www.guide.example.com
http_user_agent:Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_6_6; it-it) AppleWebKit/533.20
.25 (KHTML, like Gecko) Version/5.0.4 Safari/533.20.27
http_x_forwarded_for:203.0.113.10
remote_addr:203.0.113.10
remote_user:p288
request_length:13741
request_method:GET
request_time:71
request_uri:/request/path-1/file-1
status:200
time_local:11/Aug/2021:06:52:27
upstream_response_time:0.66
```

To meet the preceding requirements, you must convert the request_time and upstream_response_time fields in logs to metrics, and then add the host label to the metrics.

Step 1: Create a Metricstore

Create a Metricstore named service-metric to store the time series data that is returned during data transformation.

1.
2.
3. On the **Time Series Storage > Metricstore** tab, click the **+** icon.
4. In the **Create Metricstore** panel, configure the following parameters and click **OK**.

Parameter	Description

Parameter	Description
Metricstore Name	The name of the Metricstore. The name must be unique in the project to which the Metricstore belongs. After the Metricstore is created, you cannot change its name.
Permanent Storage	If you turn on **Permanent Storage**, Log Service permanently stores the collected metrics. **Note** You can use an SDK to query the data retention period. If the value is 3650, the metrics are permanently stored.
Data Retention Period	The retention period of the collected metrics in the Metricstore. Valid values: 15 to 3000. Unit: days. When metrics are stored for a period that exceeds the value of this parameter, the metrics are automatically deleted. You can configure the **Data Retention Period** parameter only if you do not turn on **Permanent Storage**. **Note** If you shorten the data retention period, Log Service deletes all expired metrics within 1 hour. The data volume that is displayed for **Storage Size(Log)** on the homepage of the Log Service console is updated the next day. For example, if you change the value of the Data Retention Period parameter from 5 to 1, Log Service deletes the metrics of the previous four days within 1 hour.
Shards	The number of shards. Log Service provides shards that allow you to read and write data. Each shard supports a write speed of 5 MB/s, 500 write operations per second, a read speed of 10 MB/s, and 100 read operations per second. You can create up to 10 shards in each Metricstore. You can create up to 200 shards in each project. For more information, see Shard.

Step 2: Create a data transformation task

Use the e_to_metric function to create a data transformation task and store transformed data in the service-metric Metricstore that you created in Step 1.

1. Go to the data transformation page.
   i.
   ii.
   iii. On the Search & Analysis page, click **Data Transformation** .
2. In the upper-right corner of the page, specify a time range for the required log data.

   Make sure that log data exists on the **Raw Logs** tab.
3. In the edit box, enter a data transformation statement.

   Rename the request_time field to RequestTime, rename the upstream_response_time field to ResponseTime, and then add the host label.

```
e_to_metric(
    names=[("request_time", "RequestTime"), ("upstream_response_time", "ResponseTime")]
,
    labels=[("host", "hostname")],
)
```

For more information, see e_to_metric.

4. Click **Preview Data** .

1	target0	__labels__ :hostname#$#www.xd.mock.com __name__ :RequestTime __time_nano__ :1628672279000000000 __value__ :53.0
2	target0	__labels__ :hostname#$#www.iuj.mock.com __name__ :RequestTime __time_nano__ :1628672279000000000 __value__ :29.0
3	target0	__labels__ :hostname#$#www.hpk.mock.com __name__ :RequestTime __time_nano__ :1628672279000000000 __value__ :41.0

5. Create a data transformation task.
   i. Click **Save as Transformation Rule** .
   ii. In the **Create Data Transformation Rule** panel, set the parameters and click **OK** . The following table describes the parameters.

Parameter	Description
Rule Name	The name of the transformation rule, for example, log2mectric.
Authorization Method	Select **Default Role** to read data from the source Logstore.
Storage Target	
Target Name	The name of the storage target, for example, log2mectric.
Target Region	The region where the destination project resides, for example, China (Hangzhou).
Target Project	The name of the destination project to which transformed data is saved.
Target Logstore	The name of the destination Metricstore to which transformed data is saved, for example, service-metric.
Authorization Method	Select **Default Role** to write transformed data to the destination service-metric Metricstore.
Processing Range	
Time Range	Select **All** .

iii. In the **Result** dialog box, click **Confirm** .

After you perform the preceding steps, Log Service transforms the log data in the source Logstore and writes transformed data to the service-metric Metricstore.

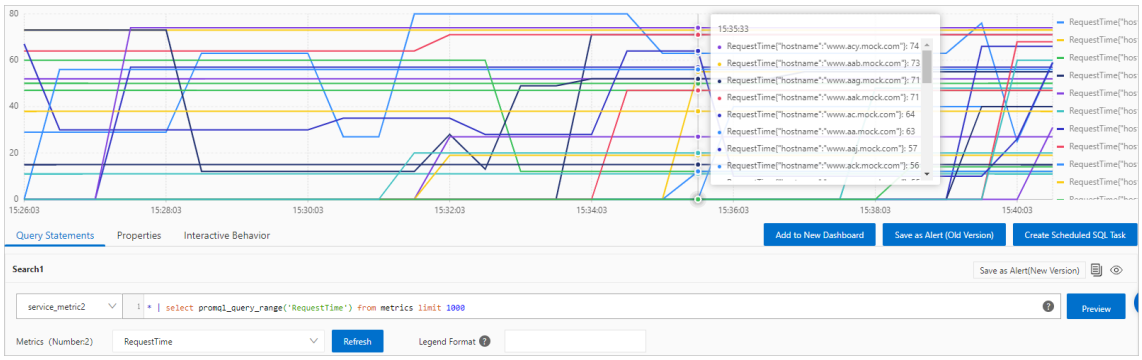
Step 3: Query time series data

1. In the left-side navigation pane, choose **Time Series Storage > Metricstore** .
2. On the Metricstore tab, select the service-metric Metricstore.
3. In the upper-right corner of the page, click **15 Minutes (Relative)** to specify a time range for the query and analysis.

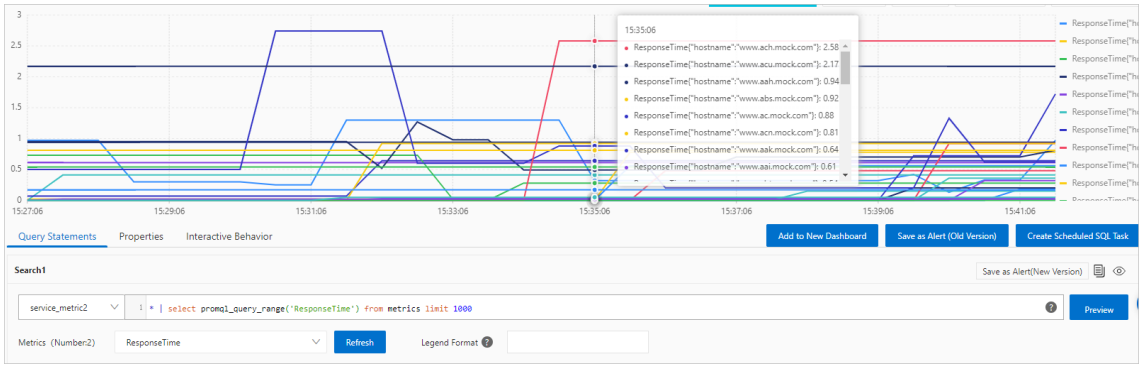
   You can select a relative time or a time frame. You can also specify a custom time range.

   > **Note** The query and analysis results may contain time series data that is generated 1 minute earlier or later than the specified time range.

4. On the **Query Statements** tab, select the RequestTime metric or the ReponseTime metric from the **Metrics** drop-down list and click **Preview** .
   - The following figure shows the change trend of the RequestTime metric for each host.



   - The following figure shows the change trend of the ReponseTime metric for each host.



15.6. Data enrichment

15.6.1. Pull data from one Logstore to enrich log data in another Logstore

This topic describes how to use resource functions to pull data from another Logstore to enrich log data in a Logstore.

Context

A hotel stores the personal information of guests in a Logstore named user_logstore and stores the check-in information of guests in a Logstore named check-in_logstore. The hotel wants to obtain some fields from the check-in_logstore Logstore and concatenate the fields with the fields in the user_logstore Logstore. The hotel can use the res_log_logstore_pull function to pull data from the check-in_logstore Logstore and use the e_table_map or e_search_dict_map function to map data. For more information about the res_log_logstore_pull function, see res_log_logstore_pull. For more information about the e_table_map function, see e_table_map. For more information about the e_search_dict_map function, see e_search_table_map.

Data transformation

- Raw data
  - The Logstore named user_logstore that is used to store personal information

```
topic:xxx
city:xxx
cid:12345
name:maki
topic:xxx
city:xxx
cid:12346
name:vicky
topic:xxx
city:xxx
cid:12347
name:mary
```

  - The Logstore named check-in_logstore that is used to store check-in information

```
time:1567038284
status:check in
cid:12345
name:maki
room_number:1111
time:1567038284
status:check in
cid:12346
name:vicky
room_number:2222
time:1567038500
status:check in
cid:12347
name:mary
room_number:3333
time:1567038500
status:leave
cid:12345
name:maki
room_number:1111
```

- Transformation rule

  > **Note** The res_log_logstore_pull function allows you to set a time range or a start time for data enrichment.
  >
  > - If you set a time range in the transformation rule, for example, **from_time=1567038284 and to_time=1567038500**, data that is received in the specified time range by Log Service is pulled for data enrichment.
  > - If you set a start time in the transformation rule, for example, **from_time="begin"**, data that is received from the specified time by Log Service is pulled for data enrichment.
  >
  > For more information about the fields of the res_log_logstore_pull function, see res_log_logstore_pull.

  - e_table_map function

    This function maps two log entries by using the cid field. If the value of the cid field in the two log entries equals each other, data mapping succeeds. The room_number field and field value are returned and concatenated with the log entry in the check-in_logstore Logstore.

    ```
    e_table_map(res_log_logstore_pull(endpoint, ak_id, ak_secret, project, logstore,
            fields=["cid","room_number"],
            from_time="begin",
            ), "cid","room_number")
    ```

  - e_search_table_map function

    This function searches for the cid field whose value is 12346 in the check-in_logstore Logstore, returns the room_number field and its value, and concatenates the field with these fields of the log entries in the user_logstore Logstore.

    ```
    e_search_table_map(res_log_logstore_pull(endpoint, ak_id, ak_secret, project, logstore,

            fields=["cid","room_number"],
            from_time="begin",
            ), "cid=12346","room_number")
    ```

- Result

- e_table_map function

```
topic:xxx
city:xxx
cid:12345
name:maki
room_nuber:1111
topic:xxx
city:xxx
cid:12346
name:vicky
room_number:2222
topic:xxx
city:xxx
cid:12347
name:mary
room_number:3333
```

- e_search_table_map function

```
topic:xxx
city:xxx
cid:12346
name:vicky
room_number:2222
```

Configure a whitelist rule and a blacklist rule to filter data

- Configure a whitelist rule
  - Transformation rule

    Use the fetch_include_data field to configure a whitelist rule. In this example, the fetch_include_data="room_number:1111" expression is included in the res_log_logstore_pull function. This expression indicates that only the log entries whose room_number field value is 1111 are pulled.

```
res_log_logstore_pull(endpoint, ak_id, ak_secret, project, logstore, ["cid","name","roo
m_number","status"],from_time=1567038284,to_time=1567038500,fetch_include_data="room_nu
mber:1111")
```

  - Retrieved data

```
status: check in
cid:12345
name:maki
room_number:1111
status:leave
cid:12345
name:maki
room_number:1111
```

- Configure a blacklist rule

- Transformation rule

  Use the fetch_exclude_data field to configure a blacklist rule. In this example, the fetch_exclude_data="room_number:1111" expression is included in the res_log_logstore_pull function. This expression indicates that only the log entries whose room_number field value is 1111 are dropped.

```
res_log_logstore_pull(endpoint, ak_id, ak_secret, project, logstore, ["cid","name","roo
m_number","status"],from_time=1567038284,to_time=1567038500,fetch_exclude_data="room_nu
mber:1111")
```

- Retrieved data

```
status:check in
cid:12346
name:vicky
room_number:2222
status:check in
cid:12347
name:mary
room_number:3333
```

- Configure a blacklist rule and a whitelist rule
  - Transformation rule

    If you configure a blacklist rule and a whitelist rule, the blacklist rule is applied first and then the whitelist rule. In this example, the fetch_exclude_data="time:1567038285",fetch_include_data="status:check in" expression is included in the res_log_logstore_pull function. This expression indicates that the log entries whose time field value is 1567038285 dropped first and then the log entries whose status field value is check in are pulled.

```
res_log_logstore_pull(endpoint, ak_id, ak_secret, project, logstore, ["cid","name","roo
m_number","status"],from_time=1567038284,to_time=1567038500,fetch_exclude_data="time:15
67038285",fetch_include_data="status:check in")
```

  - Retrieved data

```
status:check in
cid:12345
name:maki
room_number:1111
status:check in
cid:12346
name:vicky
room_number:2222
status:check in
cid:12347
name:mary
room_number:3333
```

Enable primary key maintenance to pull data from destination Logstores

You can delete pulled data before you transform other data. To do this, you can enable the primary key maintenance feature. For example, you want to pull the check-in data of customers who have checked in but not checked out from the Logstore named check-in_logstore. If a pulled log entry of a customer includes the status:leave field, the customer has checked out. You can enable the primary key maintenance feature and the log entry is not transformed.

> **Note**
>
> - You can set a single field as the value of the primary_keys parameter. The field must exist in the fields field.
> - Before you enable the primary key maintenance feature, make sure that the Logstore from which you want to pull data has only one shard.
> - If you enable the primary key maintenance feature, you cannot set the delete_data field to *None*.

- Transformation rule

```
res_log_logstore_pull(endpoint, ak_id, ak_secret, project, logstore, ["cid","name","room_number","status","time"],from_time=1567038284,to_time=None,primary_keys="cid",delete_data="status:leave")
```

- Retrieve data

  The status:leave field in the preceding log entry indicates that the customer whose name is maki has checked out. Therefore, this log entry is not transformed.

```
time:1567038284
status:check in
cid:12346
name:vicky
room_number:2222
time:1567038500
status:check in
cid:12347
name:mary
room_number:3333
```

15.6.2. Obtain the IPIP library from OSS and enrich IP address data

You can use the data transformation feature of Log Service to obtain the IPIP library from Object Storage Service (OSS) and use the library to identify the city, state, and country to which an IP address belongs.

Prerequisites

- AccessKey pairs are created to access an OSS bucket. For more information, see Create an AccessKey pair for a RAM user.

We recommend that you create an AccessKey pair that has read-only permissions on the OSS bucket and an AccessKey pair that has write-only permissions on OSS. This way, you can use the first AccessKey pair to read data from the OSS bucket and use the second AccessKey pair to write data to the OSS bucket. For more information, see Overview.

- The IPIP library is downloaded from the IPIP.NET website and uploaded to OSS. For more information, see Upload objects.

  We recommend that you use an AccessKey pair that has write-only permissions on OSS to upload the library.

Context

IPIP.NET provides a global IP address library that allows you to identity geographic locations of IP addresses around the world. You can download the IP address library from the IPIP.NET website, and upload the library to OSS. If you want to identify the city, state, and country to which an IP address belongs during data transformation, you can use the res_oss_file function to obtain the IPIP library. Then you can use the Structured data functions function to parse IP addresses and use the e_set function to add new fields to log entries.

Examples

- Raw log entry:

```
ip: 192.0.2.1
```

- Transformation rule:

```
e_set("geo",geo_parse(v("ip"), ip_db=res_oss_file(endpoint='http://oss-cn-hangzhou.aliyun
cs.com',
                                                  ak_id=res_local("AK_ID"),
                                                  ak_key=res_local("AK_KEY"),
                                                  bucket='your bucket', file='ipipfree.ipd
b',
                                                  format='binary',change_detect_interval=2
00)
```

The following table describes the fields in the res_oss_file function.

Field	Description
endpoint	The endpoint of OSS. For more information, see Regions and endpoints.
ak_id	The AccessKey ID that has read permissions on OSS. For security concerns, we recommend that you set the value to res_local("AK_ID") in the Advanced Parameter Settings field. For information about how to set the Advanced Parameter Settings field, see Create a data transformation job. 

Field	Description
ak_key	The AccessKey secret that has read permissions on OSS. For security concerns, we recommend that you set the value to res_local("AK_KEY") in the Advanced Parameter Settings field.
bucket	The OSS bucket used to store the IP address library.
file	The name of the IP address library that you have uploaded.
format	The format of the data obtained from the IP address library. Set the value to binary.

- Result:

```
ip: 192.0.2.1
city: Hangzhou
province: Zhejiang
country: China
```

15.6.3. Obtain the IP2Location library from OSS and enrich IP address data

You can use the data transformation feature of Log Service to obtain the IP2Location library from Object Storage Service (OSS) and use the library to identify the city, state, and country to which an IP address belongs.

Prerequisites

- AccessKey pairs are created to access an OSS bucket. For more information, see Create an AccessKey pair for a RAM user.

  We recommend that you create an AccessKey pair that has read-only permissions on the OSS bucket and an AccessKey pair that has write-only permissions on OSS. This way, you can use the first AccessKey pair to read data from the OSS bucket and use the second AccessKey pair to write data to the OSS bucket. For more information, see Overview.

- The IP2Location library is downloaded from the IPIP.NET website and uploaded to OSS. For more information, see Upload objects.

  We recommend that you use an AccessKey pair that has write-only permissions on OSS to upload the library.

Context

IP2Location provides a global IP address library that allows you to identity geographic locations of IP addresses around the world. You can download the IP address library from the IP2Location website and upload the library to OSS. If you want to identify the city, state, and country to which an IP address belongs during data transformation, you can use the res_oss_file function to obtain the IP2Location library from OSS. Then you can use the Structured data functions function to parse IP addresses and use the e_set function to add new fields to log entries.

Examples

- Raw log entry:

```
ip: 192.0.2.1
```

- Transformation rule:

```
e_set("geo", geo_parse(v("ip"), ip_db=res_oss_file(endpoint='http://oss-cn-hangzhou.aliyu
ncs.com',
                                                   ak_id=res_local("AK_ID"),
                                                   ak_key=res_local("AK_KEY"),
                                                   bucket='test',
                                                   file='your ip2location bin file',
                                                   format='binary', change_detect_interv
al=20),
                        provider="ip2location"))
e_json("geo")
```

The following table describes the fields in the res_oss_file function.

Field	Description
endpoint	The endpoint of OSS. For more information, see Regions and endpoints.
ak_id	The AccessKey ID that has read permissions on OSS. For security concerns, we recommend that you set the value to res_local("AK_ID") in the Advanced Parameter Settings field. For information about how to set the Advanced Parameter Settings field, see Create a data transformation job. 
ak_key	The AccessKey secret that has read permissions on OSS. For security concerns, we recommend that you set the value to res_local("AK_KEY") in the Advanced Parameter Settings field.
bucket	The OSS bucket used to store the IP address library.
file	The name of the IP address library that you have uploaded.
format	The format of the data obtained from the IP address library. Set the value to binary.

- Result:

```
ip: 192.0.2.1
city: Dearborn
province: Michigan
country: United States
```

15.6.4. Pull a CSV file from OSS to enrich data

This topic describes how to use resource functions to pull data from Object Storage Service (OSS) and use mapping functions to map the data fields to data fields in Log Service. This way, you can enrich log data in Log Service.

Prerequisites

- AccessKey pairs are created to access an OSS bucket. For more information, see Create an AccessKey pair for a RAM user.

  We recommend that you create an AccessKey pair that has read-only permissions on the OSS bucket and an AccessKey pair that has write-only permissions on OSS. This way, you can use the first AccessKey pair to read data from the OSS bucket and use the second AccessKey pair to write data to the OSS bucket. For more information, see Overview.

- A comma-separated values (CSV) file is uploaded to the OSS bucket. For more information, see Upload objects.

  We recommend that you use the AccessKey pair that has write-only permissions on the OSS bucket to upload the file.

Context

OSS provides secure, cost-effective, and highly reliable services for storing large amounts of data in the cloud. We recommend that you store infrequently updated data in OSS buckets. You are charged only for data storage. If your log data is distributed and incomplete, you can enrich your data by using OSS. The data transformation feature of Log Service provides multiple functions to transform data. You can use the res_oss_file function to pull data from OSS and use the tab_parse_csv function to create a table. Then, you can use the e_table_map function to match the specified fields, return the specified fields and field values, and generate new log entries.

Examples

- Raw log entry

  ```
  account :  Sf24asc4ladDS
  ```

- CSV data in OSS

id	account	nickname
1	Sf24asc4ladDS	Doflamingo
2	Sf24asc4ladSA	Kaido
3	Sf24asc4ladCD	Roger

- Transformation rule

Log Service maps the account field in the specified Logstore to the account field in the specified CSV file. If the value of the account field in the specified OSS bucket and Logstore equals each other, the two fields are matched and the nickname field and field value in the CSV file are concatenated with the fields in the specified Logstore.

```
e_table_map(tab_parse_csv(res_oss_file(endpoint='http://oss-cn-hangzhou.aliyuncs.com',
                                       ak_id=res_local("AK_ID"),
                                       ak_key=res_local("AK_KEY"),
                                       bucket='test',
                                       file='account.csv',change_detect_interval=30)),
            "account","nickname")
```

The following table describes the fields in the res_oss_file function.

Field	Description
endpoint	The endpoint of OSS. For more information, see Regions and endpoints.
ak_id	The AccessKey ID that has read permissions on OSS. For security concerns, we recommend that you set the value to res_local("AK_ID"). This value indicates that the AcessKey ID is obtained from the Advanced Parameter Settings field that you set in Log Service. For information about how to set the Advanced Parameter Settings field, see Create a data transformation job. 
ak_key	The AccessKey secret that has read permissions on OSS. For security concerns, we recommend that you set the value to res_local("AK_KEY"). This value indicates that the AcessKey secret is obtained from the Advanced Parameter Settings field that you set in Log Service.
bucket	The OSS bucket that is used to store the CSV file.
file	The name of the uploaded CSV file.

- Result

```
account :  Sf24asc4ladDS
nickname: Doflamingo
```

15.6.5. Pull data from an ApsaraDB RDS for MySQL database

The data transformation feature of Log Service allows you to pull data from an ApsaraDB RDS for MySQL database to enrich the data in Log Service.

Context

When you analyze data, you may need to obtain the data from separate storage resources. For example, data of user operations and user behavior is stored in Log Service, while data of user properties, registration, funds, and props is stored in an ApsaraDB RDS for MySQL database. In this case, you can use the data transformation feature to pull data from the database and store the data in a Logstore.

You can use the res_rds_mysql function to pull data from an ApsaraDB RDS for MySQL database and then use the e_table_map or e_search_table_map function to enrich data.

> **Note**
>
> - The RDS database and the Log Service project must reside in the same region. Otherwise, data cannot be obtained from the database.
> - For more information about how to use an RDS internal endpoint to access an RDS database and enrich data, see Obtain data from an ApsaraDB RDS for MySQL database over the internal network.

Enrich data by using e_table_map functions

This example describes how to use the e_table_map and res_rds_mysql functions to enrich data.

- Raw data
  - The following table shows sample data entries in an RDS table.

province	city	population	cid	eid
Shanghai	Shanghai	2000	1	00001
Tianjin	Tianjin	800	1	00002
Beijing	Beijing	4000	1	00003
Henan	Zhengzhou	3000	2	00004
Jiangsu	Nanjing	1500	2	00005

- The following content shows sample log entries in the Logstore of Log Service.

```
time:"1566379109"
data:"test-one"
cid:"1"
eid:"00001"
time:"1566379111"
data:"test_second"
cid:"1"
eid:"12345"
time:"1566379111"
data:"test_three"
cid:"2"
eid:"12345"
time:"1566379113"
data:"test_four"
cid:"2"
eid:"12345"
```

- Transformation rule

  Log Service maps the cid field in the specified Logstore to the cid field in the specified RDS table. If the value of the cid field in the specified Logstore and table equals each other, the two fields are matched and the province, city, and population fields and the field values in the RDS table are concatenated with the data fields in the specified Logstore.

  > **Note**
  > - If the value of multiple cid fields is 1, the e_table_map function maps the first data entry.
  > - The e_table_map function supports only single-row matching. To implement multi-row matching, you can use the e_search_table_map function. For more information, see Enrich data by using the e_search_map_table function.

```
e_table_map(res_rds_mysql(address="rds-host", username="mysql-username",password="xxx",da
tabase="xxx",table="xx",refresh_interval=60),"cid",["province","city","population"])
```

  For more information about how to configure an ApsaraDB RDS for MySQL database in the res_rds_mysql function, see res_rds_mysql.

- Result

```
time:"1566379109"
data:"test-one"
cid:"1"
eid:"00001"
province:"Shanghai"
city:"Shanghai"
population:"2000"
time:"1566379111"
data:"test_second"
cid:"1"
eid:"12345"
province:"Shanghai"
city:"Shanghai"
population:"2000"
time:"1566379111"
data:"test_three"
cid:"2"
eid:"12345"
province:"Henan"
city:"Zhengzhou"
population:"3000"
time:"1566379113"
data:"test_four"
cid:"2"
eid:"12345"
province:"Henan"
city:"Zhengzhou"
population:"3000"
```

Enrich data by using the e_search_map_table function

This example describes how to use the e_search_map_table and res_rds_mysql functions to enrich data.

- Raw data
  - The following table shows sample data entries in an RDS table.

content	name	age
city~=n*	aliyun	10
province~=su$	Maki	18
city:nanjing	vicky	20

  - The following content shows a sample log entry in the Logstore of Log Service.

```
time:1563436326
data:123
city:nanjing
province:jiangsu
```

- Transformation rule

A transformation rule matches log fields based on the value of the field in the specified RDS table. In this example, the field is content. The value of the field in the specified RDS table is in the key-value pair format. The key specifies the name of a log field and the value specifies the value of a log field. The value is a regular expression. After the field in the specified RDS table and a log field is matched, relevant fields and field values are concatenated with the log entry.

> **Note**
> - For more information about how to configure an ApsaraDB RDS for MySQL database in the res_rds_mysql function, see res_rds_mysql.
> - The content field is in the specified RDS table. You can use the value of this field to match a log field. You can use the regular expression matching, exact matching, or fuzzy matching method. For more information about the matching rules, see e_search.

- Single-row matching

  Matches a single field value in the specified RDS table.

```
e_search_table_map(res_rds_mysql(address="rds-host", username="mysql-username",password
="xxx",database="xxx",table="xx",refresh_interval=60),"content","name")
```

- Multi-row matching

  Matches all field values in the specified RDS table and add matched field values to the specified field.

> **Note** In multi-row matching mode, you must specify the following two parameters:
> - `multi_match=True` : enables multi-row matching.
> - `multi_join="` : multiple matched values are separated by commas (,).

```
e_search_table_map(res_rds_mysql(address="rds-host", username="mysql-username",password
="xxx",database="xxx",table="xx",refresh_interval=60),"content","name",multi_match=True
,multi_join=",")
```

- Result
  - Single-row matching

    Matches the city field whose field value matches the n* expression. If matched, returns the name field and field value from the specified RDS table and concatenates the field and field value with a log entry in Log Service.

```
time:1563436326
data:123
city:nanjing
province:jiangsu
name:aliyun
```

- Multi-row matching

  Matches the city field whose field value matches the expression n*, the province field whose field value matches the expression su$, and the city field whose field value matches the expression nanjing. A regular expression is preceded by the expression ~= . The colon (:) indicates whether the field is included. If matched, returns the name field and the three field values from the specified RDS table and concatenates the field and field value with a log entry in Log Service.

```
time:1563436326
data:123
city:nanjing
province:jiangsu
name:aliyun,Maki,vicky
```

15.6.6. Obtain data from an ApsaraDB RDS for MySQL database over the internal network

If your data is stored in Log Service and ApsaraDB RDS for MySQL, you can use the data transformation feature of Log Service to obtain data from ApsaraDB RDS for MySQL. This topic describes how to configure data transformation rules and advanced parameters to obtain data from an ApsaraDB RDS for MySQL database over the internal network.

Context

The dynamic data of shared bicycles generated in Shanghai, August 2019 is stored in a Logstore of Log Service. The data includes the order number, bicycle number, user ID, geographical location, and bicycle rider behavior. The static data such as bicycle number, brand, and batch number of the shared bicycles is stored in an ApsaraDB RDS for MySQL database. The supplier of the shared bicycles wants to analyze the static and dynamic data to enrich data and optimize bicycle scheduling.

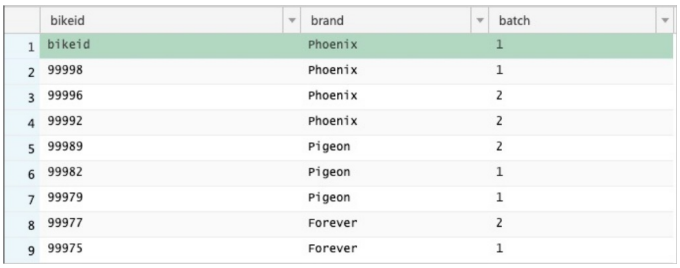
Accessing the ApsaraDB RDS for MySQL database over a public endpoint may compromise data security and system stability. Log Service provides the VPC reverse proxy feature, which allows you to access the RDS database over the internal network in a fast and secure manner.

> **Note**
>
> - The RDS database and the Log Service project must reside in the same region. Otherwise, data cannot be obtained from the database.
> - The RDS database and the Log Service project can belong to different accounts.
> - Before you access the RDS database over the internal network, you must specify a whitelist of Classless Inter-Domain Routing (CIDR) blocks for the database. In this example, the CIDR block is 100.104.0.0/16. For more information, see Control access to an ApsaraDB RDS for MySQL instance.
> - Log Service allows you to access ApsaraDB RDS for MySQL databases over the internal network. Log Service also allows you to access AnalyticDB for MySQL and PolarDB for MySQL databases over the internal network. For more information, see Access an AnalyticDB for MySQL database or PolarDB for MySQL database over the internal network.

Configurations

The following figures show how to configure data transformation rules and advanced parameters to obtain data from the ApsaraDB RDS for MySQL database over the internal network.

- Raw data
  - The following figure shows sample data records in a table of the ApsaraDB RDS for MySQL database.



	bikeid	brand	batch
1	bikeid	Phoenix	1
2	99998	Phoenix	1
3	99996	Phoenix	2
4	99992	Phoenix	2
5	99989	Pigeon	2
6	99982	Pigeon	1
7	99979	Pigeon	1
8	99977	Forever	2
9	99975	Forever	1

  - The following figure shows a sample log entry in a Logstore of Log Service.

- Data transformation procedure
  i. Enable the data transformation feature in the source Logstore.
  ii. Use the res_rds_mysql function to obtain data from the ApsaraDB RDS for MySQL database.
  iii. Save the data transformation result to the destination Logstore.
- Transformation rule:

```
e_table_map(res_rds_mysql(str_format("{}:{}",res_local("config.vpc.instance_id.test1"),res_local("config.vpc.instance_port.test1")), "your rds username", "your rds password", "your database",table="your table",primary_keys="bikeid"), "bikeid",["brand","batch"])
```

> **Note** You must use the following transformation syntax to access the ApsaraDB RDS for MySQL database over the internal network.

Transformation syntax:

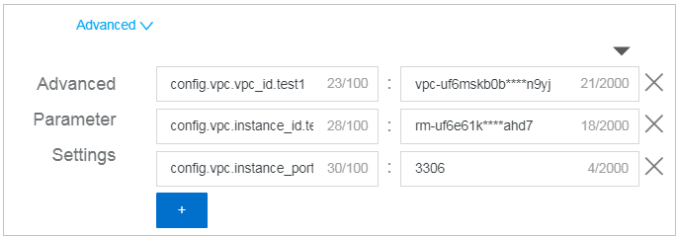
```
e_table_map(res_rds_mysql(str_format("{}:{}",res_local("config.vpc.instance_id.name"),res_local("config.vpc.instance_port.name")), "database account", "database password", "database name",table="database table name"), "field", "output_fields")
```

  - You must set the same value for the name parameter in config.vpc.instance_id.name and config.vpc.instance_port.name, and the name parameter in the **Advanced Parameter Settings** section. You can customize the value.
  - The field parameter specifies the field that you want to map. This field exists in the Logstore and ApsaraDB RDS for MySQL database. If the value of the field in the Logstore is different from the value of the field in the ApsaraDB RDS for MySQL database, the data mapping fails.

  - The output_fields parameter specifies the output fields. If the data mapping succeeds, a new log entry is generated.

- Advanced parameter settings

  You must set **Advanced Parameter Settings** when you configure the preview mode and a transformation rule. For more information about how to configure other parameters, see Create a data transformation job.



  You must set the same value for the name parameter in the config.vpc.vpc_id.name, config.vpc.instance_id.name, and config.vpc.instance_port.name parameters. The value must be the same as the name specified in the transformation rule. You can customize the value.

Key format	Sample key	Sample value	Description
config.vpc.vpc_id.name	config.vpc.vpc_id.test1	vpc-uf6mskb0b****n9yj	The vpc_id parameter specifies the ID of the VPC where the ApsaraDB RDS for MySQL database resides.
config.vpc.instance_id.name	config.vpc.instance_id.test1	rm-uf6e61k****ahd7	The instance_id parameter specifies the ID of the ApsaraDB RDS for MySQL instance.
config.vpc.instance_port.name	config.vpc.instance_port.test1	3306	The instance_port parameter specifies the internal endpoint of the ApsaraDB RDS for MySQL instance.

Troubleshooting

- A whitelist of IP addresses is not configured for the RDS instance.

  If the `reason: {"errorCode": "InvalidConfig", "errorMessage": "error when calling : res_rds_mysql\nDetail: {\"errorCode\": \"InvalidConfig\", \"errorMessage\": \"Database connection failed, cause: (2003, \\\"Can't connect to MySQL server on '192.168.1.1' (timed out)\\\")\\nDetail: None\", \"requestId\": \"\"}", "requestId": ""}` error occurs, Log Service is authorized to access the VPC where the RDS database resides. However, a whitelist of IP addresses is not configured for the RDS database. Therefore, Log Service cannot connect to the RDS database.

- Advanced parameters are invalid.

- Different names are specified in the advanced parameters.

  If the `reason: {"errorCode": "InvalidConfig", "errorMessage": "error when calling : res_rds_mysql\nDetail: {\"errorCode\": \"InvalidConfig\", \"errorMessage\": \"address check failed, please check the configuration of address. address: rm-bp***r5\\nDetail: None\", \"requestId\": \"\"}", "requestId": ""}` error occurs, the name specified in config.vpc.instance_port.name parameter is inconsistent with the name specified in the config.vpc.instance_id.name, and config.vpc.vpc_id.name parameters.

- Some advanced parameters are not specified.

  If the `reason: {"errorCode": "InvalidConfig", "errorMessage": "error when calling : res rds_mysql\nDetail: {\"errorCode\": \"InvalidConfig\", \"errorMessage\": \"address check failed, please check the configuration of address. address: rm-bp1***9r5\\nDetail: None\", \"requestId\": \"\"}", "requestId": ""}` error occurs, the config.vpc.vpc_id.name parameter is not specified.

- The syntax of the transformation rule is invalid.

  If the `reason: {"errorCode": "InvalidConfig", "errorMessage": "error when calling : res_rds_mysql\nDetail: {\"errorCode\": \"InvalidConfig\", \"errorMessage\": \"Database connection failed, cause: (2003, \\\"Can't connect to MySQL server on 'rm-bp***r5.mysql.rds.aliyuncs.com' (timed out)\\\")\\nDetail: None\", \"requestId\": \"\"}", "requestId": ""}` error occurs, the syntax of the transformation rule is invalid.

Access an AnalyticDB for MySQL database or PolarDB for MySQL database over the internal network

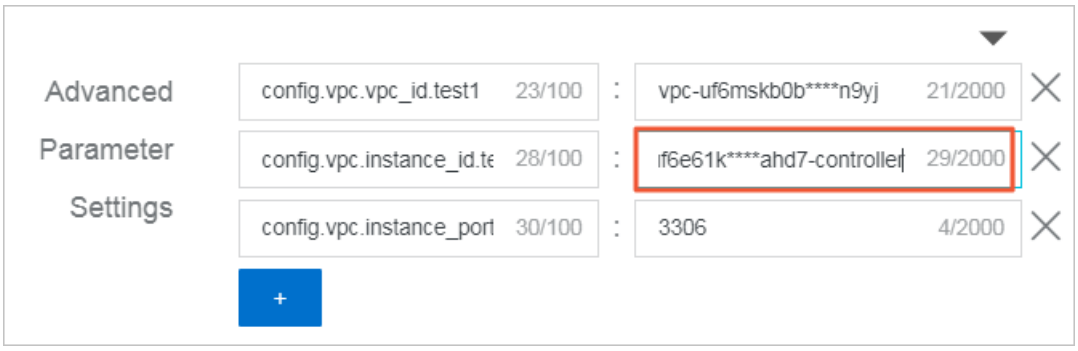
Log Service allows you to access ApsaraDB RDS for MySQL databases over the internal network. Log Service also allows you to access PolarDB for MySQL and AnalyticDB for MySQL databases over the internal network. You must complete the following configurations:

- PolarDB for MySQL databases

  Before you access a PolarDB for MySQL database over the internal network, you must specify a whitelist of Classless Inter-Domain Routing (CIDR) blocks for the database. In this example, the CIDR block is 100.104.0.0/16. For more information, see Procedure. For information about the related configurations, see Access the configurations of ApsaraDB RDS for MySQL.

- AnalyticDB for MySQL databases

Before you access an AnalyticDB for MySQL database over the internal network, you must specify a whitelist of Classless Inter-Domain Routing (CIDR) blocks for the database. In this example, the CIDR block is 100.104.0.0/16. For more information, see Configure a whitelist. For information about the related configurations, see Access the configurations of ApsaraDB RDS for MySQL. When you set the parameters in **Advanced Parameter Settings**, the value of the config.vpc.instance_id.name parameter must be in the (the name of the AnalyticDB for MySQL instance + -controller) format. The following figure shows sample settings.



15.6.7. Use resource functions to obtain incremental data

If you pull incremental data, Log Service pulls only newly added or updated data each time. This way, the data pull efficiency is improved. This topic describes how to use the res_rds_mysql function to obtain incremental data from an ApsaraDB RDS for MySQL instance.

Prerequisites

- Log Service
  - Data is uploaded to the source Logstore. For more information, see Data collection.
  - The destination Logstore is created. For more information, see Create a Logstore.
  - If you use a Resource Access Management (RAM) user, you must grant the user the permissions to transform data. For more information, see Authorize a RAM user to manage a data transformation task.
  - Indexes are configured for the source and destination Logstores. For more information, see Configure indexes.

    Data transformation is not based on indexes. However, you cannot perform query or analysis operations if you do not configure indexes.
- RDS
  - An ApsaraDB RDS database and an account that is used to connect to the database are created. For more information, see Create databases and accounts for an ApsaraDB RDS for MySQL instance.
  - Data is uploaded to the databases and tables of an ApsaraDB RDS for MySQL instance.

- A whitelist is configured for the ApsaraDB RDS for MySQL instance. For more information, see Use a database client or the CLI to connect to an ApsaraDB RDS for MySQL instance.

  > **Notice** If you use the res_rds_mysql function to pull data from an ApsaraDB RDS for MySQL database, you must create a whitelist and add *0.0.0.0* to the whitelist. This way, all IP addresses are allowed to access the database. However, this may also cause potential security risks for the database. If you want to add specific IP addresses to the whitelist, you can submit a ticket.

Context

A technology enterprise stores customer information in an ApsaraDB RDS for MySQL instance and stores customer maintenance logs in a Logstore of Log Service. The two types of data are continuously updated. The enterprise wants to perform a JOIN operation on the two types of data and save the results to a new Logstore.

In this case, you can use the res_rds_mysql function that is provided by Log Service. The function allows you to pull data from an ApsaraDB RDS for MySQL instance and save the data to a destination Logstore. If you want to pull incremental data from a database when a data transformation task is running, Log Service pulls only newly added or updated data based on the timestamp field of the database. This improves the performance of data pulls. You can use the function in a scenario that has the following characteristics: 1. A large amount of data exists in databases. 2. Data is frequently updated. 3. Data is not frequently deleted. 4. Real-time data transformation is required.

For more information about the incremental pull method and other pull methods, see res_rds_mysql.

Resources and examples

- Log Service resources
  - Project: client-project
  - Source Logstore: client-log

    Example:

  - Destination Logstore: client-information
- ApsaraDB RDS resources
  - Database: client-db

- Table: client

  Example:

c_id	name	telephone	update_time	delete_flag
1	maki	010-123	1606358931	false
2	evan	010-156	1606358931	false
3	vicky	010-166	1606358931	false

- Username and password for the database: test and test1234@@
- Public endpoint of the database: rm-bp1k****tp8o.mysql.rds.aliyuncs.com

Procedure

1.
2. Go to the data transformation page.
   i. In the Projects section, click client-project.
   ii. Choose **Log Storage > Logstores**. On the Logstores tab, click client-log.
   iii. On the Search & Analysis page, click **Data Transformation**.
3. In the upper-right corner of the page, select a time range in which you want to query log data.

   Make sure that log data exists on the **Raw Logs** tab.
4. In the edit box, enter a data transformation statement.

   For more information, see res_rds_mysql.

```
e_table_map(
    res_rds_mysql(
        "rm-bp1k****tp8o.mysql.rds.aliyuncs.com",
        "test",
        "test1234@@",
        "client-db",
        table="client",
        fields=["c_id", "name", "telephone", "update_time"],
        refresh_interval=1,
        primary_keys="c_id",
        update_time_key="update_time",
        deleted_flag_key=None,
    ),
    "c_id",
    ["name", "telephone"],
)
```

5. View data in quick preview mode.

   Check whether the transformation statement that you enter is valid in quick preview mode. For more information, see Quick preview.
   i. Click **Quick Rule Creation**.

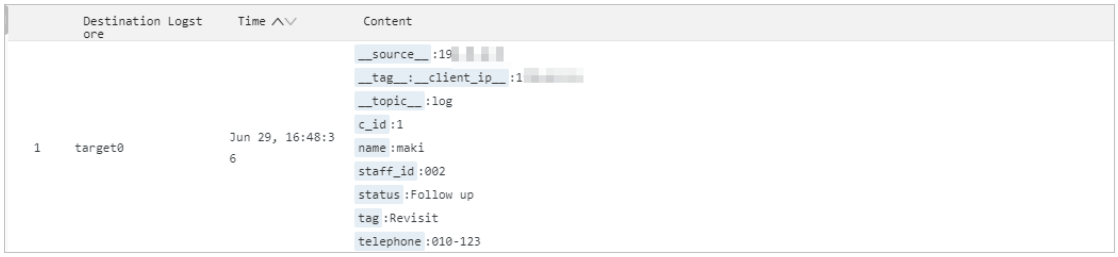
ii. Choose **Data Testing > Data**. On the Data tab, enter the following content:

```
{
  "__source__": "192.0.2.0",
  "__time__": 1624956516,
  "__topic__": "log",
  "__tag__:__client_ip__":"192.0.2.2",
  "c_id": "1",
  "staff_id": "002",
  "status": "Follow up",
  "tag": "Revisit",
}
```

iii. Choose **Data Testing > Dimension Table**. On the Dimension Table tab, enter the following content:

```
c_id,name,telephone,update_time,delete_flag
1,maki,010-123,1606358931,false
```

iv. Click **Preview Data**.
View the transformation results.



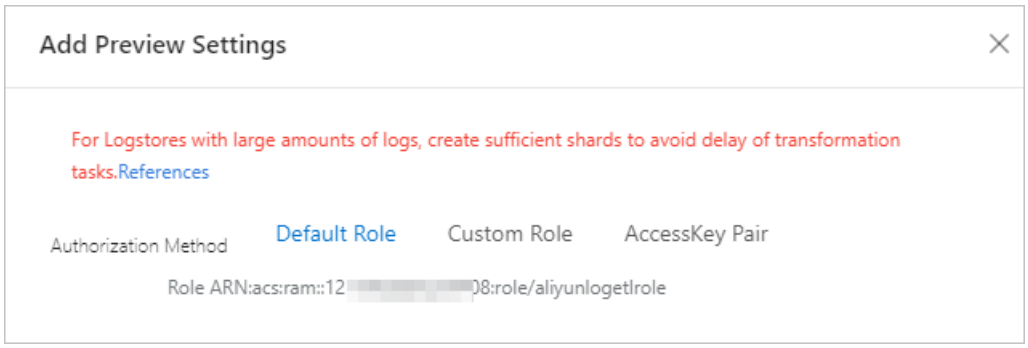
6. View data in advanced preview mode.

Check whether Log Service is connected to the ApsaraDB RDS for MySQL instance in advanced preview mode. For more information, see Advanced preview.

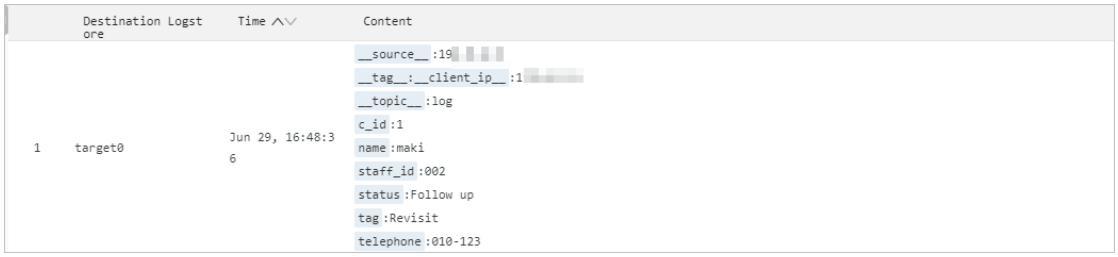
i. Click the **Advanced** tab.

ii. Click **Preview Data**.

iii. In the **Add Preview Settings** panel, set an authorization method. Then, click **OK**.



iv. View the transformation results.



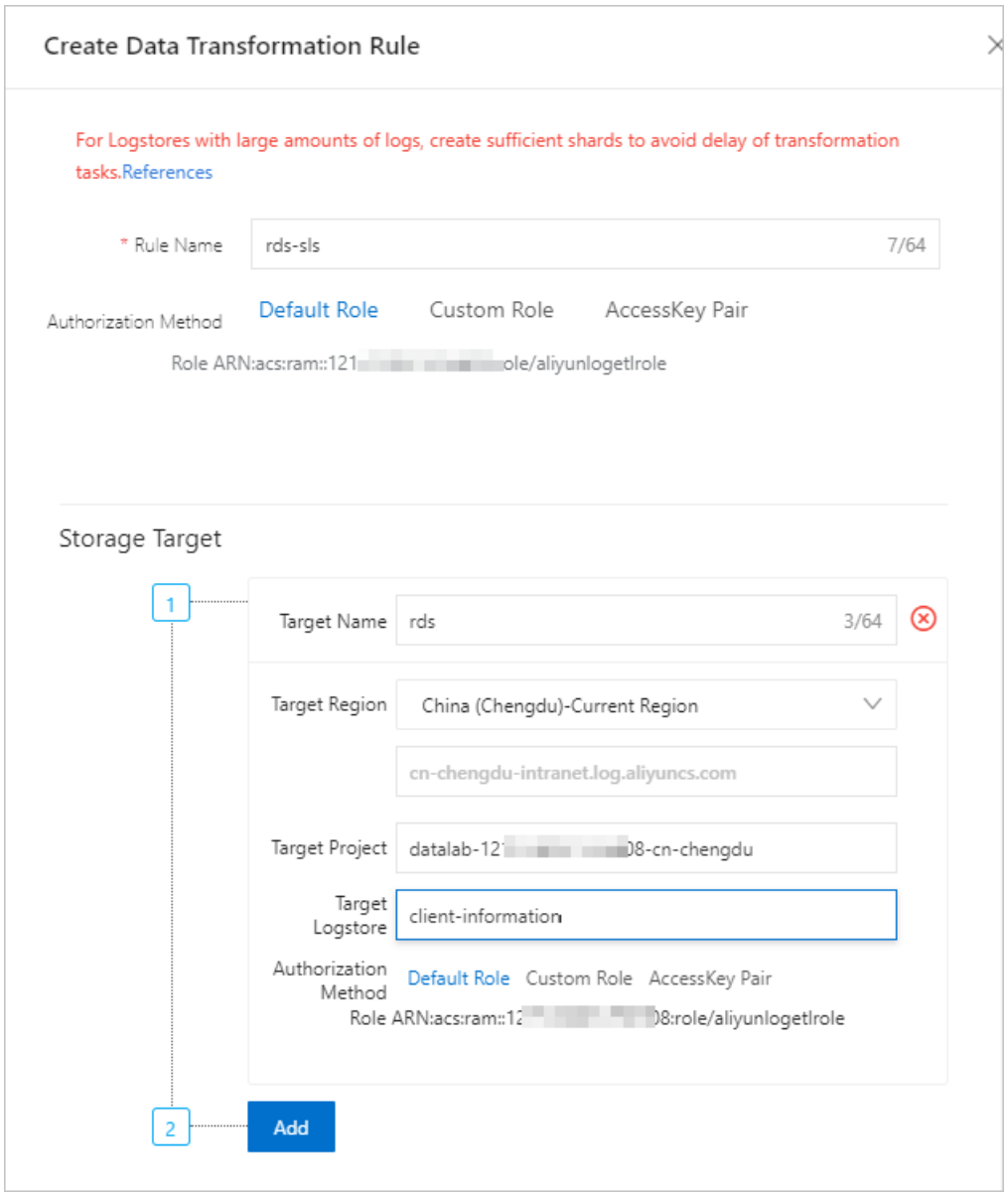
If a runtime error occurs, fix the error. For more information, see How do I fix errors in the syntax used to load data from ApsaraDB RDS for MySQL?.

7. Create a data transformation task.

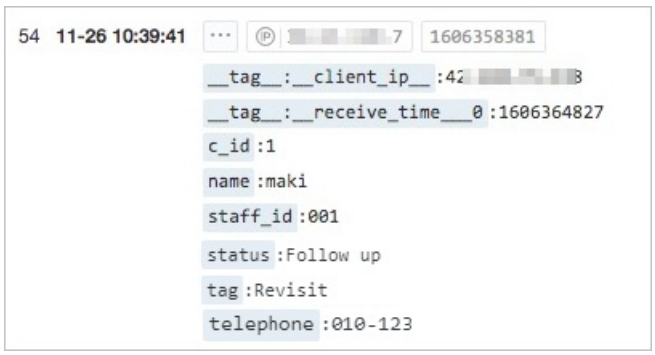
i. Click **Save as Transformation Rule**.

ii. In the **Create Data Transformation Rule** panel, configure the required parameters. Then, click **OK**.

For more information about the parameters, see Create a data transformation job.



After the data is transformed, you can view the data in the destination Logstore.



FAQ

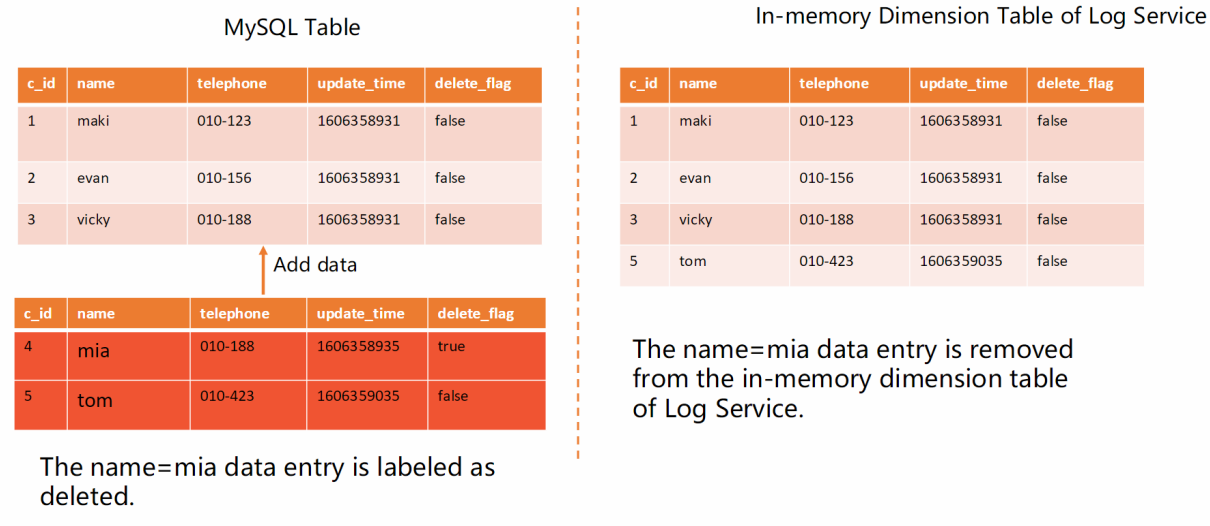
How can I use the delete feature in the incremental pull method?

In the incremental pull method, Log Service pulls only newly added or updated data based on the primary key and time field of the destination database table. If you label an entry in a database table as deleted, for example, `delete_flag=true`, Log Service still pulls and transforms the entry. To resolve this issue, Log Service adds the deleted_flag_key parameter to the res_rds_mysql function. The deleted_flag_key parameter is set in a data transformation statement. In this case, after a data transformation task obtains the updated data of a database table, the task removes the data rows whose delete_flag parameter is set to true from the in-memory dimension table. The data in the database table remains unchanged. The data rows that are removed by the data transformation task from the in-memory dimension table are excluded from the subsequent JOIN operation that is performed on the log data.

Notice

- If the delete_flag parameter is set to *true* or *1* for an entry, the entry is deleted. For more information, see res_rds_mysql.
- If the deleted_flag_key parameter is set in a data transformation statement, you must set the update_time_key parameter.

For example, if you add the name=mia and name=tom data entries to an ApsaraDB RDS for MySQL database table, the delete_flag parameter for the name=mia data entry is set to *true*. This indicates that the data entry is labeled as deleted. In this case, when Log Service updates the in-memory dimension table, the name=mia data entry is removed from the in-memory dimension table. The data entry is retained.



Example:

```
e_table_map(
    res_rds_mysql(
        "rm-bp1****l3tp.mysql.rds.aliyuncs.com",
        "test",
        "test1234@@",
        "client-db",
        table="client",
        fields=["c_id", "name", "telephone", "update_time"],
        refresh_interval=1,
        primary_keys="c_id",
        update_time_key="update_time",
        deleted_flag_key="delete_flag",
    ),
    "c_id",
    ["name", "telephone"],
)
```

15.6.8. Use the e_dict_map and e_search_dict_map functions to enrich log data

This topic describes how to enrich log data by using mapping functions such as e_dict_map and e_search_dict_map.

Context

The mapping functions in Log Service include common mapping functions and search mapping functions. This section describes the differences between these two types of functions.

- Common mapping functions map data by using the full-text matching method. Common mapping functions include the e_dict_map and e_table_map functions. The input data of the e_dict_map function is in the dictionary format. The input data of the e_table_map function is in the format of a table obtained by using resource functions. For more information about the e_dict_map function, see e_dict_map. For more information about the e_table_map function, see e_table_map. For more information about resource functions, see Resource functions.

  For example, you can use the e_dict_map function to transform HTTP status codes in NGNIX logs into data of the Text type.

HTTP status code	Text
200	Success
300	Redirect
400	Request error
500	Server error

- Search mapping functions use query strings to map fields. You can specify regular expressions or wildcard characters in query strings and use the exact match or fuzzy match method to map data.

Search mapping functions include the e_search_dict_map and e_search_table_map functions. The input data of the e_search_dict_map function is in the dictionary format. The input data of the e_search_table_map function is in the format of a table obtained by using resource functions. For more information about the e_dict_map function, see e_search_dict_map. For more information about the e_table_map function, see e_search_table_map. For more information about resource functions, see Resource functions.

For example, you can use the e_search_dict_map function to transform HTTP status codes that match the specified patterns in NGNIX logs into data of the Text type.

HTTP status code	Text
2XX	Success
3XX	Redirect
4XX	Request error
5XX	Server error

Use the e_dict_map function to enrich log data

This section describes how to use the e_dict_map function to enrich log data.

- Raw log entry

```
http_host:  example.com
http_status:  300
request_method:  GET
http_host:  example.org
http_status:  200
request_method:  POST
http_host:  example.net
http_status:  400
request_method:  GET
http_host:  aliyundoc.com
http_status:  500
request_method:  GET
```

- Transformation requirements

  Transform the status codes in the http_status field into data of the Text type and add the transformed data to the status_desc field.

- Transformation rule

```
e_dict_map({"400": "Request error", "500": "Server error", "300": "Redirect", "200": "Suc
cess"}, "status", "status_desc")
```

> **Note** The preceding transformation rule includes only four HTTP status codes. For more information, see HTTP status codes. If the value of the http_status field is *401* or *404*, the corresponding value must be included in the source dictionary. Otherwise, the data mapping will fail.

- Result

```
http_host:  example.com
http_status:  300
request_method:  GET
status_desc: Redirect
http_host:  example.org
http_status:  200
request_method:  POST
status_desc: Success
http_host:  example.net
http_status:  400
request_method:  GET
status_desc: Request error
http_host:  aliyundoc.com
http_status:  500
request_method:  GET
status_desc: Server error
```

Use the e_search_dict_map function to enrich log data

This section describes how to use the e_search_dict_map function to enrich log data.

- Raw log entry

```
http_host:  example.com
http_status:  200
request_method:  GET
body_bytes_sent: 740
http_host:  example.org
http_status:  200
request_method:  POST
body_bytes_sent: 1123
http_host:  example.net
http_status:  404
request_method:  GET
body_bytes_sent: 711
http_host:  aliyundoc.com
http_status:  504
request_method:  GET
body_bytes_sent: 1822
```

- Transformation requirements

  Add a field named type to each log entry. The value of this field is decided based on the values of the http_status and body_bytes_sent fields in each log entry.

  - If the value of the http_status field matches the *2XX* pattern and the value of the body_bytes_sent field is less than *1000* in a log entry, set the value of the type added to the log entry to *Normal*.
  - If the value of the http_status field matches the *2XX* pattern and the value of the body_bytes_sent field is equal to or greater than *1000* in a log entry, set the value of the type field added to the log entry to *Too long*.
  - If the value of the http_status field in a log entry matches the *3XX* pattern, set the value of the type field added to the log entry to *Redirect*.

- If the value of the http_status field in a log entry matches the *4XX* pattern, set the value of the type field added to the log entry to *Error*.
- If the value of the http_status field in a log entry does not match either of the preceding patterns, set the value of the type field added to the log entry to *Others*.

- Transformation rule

```
e_search_dict_map({'http_status~="2\d+" and body_bytes_sent < 1000': "Normal", 'http_stat
us~="2\d+" and body_bytes_sent >= 1000': "Too long", 'http_status~="3\d+"': "Redirect", '
http_status~="4\d+"': "Error",  "*": "Others"}, "http_status", "type")
```

If you want to use a dictionary to enrich your log data, you can create a dictionary by using braces ({}) or based on resources allocated to the task, Object Storage Service (OSS) resources, and tables. For more information, see Build dictionaries.

- Result

```
type: Normal
http_host:  example.com
http_status:  200
request_method:  GET
body_bytes_sent: 740
type: Too long
http_host:  example.org
http_status:  200
request_method:  POST
body_bytes_sent: 1123
type: Error
http_host:  example.net
http_status:  404
request_method:  GET
body_bytes_sent: 711
type: Others
http_host:  aliyundoc.com
http_status:  504
request_method:  GET
body_bytes_sent: 1822
```

15.6.9. Pull data from a Hologres database to perform data enrichment

This topic describes how to pull data from a Hologres database to perform data enrichment by using resource functions.

Prerequisites

- The endpoint, username, password, database name, and table name of the Hologres database are obtained.
- The Hologres database instance resides in the same region as the Log Service project that stores the data pulled from the Hologres database.

Context

An e-commerce platform stores sales data and customer identity information in a Hologres database. When a new user logs on to the platform, the platform recommends products to the user based on the biological sex of the user. In this scenario, the platform can pull data from the Hologres database by using the data transformation feature of Log Service. Then, the platform can store the data of the recommended products in the Logstore.

The platform can use the res_rds_mysql function to pull data from the Hologres database and use the e_search_table_map function to enrich data.

Use the e_search_table_map function to enrich log data

- Raw data
  - The following table shows a database table in Hologres.

product_id	product_name	product_price	product_sales_number	sex
2	lipstick	288	2219	girl
5	watch	1399	265	boy
6	mac	4200	265	boy
3	mouse	20	2583	boy
1	basketball	35	3658	boy
4	notebook	9	5427	girl

  - The following example shows the sample log entries in a Logstore of Log Service:

```
__source__:192.168.2.100
__tag__:__client_ip__:192.168.1.100
age:22
name:xiaoli
profession:students
sex:girl
__source__:192.168.2.200
__tag__:__client_ip__:192.168.1.200
age:21
name:xiaoming
profession:students
sex:boy
```

- Transformation rule

  Compare the sex field in the Logstore and the sex field in the Hologres database. Both fields match only when the sex values are the same. If both fields match, the value of the product_name field is pulled from the Hologres database and concatenated with the log data in the Logstore into new data.

```
e_search_table_map(res_rds_mysql(address="rds-host",
                                 username="mysql-username",password="yourpassword",databas
e="yourdatabasename",table="yourtablename",
                                 refresh_interval=60,connector='pgsql'),
                                 inpt="sex",output_fields="product_name", multi_match=True
, multi_join=",")
```

> **Note** To ensure data security and network stability, we recommend that you access the Hologres database over a virtual private cloud (VPC). You can obtain the endpoint from the network configurations of the Hologres database instance and use the endpoint to access a Hologres database over a VPC. After you obtain the endpoint, replace the value of the address field with the endpoint.

- Result

```
__source__:192.168.2.100
__tag__:__client_ip__:192.168.1.100
__tag__:__receive_time__:1615518043
__topic__:
age:22
name:xiaoli
product_name:lipstick,notebook
profession:students
sex:girl
__source__:192.168.2.200
__tag__:__client_ip__:192.168.1.200
__tag__:__receive_time__:1615518026
__topic__:
age:21
name:xiaoming
product_name:basketball,watch,mac,mouse
profession:students
sex:boy
```

15.6.10. Build dictionaries and tables for data enrichment

Dictionaries and tables are two major types of data structures used for data enrichment. This topic describes common methods for building dictionaries and tables and compares the advantages and disadvantages of different building methods.

Build dictionaries

- Build a dictionary directly

```
e_dict_map({"400": "error", "200": "ok", "*": "other"}, "status", "message")
```

- Build a dictionary from data configuration items

```
e_dict_map(res_local("http_code_map"), "status", "message")
```

where, `http_code_map` is an advanced task configuration item. Its values are as follows:

```
{"400": "error", "200": "ok", "*": "other"}
```

- Build a dictionary from a table

  Use the `tab_to_dict` function to build a dictionary from a table. Details on how to build a table are available in later part of this topic.

```
e_dict_map(tab_to_dict(tab_parse_csv("status_code,status_info\n400,error\n200,ok\n*,other
"), "status_code", "status_info"), "status", "message")
```

- Build a dictionary by using the dct_make function

```
e_dict_map(dct_make("400", "error", "200", "ok", "*", "other"), "status", "message")
```

- Build a dictionary by using another expression

```
e_dict_map(json_parse(v("http_code_map")), "status", "message")
```

  where, the data mappings are obtained from the `http_code_map` field in source logs.

The following table compares these dictionary building methods.

Method	Advantage	Disadvantage
Build a dictionary directly	This method is intuitive, simple, and easy to use.	The rules will be lengthy if too much content is involved. In addition, the built dictionary is static and inflexible.
Build a dictionary from data configuration items	We recommend that you use this method if the dictionary contains a large amount of data and is frequently modified. This method features easy maintenance.	A dictionary built in this method is not scalable or reusable across tasks. In addition, it does not support automatic update.
Build a dictionary from a table	Featuring flexible maintenance, this method is used in advanced scenarios.	You need to build and maintain a table for building a dictionary, making the configuration process complicated.
Build a dictionary by using the dct_make function	You can build a dictionary based on dynamic logic in specific scenarios.	This method is relatively advanced and difficult to maintain.
Build a dictionary by using another expression	This method dynamically extracts data mappings from fields in log events. It applies in specific scenarios.	This method is relatively advanced and difficult to maintain.

Build tables

- Build a table from a text file

```
e_table_map(tab_parse_csv("city,name,age\nshanghai,aliyun,10\ncity:nanjing,Maki,18"), "na
me",["city", "age"])
```

- Build a table from data configuration items

```
e_search_table_map(tab_parse_csv(res_local("table_info")), "name",["city", "age"])
```

where, `table_info` is a task configuration item in data transformation rules. Its values are as follows:

```
content,name,age
shanghai,aliyun,10
nanjing,Maki,18
```

- Build a table from an ApsaraDB for Relational Database Service (RDS) database

```
e_table_map(tab_parse_csv(res_rds_mysql(...database="db", table="city")), "name",["city",
"age"])
```

The `city` table in the RDS database contains the following information:

```
content,name,age
shanghai,aliyun,10
nanjing,Maki,18
```

- Build a table from another Logstore

```
e_table_map(res_log_logstore_pull(..., project="project_name", logstore="logstore_name",
fields=["city","name","age"]),, "name",["city", "age"])
```

The specified Logstore contains the following log events:

```
"Log 1"
{
  "city": "shanghai",
  "name": "aliyun",
  "age": "10"
}
"Log 2"
{
  "city": "city:nanjing and data > 100",
  "name": "Maki",
  "age": "18"
}
```

The following table compares these table building methods.

Method	Advantage	Disadvantage
Build a table from a text file	This method is intuitive, simple, and easy to use.	The rules will be lengthy if too much content is involved. A table built in this method is difficult to maintain, scale up, or reuse.

Method	Advantage	Disadvantage
Build a table from data configuration items	We recommend that you use this method if the table contains a large amount of data and is frequently modified. This method features easy maintenance.	A table built in this method is not scalable or reusable across tasks. In addition, it does not support automatic update.
Build a table from an ApsaraDB for RDS database	• We recommend that you use this method if the table contains a large amount of data and is frequently modified. This method features easy maintenance. • This method supports automatic update. • A table built in this method can be reused across tasks.	You need to connect to an external ApsaraDB for RDS database for building a table, making the configuration process complicated.
Build a table from another Logstore	This method reads data in real time. Featuring flexible maintenance, this method is used in advanced scenarios.	You need to connect to another Logstore for building a table, making the configuration process complicated.

15.6.11. Use the e_table_map function to enrich HTTP response status codes

NGINX logs include important information that can be used for website O&M. Log Service provides the e_table_map function that you can use to enrich HTTP response status codes for NGINX log analysis. This topic describes how to use the data transformation feature of Log Service to enrich HTTP response status codes.

Prerequisites

NGINX logs are collected. For more information, see Data collection overview.

Scenarios

For example, you have developed Application A and defined HTTP response status codes to maintain the application. The data volume is updated at irregular intervals. However, only the http_code field in raw NGINX logs indicates the status of HTTP requests. This field cannot help you identify issues in an efficient manner.

To meet the preceding requirements, you must use the e_table_map function to enrich the log field based on a mapping table of HTTP response status codes. Then, you can identify the status of HTTP requests in an efficient manner.

- Raw log

```
body_bytes_sent:1750
host:www.example.com
http_referer:www.example.aliyundoc.com
http_user_agent:Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_6_6; it-it) AppleWebKit/533.
20.25 (KHTML, like Gecko) Version/5.0.4 Safari/533.20.27
http_x_forwarded_for:203.0.103.10
remote_addr:203.0.103.10
remote_user:p288
request_length:13741
request_method:GET
request_time:71
request_uri:/request/path-1/file-1
http_code:200
time_local:11/Aug/2021:06:52:27
upstream_response_time:0.66
```

This raw log is stored in a Logstore named nginx-demo. The value of the http_code field is an HTTP response status code.

- Mapping table of HTTP response status codes

  The following table is a typical mapping table of HTTP response status codes.

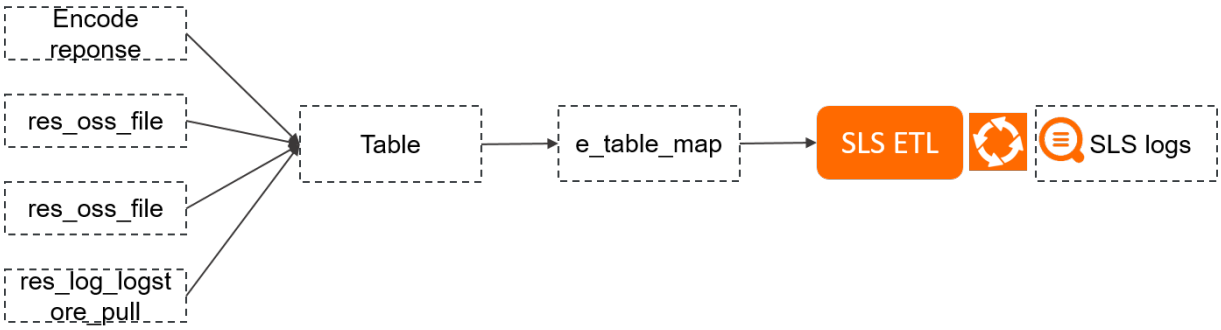
code	alias	category	desc
100	1xx	Informational	Continue
200	2xx	Success	OK
300	3xx	Redirection	Multiple Choices
400	4xx	Client Error	Bad Request

- Enriched log

```
body_bytes_sent:1750
host:www.example.com
http_code:200
http_code_alias:2xx
http_code_category:Success
http_code_desc:OK
http_referer:www.example.aliyundoc.com
http_user_agent:Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_6_6; it-it) AppleWebKit/533.
20.25 (KHTML, like Gecko) Version/5.0.4 Safari/533.20.27
http_x_forwarded_for:203.0.103.10
remote_addr:203.0.103.10
remote_user:p288
request_length:13741
request_method:GET
request_time:71
request_uri:/request/path-1/file-1
time_local:11/Aug/2021:06:52:27
upstream_response_time:0.66
```

Solutions

Transformation process



1. Convert an HTTP response status code to a table object.
2. Use the e_table_map function to transform and enrich data.

Solutions

The following table describes the solutions that you can use to enrich data.

Solution	Supported data volume	Incremental update	Batch update	Scenario
Use a Logstore to enrich data (recommended)	Large	Supported	Supported	A mapping table that contains a large volume of data and is frequently updated
Use a MySQL table to enrich data	Medium	Not supported	Supported	A mapping table that is frequently updated
Use an Object Storage Service (OSS) object to enrich data	Medium	Not supported	Supported	A mapping table that is infrequently updated
Encode responses to enrich data	Small	Not supported	Not supported	A simple mapping table of HTTP response status codes

Solution 1: Use a Logstore to enrich data (recommended)

1. Use an SDK to write HTTP response status codes to a Logstore named http_code.

   The following example shows a log that contains an HTTP response status code in the http_code Logstore:

```
__source__:203.0.103.10
__tag__:__receive_time__:1595424194
__topic__:
code:200
alias:2xx
description:OK
category:Success
```

For more information, see Log Service SDK overview.

2. Obtain the name, endpoint, and AccessKey pair of the Logstore that stores HTTP response status codes. The obtained information is used to write a data transformation statement.

   For more information about Log Service endpoints and AccessKey pairs, see Endpoints and AccessKey pair.

3. Go to the data transformation page of the nginx-demo Logstore that stores raw logs.

   For more information, see Create a data transformation job.

4. In the edit box, enter a data transformation statement.

   Read data from the http_code Logstore that stores HTTP response status codes and use the e_table_map function to return the values of the matched fields.

```
e_table_map( res_log_logstore_pull("cn-hangzhou-intranet.log.aliyuncs.com",
        res_local("AK_ID"),res_local("AK_KEY"),"live-demo","http_code",
        ["code","alias","description","category"]),
              [("http_code","code")],
              [("alias","http_code_alias"), ("description","http_code_desc"),
              ("category","http_code_category")])
```

   Notice To ensure data security, we recommend that you specify an AccessKey pair in the Advanced Parameter Settings field. For more information about how to set the advanced parameters, see Create a data transformation job.

   - The res_log_logstore_pull function is used to pull data from another Logstore when you transform data in a Logstore. For more information, see res_log_logstore_pull.
   - The e_table_map function is used to map the value of a specified field to a row in a table, and then return the value of the field in this row. For more information, see e_table_map.

5. Click **Preview Data**.

   After the raw NGINX log is enriched, new fields that contain the information of the HTTP response status code are added to the log.

```
body_bytes_sent:1750
host:www.example.com
http_code:200
http_code_alias:2xx
http_code_category:Success
http_code_desc:OK
http_referer:www.example.aliyundoc.com
http_user_agent:Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_6_6; it-it) AppleWebKit/53
3.20.25 (KHTML, like Gecko) Version/5.0.4 Safari/533.20.27
http_x_forwarded_for:203.0.103.10
remote_addr:203.0.103.10
remote_user:p288
request_length:13741
request_method:GET
request_time:71
request_uri:/request/path-1/file-1
time_local:11/Aug/2021:06:52:27
upstream_response_time:0.66
```

6. Create a data transformation task.

   For more information, see Create a data transformation job.

Solution 2: Use a MySQL table to enrich data

1. Save HTTP response status codes to an ApsaraDB RDS for MySQL database.

   The following figure shows the mapping table of HTTP response status codes in the ApsaraDB RDS for MySQL database.

```
MySQL [etl-test]> select * from http_code limit 10;
+------+-------+---------------+--------------------------------+
| code | alias | category      | description                    |
+------+-------+---------------+--------------------------------+
| 100  | 1xx   | Informational | Continue                       |
| 101  | 1xx   | Informational | Switching Protocols            |
| 102  | 1xx   | Informational | Processing (WebDAV)            |
| 200  | 2xx   | Success       | OK                             |
| 201  | 2xx   | Success       | Created                        |
| 202  | 2xx   | Success       | Accepted                       |
| 203  | 2xx   | Success       | Non-Authoritative Information  |
| 204  | 2xx   | Success       | No Content                     |
| 205  | 2xx   | Success       | Reset Content                  |
| 206  | 2xx   | Success       | Partial Content                |
+------+-------+---------------+--------------------------------+
10 rows in set (0.04 sec)
```

2. Obtain the host address, username, password, and table of the ApsaraDB RDS for MySQL database. The obtained information is used to write a data transformation statement.
3. Go to the data transformation page of the nginx-demo Logstore that stores raw logs.

   For more information, see Create a data transformation job.

4. In the edit box, enter a data transformation statement.

   Read data from the MySQL database and use the e_table_map function to return the values of the matched fields.

```
e_table_map(res_rds_mysql(address="MySQL host address",
                  username="username", password="password",
                  database="database",table="table name", refresh_interval=300),
              [("http_code","code")],
              [("alias","http_code_alias"), ("description","http_code_desc"),
              ("category","http_code_category")])
```

   Notice To ensure data security, we recommend that you specify an AccessKey pair in the Advanced Parameter Settings field. For more information about how to set the advanced parameters, see Create a data transformation job.

   - The res_rds_mysql function is used to pull data from a specified table in an ApsaraDB RDS for MySQL database. For more information, see res_rds_mysql.
   - The e_table_map function is used to map the value of a specified field to a row in a table, and then return the value of the field in this row. For more information, see e_table_map.

5. Click **Preview Data**.

   After the raw NGINX log is enriched, new fields that contain the information of the HTTP response status code are added to the log.

```
body_bytes_sent:1750
host:www.example.com
http_code:200
http_code_alias:2xx
http_code_category:Success
http_code_desc:OK
http_referer:www.example.aliyundoc.com
http_user_agent:Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_6_6; it-it) AppleWebKit/53
3.20.25 (KHTML, like Gecko) Version/5.0.4 Safari/533.20.27
http_x_forwarded_for:203.0.103.10
remote_addr:203.0.103.10
remote_user:p288
request_length:13741
request_method:GET
request_time:71
request_uri:/request/path-1/file-1
time_local:11/Aug/2021:06:52:27
upstream_response_time:0.66
```

6. Create a data transformation task.

   For more information, see Create a data transformation job.

Solution 3: Use an OSS object to enrich data

1. Save HTTP response status codes to an object named *http_code.csv* and upload the object to an OSS bucket.

   For more information, see Upload objects.

2. Obtain the name, endpoint, and the AccessKey pair of the OSS bucket to which the *http_code.csv* object is saved. The obtained information is used to write a data transformation statement.

   For more information about OSS endpoints, see Regions and endpoints.

3. Go to the data transformation page of the nginx-demo Logstore that stores raw logs.

   For more information, see Create a data transformation job.

4. In the edit box, enter a data transformation statement.

   Read data from the OSS bucket and use the e_table_map function to return the values of the matched fields.

```
e_table_map(
      tab_parse_csv(
           res_oss_file(endpoint="oss-cn-shanghai-internal.aliyuncs.com",
              ak_id=res_local("AK_ID"), ak_key=res_local("AK_KEY"),
              bucket="ali-sls-etl-test",
              file="http_code.csv", format='text')),
              [("http_code","code")],
              [("alias","http_code_alias"),
               ("description","http_code_desc"),
               ("category","http_code_category")])
```

   > **Notice** To ensure data security, we recommend that you specify an AccessKey pair in the Advanced Parameter Settings field. For more information about how to set the advanced parameters, see Create a data transformation job.

   - The res_oss_file function is used to obtain the object content of a specified bucket from OSS. The object can be refreshed at regular intervals. For more information, see res_oss_file.
   - The tab_parse_csv function is used to construct a table from a comma-separated values (CSV) file. For more information, see tab_parse_csv.
   - The e_table_map function is used to map the value of a specified field to a row in a table, and then return the value of the field in this row. For more information, see e_table_map.

5. Click **Preview Data**.

   After the raw NGINX log is enriched, new fields that contain the information of the HTTP response status code are added to the log.

```
body_bytes_sent:1750
host:www.example.com
http_code:200
http_code_alias:2xx
http_code_category:Success
http_code_desc:OK
http_referer:www.example.aliyundoc.com
http_user_agent:Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_6_6; it-it) AppleWebKit/53
3.20.25 (KHTML, like Gecko) Version/5.0.4 Safari/533.20.27
http_x_forwarded_for:203.0.103.10
remote_addr:203.0.103.10
remote_user:p288
request_length:13741
request_method:GET
request_time:71
request_uri:/request/path-1/file-1
time_local:11/Aug/2021:06:52:27
upstream_response_time:0.66
```

6. Create a data transformation task.

   For more information, see Create a data transformation job.

Solution 4: Encode responses to enrich data

1. Prepare a mapping table of HTTP response status codes in the CSV format.
2. Go to the data transformation page of the nginx-demo Logstore that stores raw logs.

   For more information, see Create a data transformation job.
3. In the edit box, enter a data transformation statement.

   Use the tab_parse_csv function to convert the HTTP response status codes in the CSV format and use the e_table_map function to return the values of the matched fields.

```
e_table_map(tab_parse_csv("code,alias,category,description\n100,1xx,Informational,Conti
nue\n101,1xx,Informational,Switching Protocols\n102,1xx,Informational,Processing (WebDA
V)\n200,2xx,Success,OK\n201,2xx,Success,Created\n202,2xx,Success,Accepted\n203,2xx,Succ
ess,Non-Authoritative Information\n204,2xx,Success,No Content\n205,2xx,Success,Reset Co
ntent\n206,2xx,Success,Partial Content\n207,2xx,Success,Multi-Status (WebDAV)\n208,2xx,
Success,Already Reported (WebDAV)\n226,2xx,Success,IM Used\n300,3xx,Redirection,Multipl
e Choices\n301,3xx,Redirection,Moved Permanently\n302,3xx,Redirection,Found\n303,3xx,Re
direction,See Other\n304,3xx,Redirection,Not Modified\n305,3xx,Redirection,Use Proxy\n3
06,3xx,Redirection,(Unused)\n307,3xx,Redirection,Temporary Redirect\n308,3xx,Redirectio
n,Permanent Redirect (experimental)\n400,4xx,Client Error,Bad Request\n401,4xx,Client E
rror,Unauthorized\n402,4xx,Client Error,Payment Required\n403,4xx,Client Error,Forbidde
n\n404,4xx,Client Error,Not Found\n405,4xx,Client Error,Method Not Allowed\n406,4xx,Cli
ent Error,Not Acceptable\n407,4xx,Client Error,Proxy Authentication Required\n408,4xx,C
lient Error,Request Timeout\n409,4xx,Client Error,Conflict\n410,4xx,Client Error,Gone\n
411,4xx,Client Error,Length Required\n412,4xx,Client Error,Precondition Failed\n413,4xx
,Client Error,Request Entity Too Large\n414,4xx,Client Error,Request-URI Too Long\n415,
4xx,Client Error,Unsupported Media Type\n416,4xx,Client Error,Requested Range Not Satis
fiable\n417,4xx,Client Error,Expectation Failed\n418,4xx,Client Error,I'm a teapot (RFC
2324)\n420,4xx,Client Error,Enhance Your Calm (Twitter)\n422,4xx,Client Error,Unprocess
able Entity (WebDAV)\n423,4xx,Client Error,Locked (WebDAV)\n424,4xx,Client Error,Failed
Dependency (WebDAV)\n425,4xx,Client Error,Reserved for WebDAV\n426,4xx,Client Error,Upg
rade Required\n428,4xx,Client Error,Precondition Required\n429,4xx,Client Error,Too Man
y Requests\n431,4xx,Client Error,Request Header Fields Too Large\n444,4xx,Client Error,
No Response (Nginx)\n449,4xx,Client Error,Retry With (Microsoft)\n450,4xx,Client Error,
Blocked by Windows Parental Controls (Microsoft)\n451,4xx,Client Error,Unavailable For
Legal Reasons\n499,4xx,Client Error,Client Closed Request (Nginx)\n500,5xx,Server Error
,Internal Server Error\n501,5xx,Server Error,Not Implemented\n502,5xx,Server Error,Bad
Gateway\n503,5xx,Server Error,Service Unavailable\n504,5xx,Server Error,Gateway Timeout
\n505,5xx,Server Error,HTTP Version Not Supported\n506,5xx,Server Error,Variant Also Ne
gotiates (Experimental)\n507,5xx,Server Error,Insufficient Storage (WebDAV)\n508,5xx,Se
rver Error,Loop Detected (WebDAV)\n509,5xx,Server Error,Bandwidth Limit Exceeded (Apach
e)\n510,5xx,Server Error,Not Extended\n511,5xx,Server Error,Network Authentication Requ
ired\n598,5xx,Server Error,Network read timeout error\n599,5xx,Server Error,Network con
nect timeout error\n"),
              [("http_code","code")],
              [("alias","http_code_alias"), ("description","http_code_desc"),
              ("category","http_code_category")])
```

Notice To ensure data security, we recommend that you specify an AccessKey pair in the Advanced Parameter Settings field. For more information about how to set the advanced parameters, see Create a data transformation job.

- The tab_parse_csv function is used to construct a table from a CSV file. For more information, see tab_parse_csv.
- The e_table_map function is used to map the value of a specified field to a row in a table, and then return the value of the field in this row. For more information, see e_table_map.

4. Click **Preview Data**.

   After the raw NGINX log is enriched, new fields that contain the information of the HTTP response status code are added to the log.

```
body_bytes_sent:1750
host:www.example.com
http_code:200
http_code_alias:2xx
http_code_category:Success
http_code_desc:OK
http_referer:www.example.aliyundoc.com
http_user_agent:Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_6_6; it-it) AppleWebKit/53
3.20.25 (KHTML, like Gecko) Version/5.0.4 Safari/533.20.27
http_x_forwarded_for:203.0.103.10
remote_addr:203.0.103.10
remote_user:p288
request_length:13741
request_method:GET
request_time:71
request_uri:/request/path-1/file-1
time_local:11/Aug/2021:06:52:27
upstream_response_time:0.66
```

5. Create a data transformation task.

   For more information, see Create a data transformation job.

15.7. Data forwarding

15.7.1. Replicate data from a Logstore

Log Service allows you to replicate data from a source Logstore to a destination Logstore. To replicate the data, you can create a data transformation rule for the source Logstore. This topic describes how to replicate data from a source Logstore to a destination Logstore in a typical scenario.

Scenario

In this example, the access logs of a company are collected and stored in multiple projects of an Alibaba Cloud account. The company wants to replicate the access logs from the logstore-a Logstore of the project-a project to the logstore-b Logstore of the project-b project. Then, the company can query and analyze the data in the project-b project in a centralized manner. To replicate data from the logstore-a Logstore to the logstore-b Logstore, the company can use the replication feature in Log Service.

To use the replication feature, you must make sure that the following requirements are met:

- A Logstore named logstore-a is created and data is written to the Logstore.
- The performance of the logstore-b Logstore is evaluated, for example, the number of shards. For more information, see Performance guide.
- A Logstore named logstore-b is created. For more information, see Manage a Logstore.

Procedure

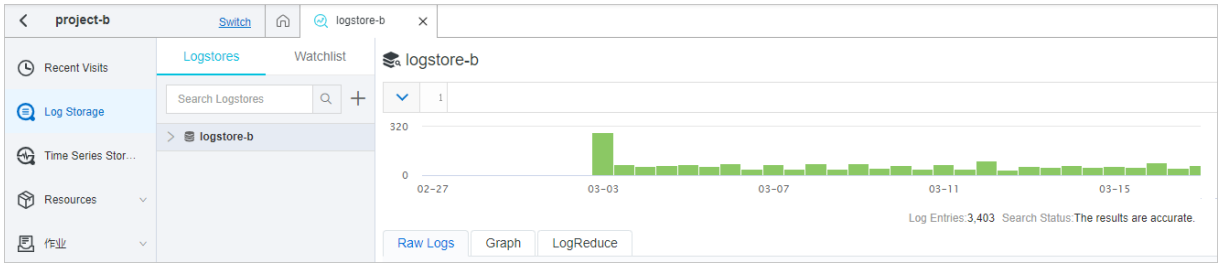
1.
2. In the **Projects** section, click the project-a project.
3. On the **Log Storage > Logstores** tab, click the logstore-a Logstore.
4. On the Search & Analysis page, click **Data Transformation** in the upper-right corner to enable the data transformation mode.
5. On the page that appears, click **Save as Transformation Rule**.
6. In the **Create Data Transformation Rule** panel, set the required parameters. The following table describes the parameters.

Parameter	Description
Rule Name	The name of the transformation rule. Enter test.
Authorization Method	Authorizes Log Service to read data from the logstore-a Logstore. The default role is used as an example. Select Default Role.
Target Name	The name of the storage target. Enter test.
Target Region	The region where the destination project resides. Select China (Hangzhou).
Target Project	The name of the destination project. Enter project-b.
Target Logstore	The name of the destination Logstore. Enter logstore-b.
Authorization Method	Authorizes Log Service to read data from and write data to the logstore-b Logstore. The default role is used as an example. Select Default Role.
Time Range	The time when the logs are received. The All value indicates that the data in the source Logstore is transformed from the first log entry until the transformation task is manually stopped. Select All.

   For information about other parameters, see Create a data transformation job.
7. Click **OK**.

Result

In the Projects section, click the project-b project. On the **Log Storage > Logstores** tab, select the logstore-b Logstore. You can view the data that is replicated from the logstore-a Logstore.

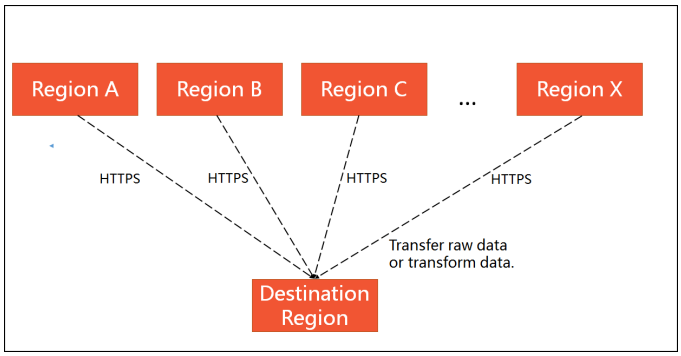


15.7.2. Transfer data across regions

Log Service allows you to transfer data across regions by using the data transformation feature. This topic describes the scenarios, procedure, and billing of data transmission across regions.

Scenarios

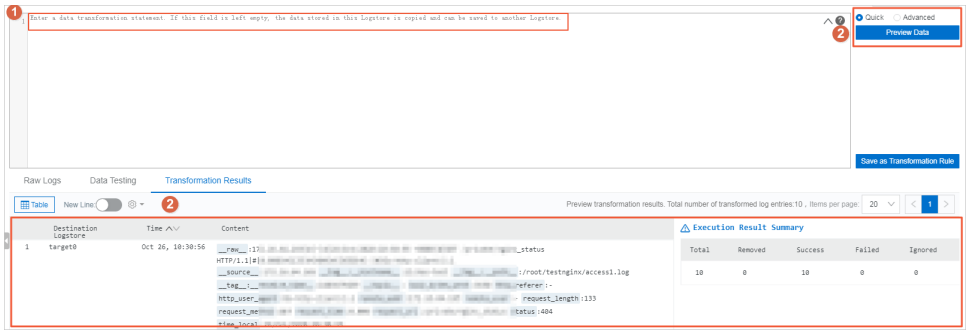
If your cloud services are deployed in different regions, the service-related logs are distributed across these regions. If you want to collect these logs, you can use the data transformation feature of Log Service.



Procedure

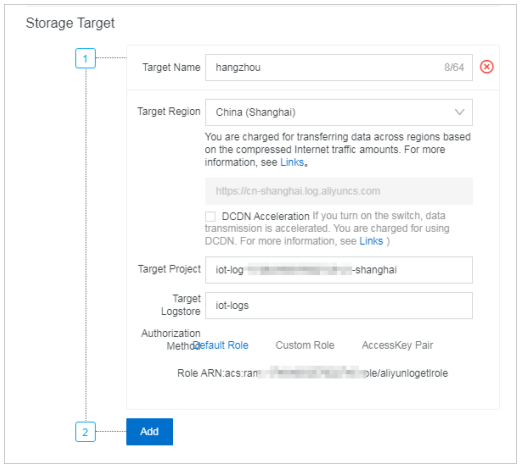
1. Enter a transformation rule on the data transformation page and click **Preview Data** to view the result. For more information, see Create a data transformation job.

   > **Note** You do not need to enter any transformation rule if you want to transfer raw logs.



2. Click **Save as Transformation Rule** if no error occurs.
3. In the **Create Data Transformation Rule** dialog box, set the **Target Region** parameter.

   For information about other parameters, see Create a data transformation rule.



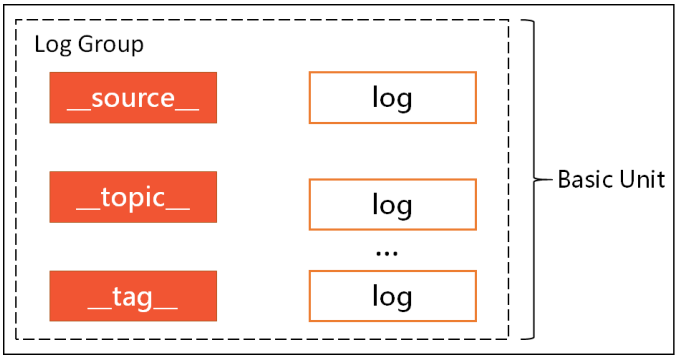
FAQ

- How am I charged for data transmission across regions?

  You are charged based on the size of compressed data during data transmission. For example, assume that the compression ratio is 10:1 and you need to transfer 10 MB of raw data. The size of the data after compression is 1 MB. This way, only 1 MB of data incurs fees. For more information about the billing methods, see Billable items.

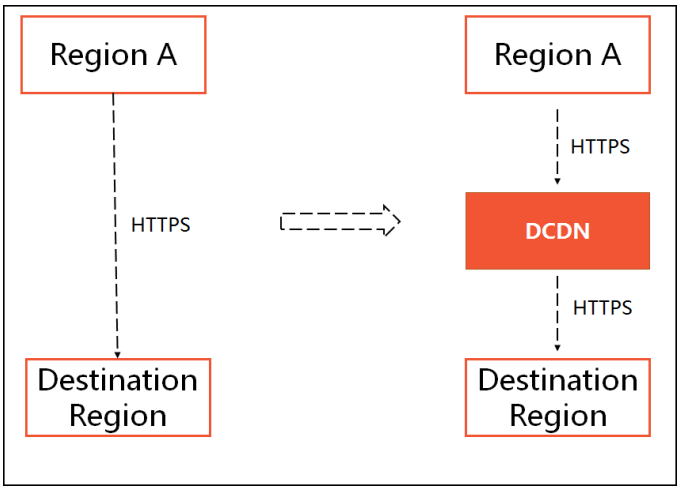
- How do I increase the data compression ratio?

  For example, to increase the data compression ratio in the web tracking scenario, you can set fixed values for the __topic__, __tag__, and __source__ fields in different log entries to transfer them as a log group. This is because log data is compressed by log group. If you do not set fixed values for the __source__, __topic__, and __tag__ fields, multiple log groups may be generated and the data compression ratio will be reduced.



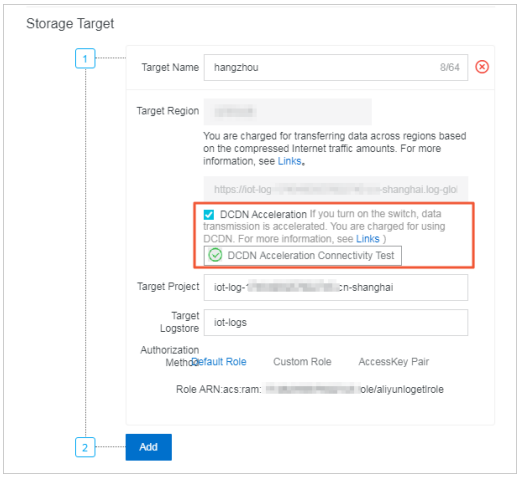
- What can I do if the network is unstable?
  - If the network is unstable when you transfer data across borders, you do not need to perform operations. The data transmission task automatically retries if the network is unstable.

- You can activate Dynamic Route for CDN (DCDN) to improve network stability.



a. For more information about how to activate DCDN, see Enable the global acceleration feature.

b. On the **Create Data Transformation Rule** page, select **DCDN Acceleration** and click **DCDN Acceleration Connectivity Test**.



15.7.3. Distribute data to multiple destination Logstores

This topic describes how to distribute data to multiple destination Logstores in various scenarios. These scenarios include dynamic distribution, cross-account distribution, dynamic cross-account distribution, and multi-source dynamic distribution.

Context

The data transformation feature of Log Service allows you to distribute data transformation results to multiple destination Logstores. When you save data transformation results, you can configure AccessKey pairs of different accounts. This way, you can distribute data transformation results to the Logstores of these accounts. You can also use the e_output or e_coutput function to dynamically set destination projects and Logstores, and distribute data transformation results to these Logstores.

> **Note**
>
> - In cross-account distribution scenarios, you can distribute data transformation results to a maximum of 20 Logstores that belong to different accounts.
> - You can use the e_output function to send log data to specified Logstores. In this case, after log data is sent to the specified Logstores, the subsequent transformation rules do not take effect on the log data. If you want to use the subsequent transformation rules to transform the log data, you can replace the e_output function with the e_coutput function. For more information, see e_output and e_coutput. In the following scenarios, the e_output function is used to describe how to distribute log data.

Scenario 1: cross-account distribution

For example, assume that all access log entries of your website are stored in a Logstore. You want to distribute the log entries to Logstores of different accounts based on the value of the http_status field in the log entries.

In this scenario, you can use the data transformation feature to distribute the log entries.

- Raw log entries:

```
http_host:  example.com
http_status:  200
request_method:  GET
request_uri:  /pic/icon.jpg
scheme:  https
http_host:  example.org
http_status:  301
request_method:  POST
request_uri:  /data/data.php
scheme:  http
http_host:  example.net
http_status:  404
request_method:  GET
request_uri:  /category/abc/product_id
scheme:  https
http_host:  aliyundoc.com
http_status:  504
request_method:  GET
request_uri:  /data/index.html
scheme:  https
```

- Distribution requirements
  - Store the log entries whose http_status field value is 2XX as target0 in Logstore0 and set the topic of the log entries to success_event.
  - Store the log entries whose http_status field value is 3XX as target1 in Logstore1 and set the topic of the log entries to redirection_event.
  - Store the log entries whose http_status field value is 4XX as target2 in Logstore2 and set the topic of the log entries to unauthorized_event.
  - Store the log entries whose http_status field value is 5XX as target3 in Logstore3 and set the topic of the log entries to internal_server_error_event.

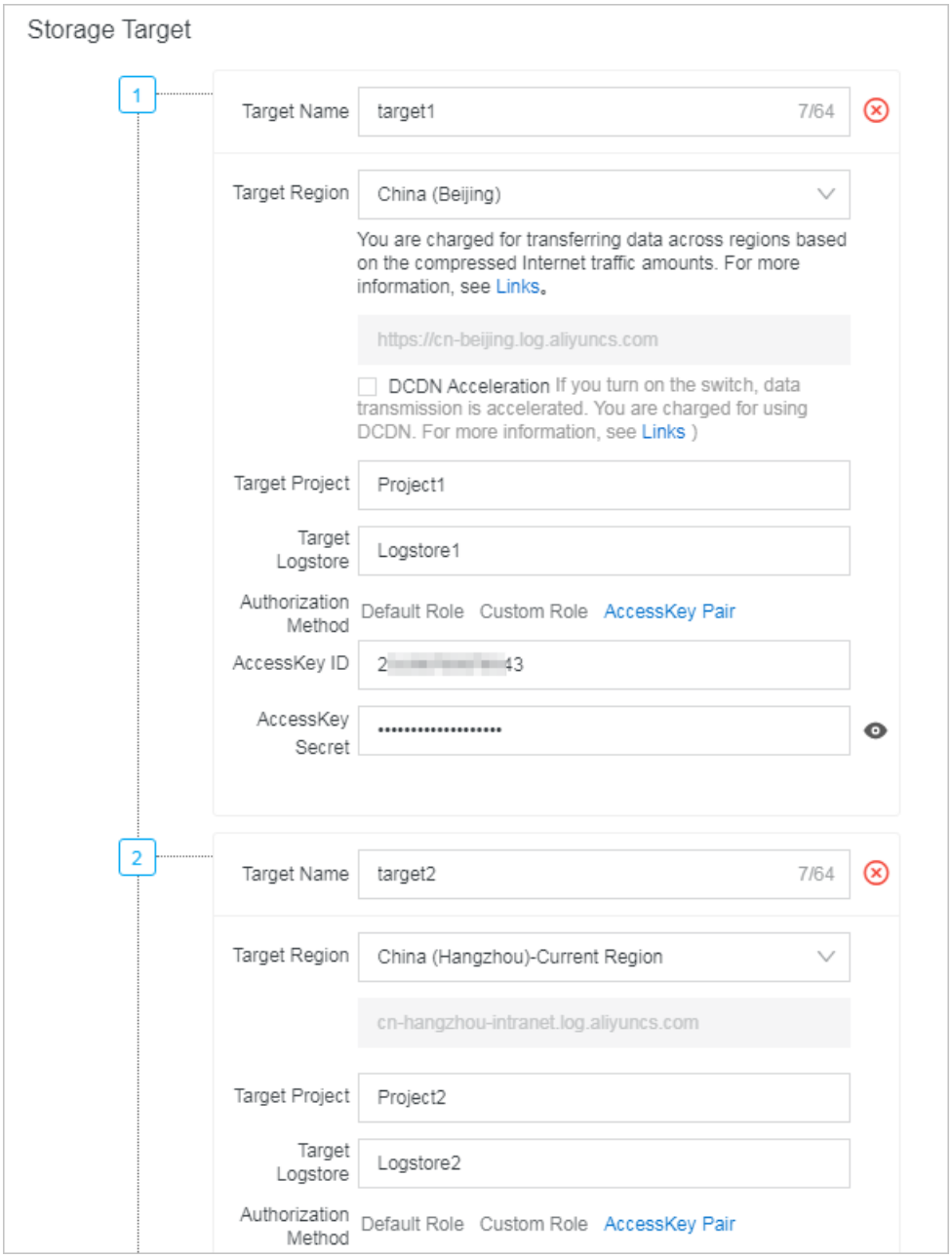
The log entries that are stored as target0 belong to Account A and the log entries that are stored as target1, target2, and target3 belong to Account B.

- Transformation rule:

```
e_switch(e_match("status", r"2\d+"), e_set("__topic__", "success_event"),
         e_match("status", r"3\d+"), e_compose(e_set("__topic__", "redirection_event"), e
_output("target1")),
         e_match("status", r"4\d+"), e_compose(e_set("__topic__", "unauthorized_event"),
e_output("target2")),
         e_match("status", r"5\d+"), e_compose(e_set("__topic__", "internal_server_error_
event`"), e_output("target3"))
    )
```

- Storage target

  On the **Create Data Transformation Rule** page, configure storage targets. For information about the parameters of storage targets, see Create a data transformation job.



Label	Storage target	Destination project and Logstore	AccessKey
1	target0	Project0 and Logstore0	The AccessKey pair of Account A
2	target1	Project1 and Logstore1	The AccessKey pair of Account B
3	target2	Project2 and Logstore2	The AccessKey pair of Account B
4	target3	Project3 and Logstore3	The AccessKey pair of Account B

- Result:

```
## The log entries whose http_status field value is 2XX are distributed to Logstore0 of A
ccount A.
__topic__:  success_event
http_host:  example.com
http_status:  200
request_method:  GET
request_uri:  /pic/icon.jpg
scheme:  https
## The log entries whose http_status field value is 3XX are distributed to Logstore1 of A
ccount B.
__topic__:  redirection_event
http_host:  example.org
http_status:  301
request_method:  POST
request_uri:  /data/data.php
scheme:  http
## The log entries whose http_status field value is 4XX are distributed to Logstore2 of A
ccount B.
__topic__: unauthorized_event
http_host:  example.net
http_status:  404
request_method:  GET
request_uri:  /category/abc/product_id
scheme:  https
## The log entries whose http_status field value is 5XX are distributed to Logstore3 of A
ccount B.
__topic__: internal_server_error_event
http_host:  aliyundoc.com
http_status:  504
request_method:  GET
request_uri:  /data/index.html
scheme:  https
```

Scenario 2: dynamic distribution

For example, assume that all access log entries of your website are stored in a Logstore. You want to distribute the log entries to different Logstores based on the project and logstore fields in the log entries.

In this scenario, you can use the data transformation feature to distribute the log entries.

- Raw log entries:

```
__tag__:type: dynamic_dispatch
host:  example.aliyundoc.com
project: Project1
logstore: Logstore1
http_status:  200
request_method:  GET
request_uri:  /pic/icon.jpg
scheme:  https
__tag__:type: dynamic_dispatch
host:  demo.aliyundoc.com
project: Project1
logstore: Logstore2
http_status:  301
request_method:  POST
request_uri:  /data/data.php
scheme:  http
__tag__:type:  dynamic_dispatch
host:   learn.aliyundoc.com
project: Project2
logstore: Logstore1
http_status:  404
request_method:  GET
request_uri:  /category/abc/product_id
scheme:  https
__tag__:type: dynamic_dispatch
host:   guide.aliyundoc.com
project: Project2
logstore: Logstore2
http_status:  504
request_method:  GET
request_uri:  /data/index.html
scheme:  https
```

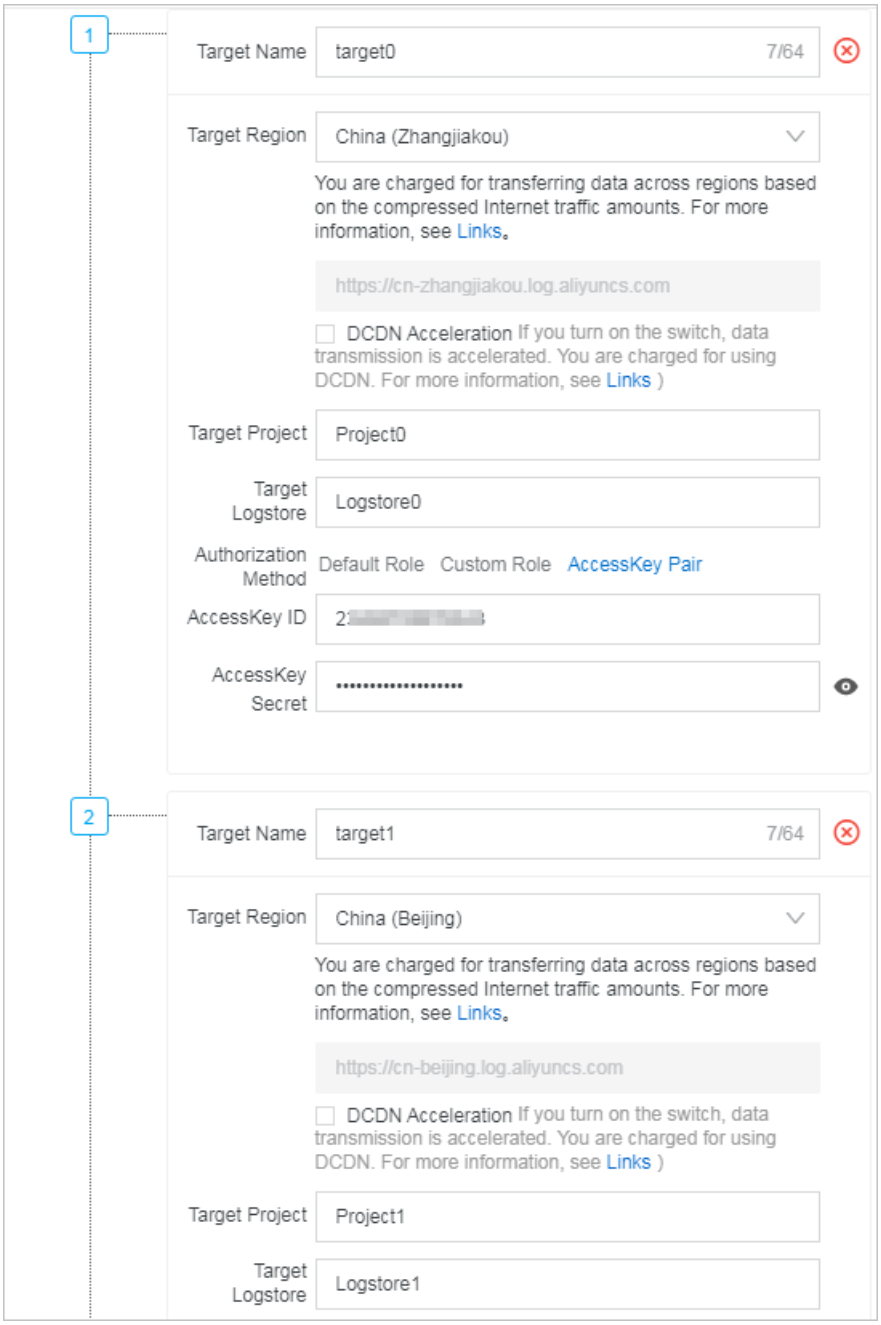
- Distribution requirements
  - Distribute log entries based on the values of the project and logstore fields.
  - Add the __tag:__type field to each log entry and set the value of the field to dynamic_dispatch.
- Transformation rule:

```
e_output(project=v("project"), logstore=v("logstore"), tags={"type": "dynamic_dispatch"})
```

  The e_output function extracts the values of the project and logstore fields from each log entry and distributes each log entry based on the values.
- Storage target

  On the **Create Data Transformation Rule** page, configure storage targets.

  > **Note** In this scenario, the destination projects and Logstores are determined based on the settings of the project and logstore parameters in the e_output function. These projects and Logstores are irrelevant to the destination project and Logstore that you configure to store the default storage target (labeled 1) on the **Create Data Transformation Rule** page.


1
Target Name
target0
7/64
Target Region
China (Zhangjiakou)
You are charged for transferring data across regions based on the compressed Internet traffic amounts. For more information, see Links.
https://cn-zhangjiakou.log.aliyuncs.com
DCDN Acceleration If you turn on the switch, data transmission is accelerated. You are charged for using DCDN. For more information, see Links)
Target Project
Project0
Target Logstore
Logstore0
Authorization Method
Default Role Custom Role AccessKey Pair
AccessKey ID
AccessKey Secret
2
Target Name
target1
7/64
Target Region
China (Beijing)
You are charged for transferring data across regions based on the compressed Internet traffic amounts. For more information, see Links.
https://cn-beijing.log.aliyuncs.com
DCDN Acceleration If you turn on the switch, data transmission is accelerated. You are charged for using DCDN. For more information, see Links)
Target Project
Project1
Target Logstore
Logstore1

- Result:

```
## Log entry that is distributed to Logstore1 of Project1.
__tag__:type: dynamic_dispatch
host:  example.aliyundoc.com
project: Project1
logstore: Logstore1
http_status:  200
request_method:  GET
request_uri:  /pic/icon.jpg
scheme:  https
## Log entry that is distributed to Logstore2 of Project1.
__tag__:type: dynamic_dispatch
host:  demo.aliyundoc.com
project: Project1
logstore: Logstore2
http_status:  301
request_method:  POST
request_uri:  /data/data.php
scheme:  http
## Log entry that is distributed to Logstore1 of Project2.
__tag__:type:  dynamic_dispatch
host:   learn.aliyundoc.com
project: Project2
logstore: Logstore1
http_status:  404
request_method:  GET
request_uri:  /category/abc/product_id
scheme:  https
## Log entry that is distributed to Logstore2 of Project2.
__tag__:type: dynamic_dispatch
host:   guide.aliyundoc.com
project: Project2
logstore: Logstore2
http_status:  504
request_method:  GET
request_uri:  /data/index.html
scheme:  https
```

Scenario 3: Dynamic cross-account distribution

For example, assume that all access log entries of your website are stored in a Logstore. You want to distribute the log entries to Logstores of different accounts based on the project and logstore fields in the log entries.

In this scenario, you can use the data transformation feature to distribute the log entries.

- Raw log entries:

```
host:  example.aliyundoc.com
project: Project1
logstore: Logstore1
http_status:  200
request_method:  GET
request_uri:  /pic/icon.jpg
scheme:  https
host:  demo.aliyundoc.com
project: Project1
logstore: Logstore2
http_status:  301
request_method:  POST
request_uri:  /data/data.php
scheme:  http
host:   learn.aliyundoc.com
project: Project2
logstore: Logstore1
http_status:  404
request_method:  GET
request_uri:  /category/abc/product_id
scheme:  https
host:   guide.aliyundoc.com
project: Project2
logstore: Logstore2
http_status:  504
request_method:  GET
request_uri:  /data/index.html
scheme:  https
```

- Distribution requirements

  Distribute log entries to projects and Logstores of different accounts based on the values of the project and logstore fields. The destination projects include Project1 and Project2. Project1 includes two Logstores named Logstore1 and Logstore2 and belongs to Account A. Project2 includes two Logstores named Logstore1 and Logstore2 and belongs to Account B.

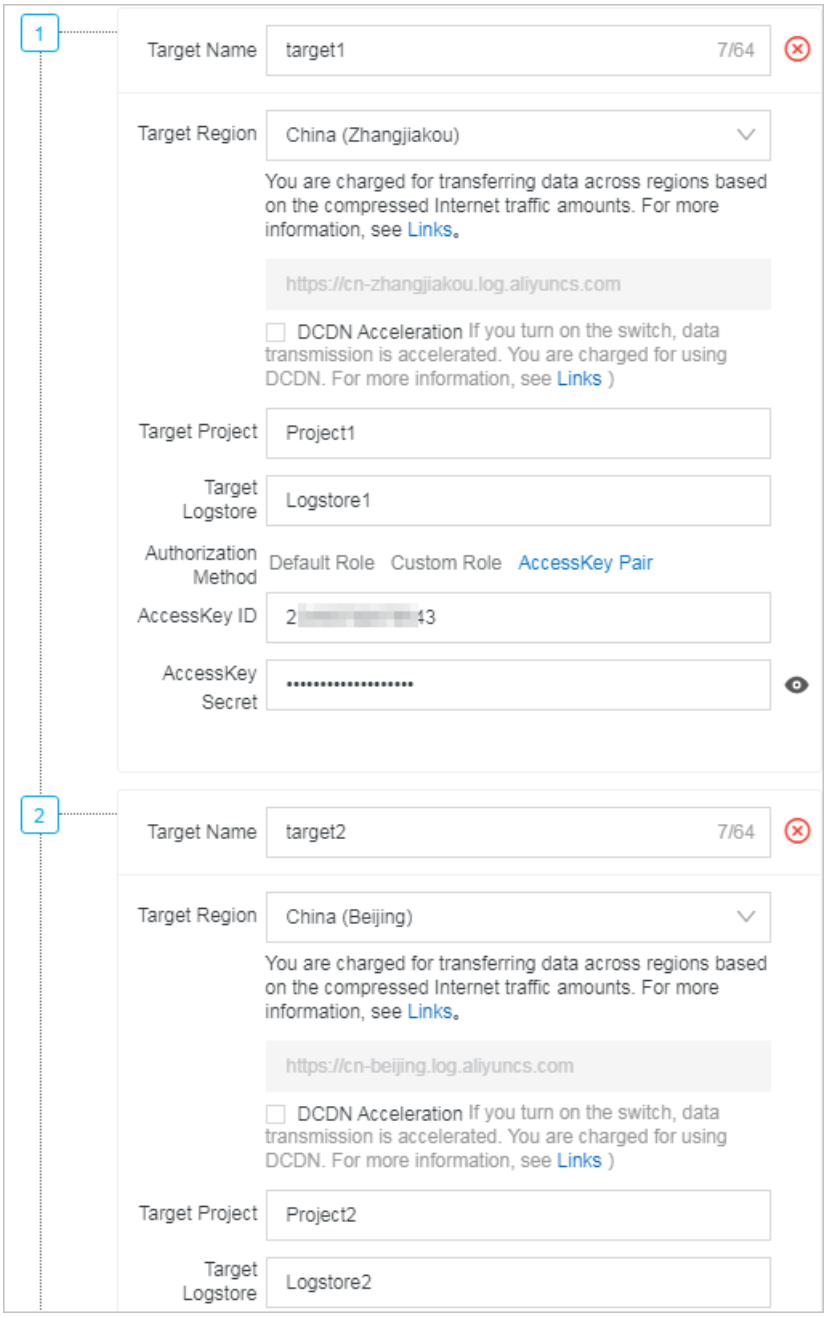
- Transformation rule:

```
e_switch(e_match(v("project"), "Project1"), e_output(name="target1", project=v("project")
, logstore=v("logstore")),
         e_match(v("project"), "Project2"), e_output(name="target2", project=v("project")
, logstore=v("logstore")))
```

- Storage target

  On the **Create Data Transformation Rule** page, configure storage targets. For information about the parameters of storage targets, see Create a data transformation job.

  > **Note** In this scenario, the destination projects and Logstores are determined based on the settings of the project and logstore parameters in the e_output function. These projects and Logstores are irrelevant to the destination project and Logstore that you configure to store the default storage target (labeled 1) on the **Create Data Transformation Rule** page.



Label	Storage target	Destination project and Logstore	AccessKey
1	target0	Project0 and Logstore0	None
2	target1	Randomly selected by the e_output function	The AccessKey pair of Account A
3	target2	Randomly selected by the e_output function	The AccessKey pair of Account B

- Result:

```
## Log entry that is distributed to Logstore1 of Project1 that belongs to Account A.
host:  example.aliyundoc.com
project: Project1
logstore: Logstore1
http_status:  200
request_method:  GET
request_uri:  /pic/icon.jpg
scheme:  https
## Log entry that is distributed to Logstore2 of Project1 that belongs to Account A.
host:  demo.aliyundoc.com
project: Project1
logstore: Logstore2
http_status:  301
request_method:  POST
request_uri:  /data/data.php
scheme:  http
## Log entry that is distributed to Logstore1 of Project2 that belongs to Account B.
host:   learn.aliyundoc.com
project: Project2
logstore: Logstore1
http_status:  404
request_method:  GET
request_uri:  /category/abc/product_id
scheme:  https
## Log entry that is distributed to Logstore2 of Project2 that belongs to Account B.
host:   guide.aliyundoc.com
project: Project2
logstore: Logstore2
http_status:  504
request_method:  GET
request_uri:  /data/index.html
scheme:  https
```

Scenarios 4: multi-source dynamic distribution

For example, assume that you advertise a game and store log entries of all API requests of the game in a Logstore. You want to parse the user-agent request header and then distribute the log entries of the requests from iOS, Android, and Windows platforms based on the user-agent request header. You also want to analyze the advertisement conversion rate based on the request_method field.

In this scenario, you can use the data transformation and query features to distribute the log entries.

- Raw log entry:

```
__source__:127.0.0.0
__tag__:__receive_time__: 1589541467
ip:10.0.0.0
request_method: GET
user_agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:50.0) Gecko/20100101 Firefox/
50.0
```

- Distribution requirements
  - Store log entries of requests from Windows users as target1 in Logstore1.
  - Store log entries of requests from iOS users as target2 in Logstore2.

  - Store log entries of requests from Android users as target3 in Logstore3.

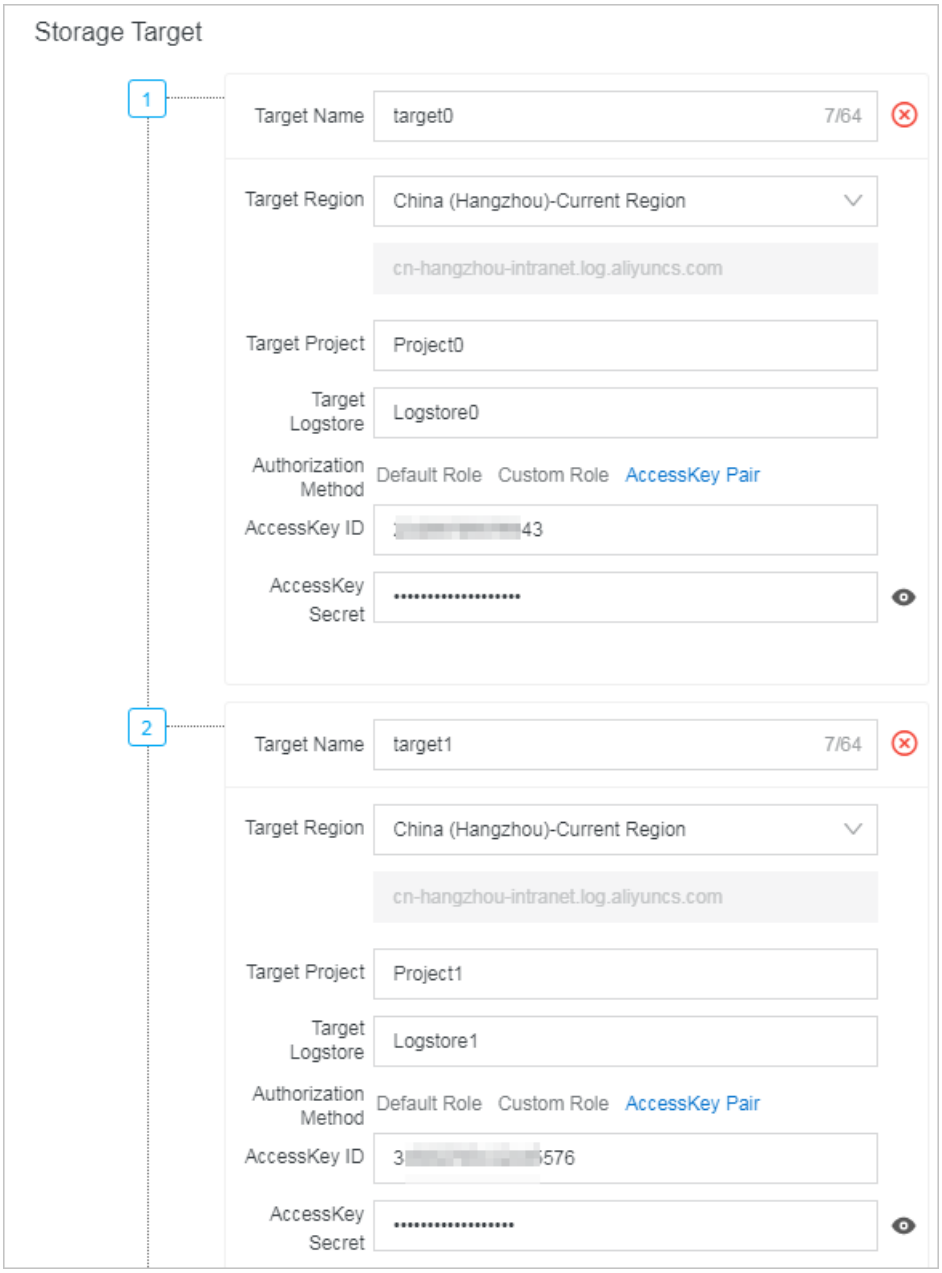
- Transformation rule:

  In this scenario, you can use the ua_parse_os function to parse the user_agent field and use the dct_get function to retrieve the information of the operating system from the user_agent request header. Then, you can use the e_set function to add the os field to each log entry and use the e_output and e_if functions to distribute log entries. The value of the os field is the information of the operating system.

```
e_set("os", dct_get(ua_parse_os(v("user_agent")),"family"))
e_if(e_search("os==Windows"),e_output(name="target1"))
e_if(e_search("os=iOS"),e_output(name="target2"))
e_if(e_search("os==Android"),e_output(name="target3"))
```

- Storage target

  On the **Create Data Transformation Rule** page, configure storage targets. For information about the parameters of storage targets, see Create a data transformation job.



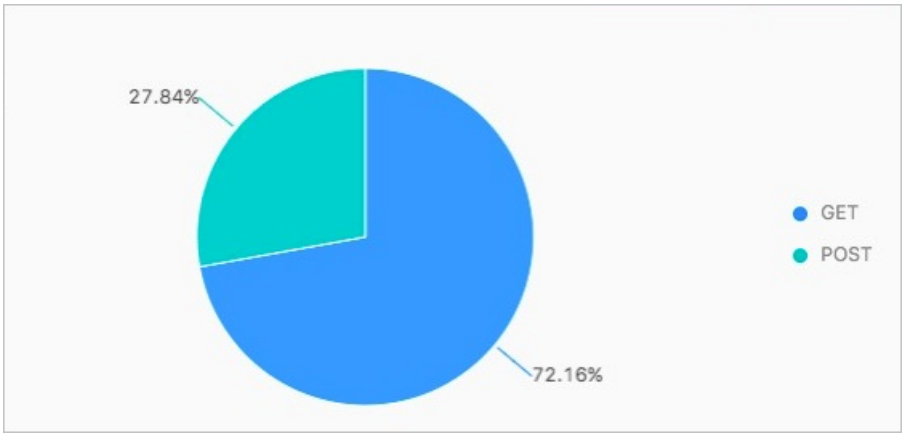
Label	Storage target	Destination project and Logstore
1	target0	Project0 and Logstore0
2	target1	Project1 and Logstore1
3	target2	Project2 and Logstore2
4	target3	Project3 and Logstore3

- Log query and analysis

  You can query log data in the destination Logstores to retrieve the advertisement conversion rate. The following query results show that Android users have a higher advertisement conversion rate. For more information about how to query log data, see Query and analyze logs.

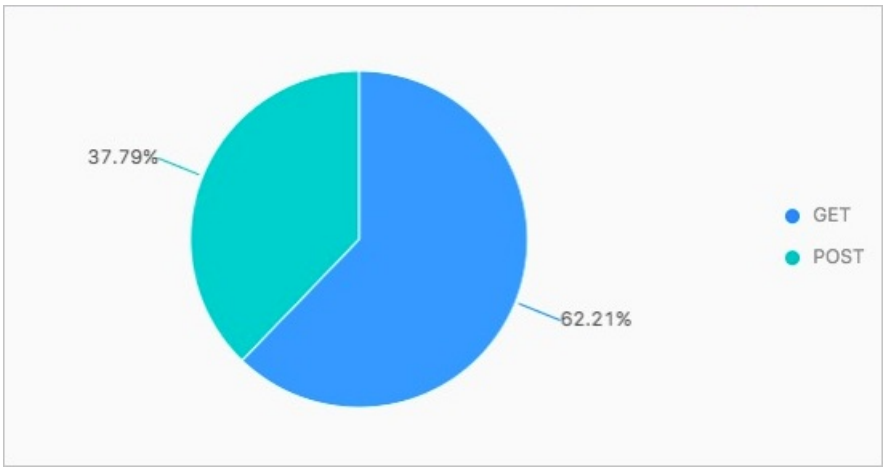
- On the Search & Analysis page of Logstore2, run the following query statement to retrieve the ratio of GET and POST requests from iOS users:

```
* | SELECT Request_method, COUNT(*) as number GROUP BY Request_method
```



- On the Search & Analysis page of Logstore3, run the following query statement to retrieve the ratio of GET and POST requests from Android users:

```
* | SELECT Request_method, COUNT(*) as number GROUP BY Request_method
```



15.7.4. Aggregate data from multiple source Logstores

Log Service allows you to aggregate data from multiple source Logstores to a single destination Logstore. To do this, you can configure a data transformation task for each source Logstore. This topic describes how to aggregate data from multiple source Logstores to a destination Logstore in a typical scenario.

Context

For example, assume that a multinational information company stores the access logs of its website in different Logstores that belong to different Alibaba Cloud accounts. The company needs to aggregate the access logs from the Logstores in the UK (London) region for data query and analysis. To do this, the company can use the e_output function in Log Service to aggregate the transformation

results to a destination Logstore.

Aggregate data from multiple source Logstores

- Raw log entries
  - The raw log entries are stored in a Logstore named Logstore_1 of a project named Project_1. The project belongs to Account 1 and resides in the UK (London) region.

```
Log entry 1
request_id: 1
http_host:  example.com
http_status:  200
request_method:  GET
request_uri:  /pic/icon.jpg
Log entry 2
request_id: 2
http_host:  aliyundoc.com
http_status:  301
request_method:  POST
request_uri:  /data/data.php
```

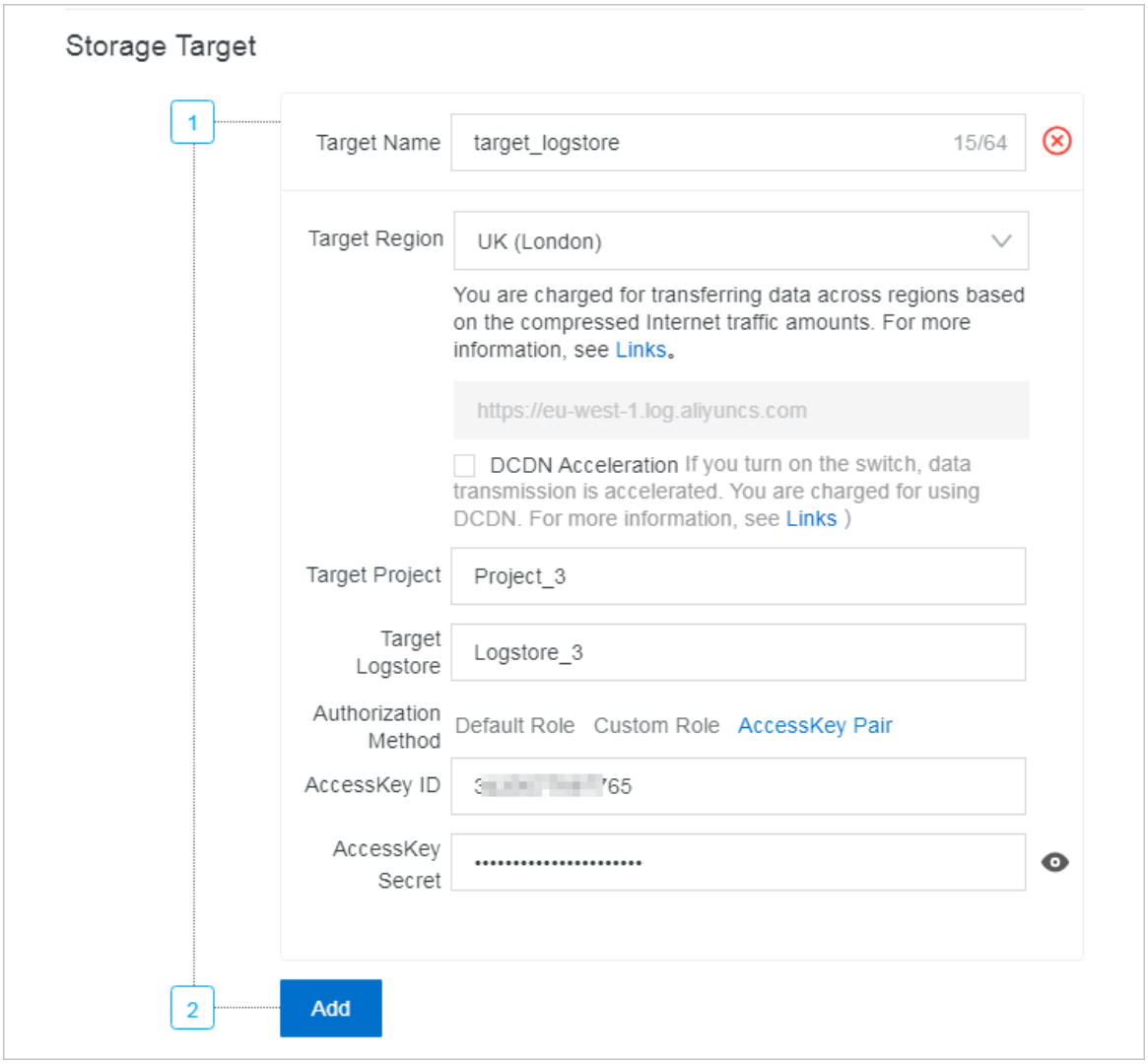
  - The raw log entries are stored in a Logstore named Logstore_2 of a project named Project_2. The project belongs to Account 2 and resides in the UK (London) region.

```
Log entry 1
request_id: 3
host:  example.edu
status:  404
request_method:  GET
request_uri:  /category/abc/product_id
Log entry 2
request_id: 4
host:  example.net
status:  200
request_method:  GET
request_uri:  /data/index.html
```

- Transformation requirements
  - Aggregate the log entries whose `http_status` is *200* in Logstore_1 of Account 1 and Logstore_2 of Account 2 to Logstore_3 of Account 3.
  - Set the log field names in Logstore_1 of Account 1 and Logstore_2 of Account 2. Set the field name `host` to `http_host` and the field name `status` to `http_status`.
- DSL orchestration

- Configure the following transformation rules for Logstore_1 of Account 1. On the **Create Data Transformation Rule** page, set Target Name to target_logstore, set Target Project to Project_3, set Target Logstore to Logstore_3, and specify the authorization method. For more information, see Create a data transformation job.

```
e_if(e_match("http_status", "200"), e_output("target_logstore"))
```



- Configure the following transformation rules for Logstore_2 of Account 2. On the **Create Data Transformation Rule** page, set Target Name to target_logstore, set Target Project to Project_3, set Target Logstore to Logstore_3, and specify the authorization method.

```
e_if(e_match("status", "200"), e_compose(e_rename("status", "http_status", "host", "http_host"), e_output("target_logstore")))
```

- Result

  The following log entries are aggregated in the *Logstore_3* of *Project_3* that belongs to Account 3. The project resides in the UK (London) region.

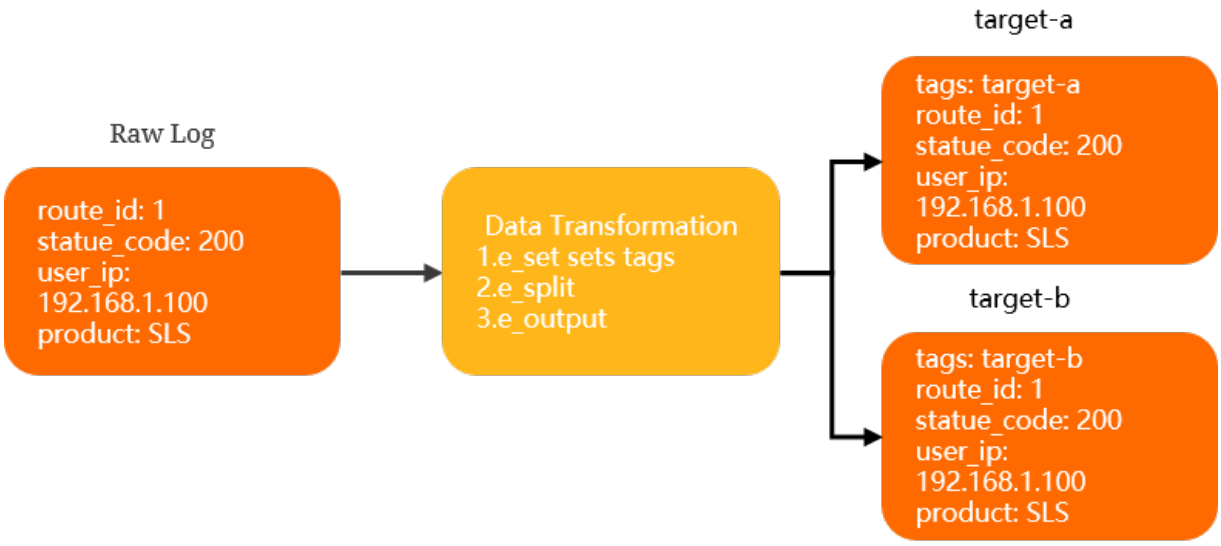
```
Log entry 1
request_id: 1
http_host:  example.com
http_status:  200
request_method:  GET
request_uri:  /pic/icon.jpg
Log entry 2
request_id: 4
http_host:  example.net
http_status:  200
request_method:  GET
request_uri:  /data/index.html
```

15.7.5. Replicate and distribute data

Log Service allows you to replicate data from a source Logstore and distribute the data to multiple destination Logstores. Before you can replicate and distribute the data, you must create a data transformation rule for the source Logstore. This topic describes how to replicate data from a source Logstore and distribute the data to multiple destination Logstores in a typical use scenario.

Use scenario

A data analytics company wants to replicate all log entries from a source Logstore and distribute the log entries to two destination Logstores. The replication and distribution features of Log Service allow the company to use the e_set function to specify tags, use the e_split function to categorize the log entries based on the tags that you specify, and then use the e_output function to distribute each category of log entries to the destination Logstore that matches the tag for the category. The following figure shows the basic logic of the replication and distribution features.



Before you begin, make sure that the following operations are complete:

- Evaluate the performance of the two destination projects, target-a and target-b. For example, evaluate the number of shards in each of these Logstores. For more information, see Performance guide.
- Create two projects, target-a and target-b. Then, create two Logstores, logstore-a and logstore-b. For more information, see Manage a project and Manage a Logstore.

Procedure

1.
2.
3.
4. In the upper-right corner of the page that appears, click **Data Transformation**.
5. In the editor that appears, enter the following statements:

```
e_set("tags", "target-a,target-b")
e_split("tags")
e_if(op_eq(v("tags"), "target-a"), e_output("target-a"))
e_if(op_eq(v("tags"), "target-b"), e_output("target-b"))
e_drop()
```

   - Use the e_set function to specify target-a and target-b as tags for raw log entries. For more information, see e_set.
   - Use the e_split function to categorize raw log entries based on the tags that you specify. For more information, see e_split.
   - Use the e_output function to distribute the log entries that are categorized by the e_split function to the target-a and target-b storage destinations. For more information, see e_output.
   - Use the e_drop() function to specify conditions. If a log entry does not meet the specified conditions, the log entry is dropped and is not distributed. For more information, see e_drop.
6. Click **Preview Data**.

   On the Transformation Results tab, the specified tags are added to raw log entries. The raw log entries with the target-a tag are distributed to the target-a storage destination, and the raw log entries with the target-b tag are distributed to the target-b storage destination.
7. Click **Save as Transformation Rule**.
8. In the **Create Data Transformation Rule** panel, configure the parameters.
   i. Configure the basic parameters.

Parameter	Description
Rule Name	The name of the data transformation rule. For this example, enter test.
Authorization Method	The role that is attached to Log Service. The role specifies the permissions that Log Service has to read data from the source Logstore. For this example, select Default Role.

ii. Configure the parameters for the target-a storage destination.

Parameter	Description
Target Name	The name of the storage destination. For this example, enter target-a.
Target Region	The region where the destination project resides. For this example, select China (Hangzhou).
Target Project	The name of the destination project. For this example, enter target-a.
Target Logstore	The name of the destination Logstore. For this example, enter logstore-a.
Authorization Method	The role that is attached to Log Service. The role specifies the permissions that Log Service has to read and write data to the target-a storage destination. For this example, select Default Role.

iii. Configure the parameters for the target-b storage destination.

Parameter	Description
Target Name	The name of the storage destination. For this example, enter target-b.
Target Region	The region where the destination project resides. For this example, select China (Hangzhou).
Target Project	The name of the destination project. For this example, enter target-b.
Target Logstore	The name of the destination Logstore. For this example, enter logstore-b.
Authorization Method	The role that is attached to Log Service. The role specifies the permissions that Log Service has to read and write data to the target-b storage destination. For this example, select Default Role.

iv. Configure the parameters in the Processing Range section.

Parameter	Description
Time Range	The time range over which the log entries that are processed by Log Service are generated. If you select All, Log Services transforms all log entries in the source Logstore.

9. Click **OK**.

Result

- Open the target-a project. On the **Logstores** tab of the **Log Storage** page, select the logstore-a Logstore. Then, you can view the log entries that are distributed to the logstore-a Logstore.
- Open the target-b project. On the **Logstores** tab of the **Log Storage** page, select the logstore-b Logstore. Then, you can view the log entries that are distributed to the logstore-b Logstore.

16.FAQ

16.1. FAQ about data transformation

This topic lists some frequently asked questions about the data transformation feature of Log Service.

- Troubleshooting overview
- How can I fix the startup errors of the data transformation engine?
- How can I fix errors that occur when the data transformation engine reads data from a source Logstore?
- How can I fix errors related to data transformation rules?
- How can I fix data pull errors?
- How can I fix errors that occur during data outputs to the target Logstore?
- How can I fix errors that occur when I pull Logstore data (dimension table)?
- How can I fix errors that occur during data pulls from OSS?
- How do I fix errors in the syntax used to load data from ApsaraDB RDS for MySQL?
- How do I dynamically construct a field?
- How do I send a log entry to different storage targets with different field sets?
- What do I do if the destination Logstores contain no data?
- What do I do if the destination Logstores contain unexpected data?
- What do I do if latency occurs during data transformation?

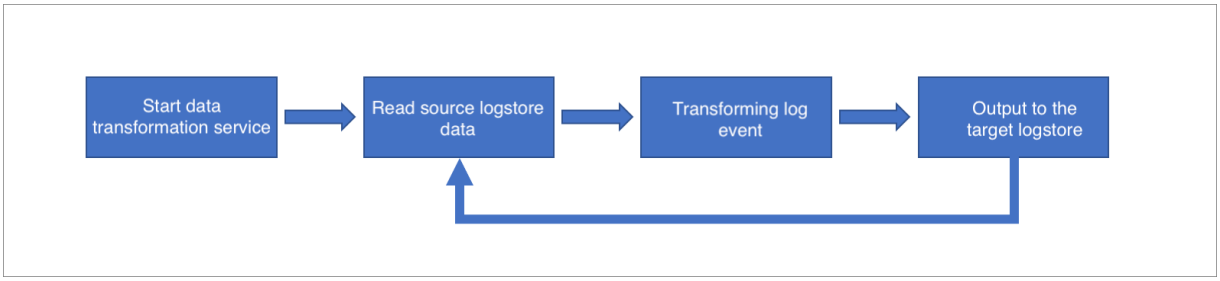
16.2. Troubleshooting overview

After a data transformation task is started, the data transformation engine sends the transformation results to the destination Logstores based on routing rules. This topic provides troubleshooting methods that you can use if a data transformation task fails. For example, no log is generated in the destination Logstores or a long delay occurs during the data transformation process.

Analyze errors

If an error occurs, you can first locate the error in the data transformation process.

Each data transformation task consists of four steps based on Data transformation basics. The following figure shows the steps.



Errors may occur in each of these steps. The causes, impacts, and troubleshooting methods vary depending on the steps.

- Start the data transformation engine.

- Errors may occur in this step if domain-specific language (DSL) rules fail the security check in the data transformation engine.
- If an error occurs in this step, the data transformation task stops. You must modify the DSL rules and restart the data transformation task. If the retry succeeds, the transformation task continues and no data loss or redundancy occurs.

For more information about the troubleshooting methods in this step, see How can I fix the startup errors of the data transformation engine?.

- Read data from the source Logstore.
  - Errors may occur in this step due to a failure to access the source Logstore. The failure may be caused by invalid configurations of the source Logstore, network errors, or an update of the source Logstore.
  - If an error occurs in this step, the data transformation task keeps retrying until data reading succeeds or is manually stopped. If the retry succeeds, the transformation task continues and no data loss occurs.
  - If an error occurs after some data is read, the data transformation task saves the checkpoint and keeps retrying. After the retry succeeds, it continues to read data from the checkpoint and no data loss or redundancy occurs. If the task is stopped during the retry process, no data loss or redundancy occurs.

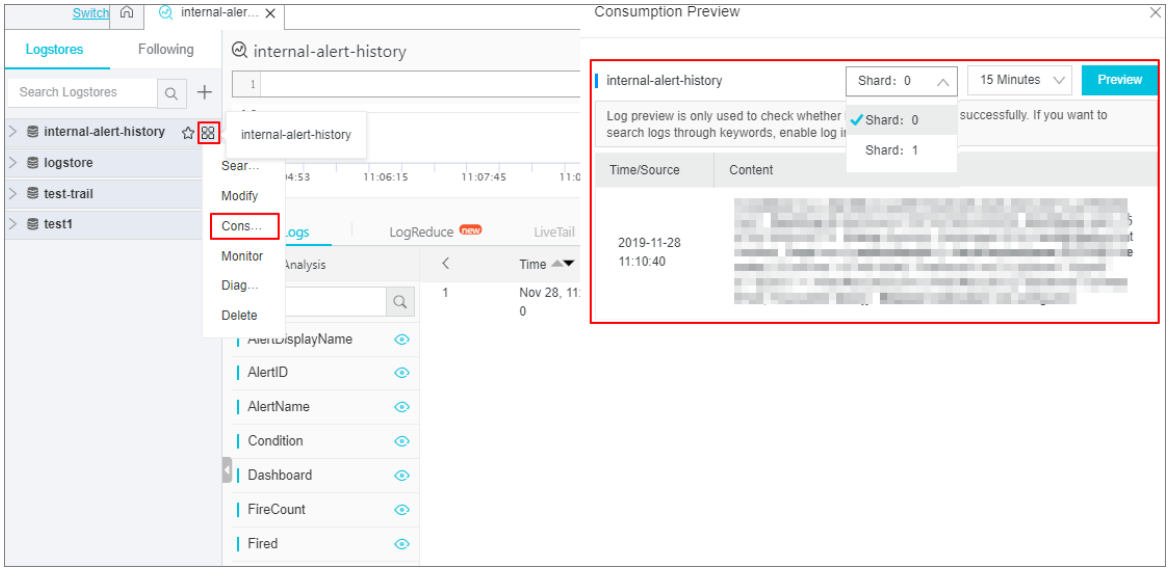
  For more information about the troubleshooting methods in this step, see How can I fix errors that occur when the data transformation engine reads data from a source Logstore?.
- Transform log events.
  - Errors may occur in this step if the transformation rules do not apply to all or some log events during the data transformation process.
  - During the transformation process, log events that conflict with the transformation rules cause errors. These errors are divided into WARNING and ERROR levels. The error levels are identified by the logging.levelname field.
    - For ERROR-level errors, the relevant log events are dropped. The transformation result does not contain these log events.
    - For WARNING-level errors, data is not transformed in the current DSL. For example, if the log events do not match the specified regular expression, the next step is performed.

  For more information about the troubleshooting methods in this step, see How can I fix errors related to data transformation rules?.
- Export the transformation results to the destination Logstores.
  - Errors may occur in this step due to a failure to access the destination Logstore. The failure may be caused by invalid configurations of the destination Logstore, network errors, or an update of the destination Logstore.
  - If an error occurs in this step, the data transformation task keeps retrying until data reading succeeds or is manually stopped. If the retry succeeds, the transformation task continues and no data loss occurs.
  - If an error occurs after some data is exported, the transformation task keeps retrying. For example, assume that two destination Logstores are specified. Data export from one Logstore succeeds, but data from the other Logstore fails. In this case, the transformation task saves the breakpoint and keeps retrying. After the retry succeeds, no data loss or redundancy occurs. If the data transformation task is stopped and then restarted when the error occurs, the data transformation task continues from the breakpoint. In this case, no data is lost, but data redundancy may occur.

Troubleshoot common errors

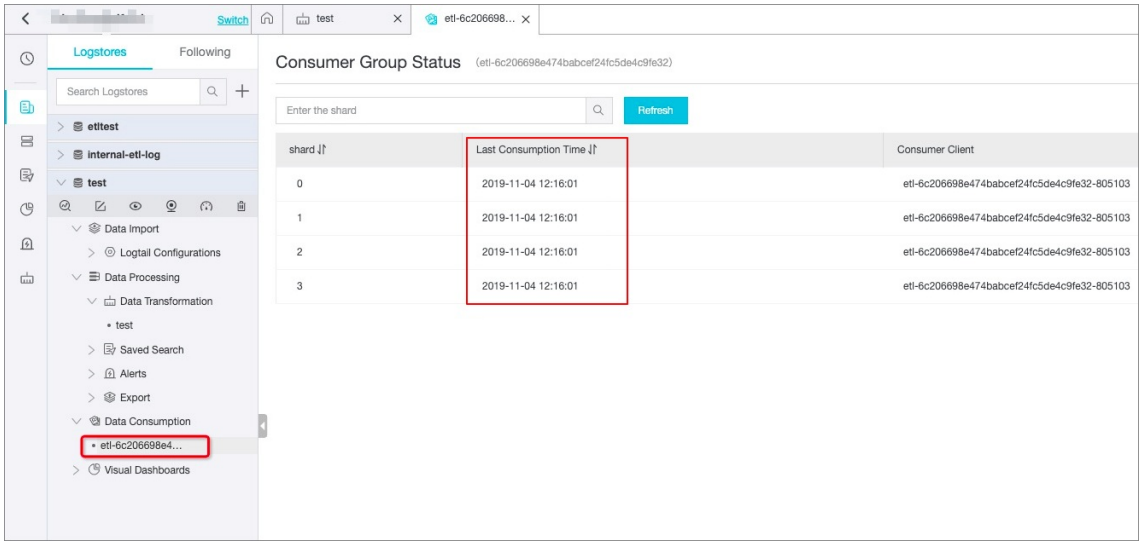
1. Check whether data has been written to a destination Logstore.

   To check whether data has been written to a destination Logstore recently, you can view the data on the Consumption Preview page of the destination Logstore.



   > **Note** The consumption preview data may be inaccurate due to the following reasons:
   >
   > - In Log Service, logs are transformed based on the time when logs are received. In scenarios where historical logs are being transformed, the time when the logs are written may not be within the time range specified for querying historical logs.
   > - You can query log data based on the indexes of historical logs. However, a delay of several minutes may occur. If historical logs are being written in a data transformation task, the log data may not be immediately queried.

2. View the status of a data transformation task.
   - Check whether the current task is started. For more information, see View the status of a job. Tasks with a fixed time range automatically stop at the specified end time.
   - Check whether the consumer group in the current task is enabled and updated.



- Check whether errors occur. For more information, see View error logs. If an error occurs, identify the error cause and fix the error. For more information, see Analyze errors.

3. Check whether data is generated in the source Logstore.

   Check whether logs exist in the source Logstore within the time range of the current data transformation task.

   - If the end time of the time range is not specified, check whether new logs are generated in the source Logstore. If no new logs are generated and no historical logs exist within the specified time range, the data transformation task cannot be performed.
   - If you select a historical time range, check whether logs exist in the source Logstore within the time range.

   Click Modify a transformation rule of the data transformation task, select a time range, and then check whether raw logs exist in the specified time range.

4. Check whether the transformation rules are valid.

   Check whether code errors exist in the transformation rule. Example:

   - The log time is modified. As a result, no logs can be found in the specified time range.
   - Logs are dropped based on the transformation rule under specific conditions.

     For example, assume that the transformation rule contains the following code. If a log does not contain the `name` field or the value of this field is null, the log is dropped. The pre-logic in the code is used to construct the `name` field. If the `name` field is not constructed due to a pre-logic issue, no log is generated.

     ```
     # .... The pre-logic.
     #  .... Build the name field...
     e_keep(e_search('name: "?"'))
     ```

   - If the task pulls data from a third party for data enrichment, check whether the data size of the third party is too large. If so, the data transformation task cannot immediately start to consume data and may stay in the initializing state for a long time. Example:

     ```
     e_dict_map(res_rds_mysql(..database="userinfo", table="user"), "username", ["city", "
     school", "age"])
     ```

Click Modify a transformation rule of the data transformation task, select a time range, and then click Preview Data to view the result.

If logs are queried, comment out the specific statement that causes the error and preview the result again.

5. Check whether the number of shards is as expected.

   If data transformation is too slow, check whether the source and destination Logstores meet your performance expectations. We recommend that you adjust the number of shards in the source or destination Logstore.

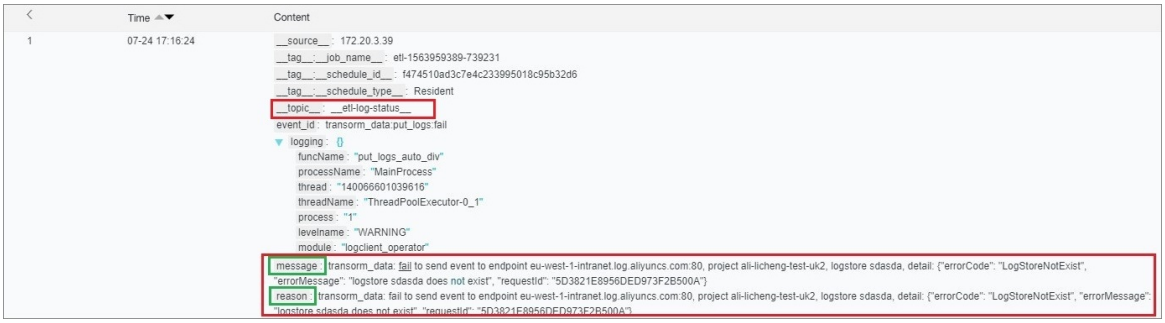
View error logs

You can view error logs by using the following methods:

- View error logs in the `internal-etl-log` Logstore.

  Logs generated by a data transformation task are stored in the `internal-etl-log` Logstore. This Logstore is automatically generated after Log Service performs the data transformation task.

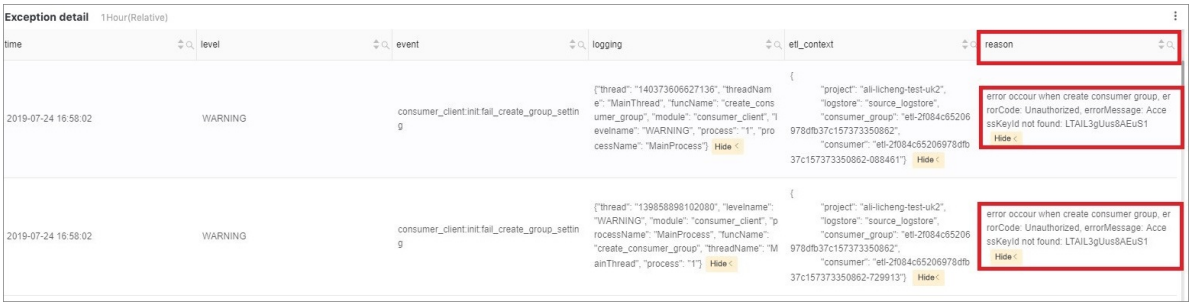
  - The `internal-etl-log` Logstore is a dedicated Logstore that is provided free of charge. You cannot modify its configurations or write data to it.
  - In the `internal-etl-log` Logstore, the `__topic__` field of each log event indicates the status of the data transformation task. You can check whether an error occurs in the data transformation task based on this field.
  - You can check the `message` and `reason` fields of each log event to view the detailed error information. The following figure shows the fields.



- View error logs on the dashboard.

  Click a data transformation task and check the dashboard in the **Status** section on the **Data Transformation Overview** page.

  The error information is shown in the `reason` column in the Exception detail section.



- View error logs in the console.

Error logs in the preview phase are displayed in the Log Service console. In the preview phase, Log Service simulates the operations specified by the transformation rule and check whether the data is transformed as expected. No changes are made to the source or destination Logstore. Therefore, errors that occur in the preview phase do not affect the source log events.

Preview limits

Compared with real data transformation tasks, data transformation in the preview phase has the following limits:

- No error occurs even if a RAM user uses an invalid AccessKey pair to access the source Logstore.

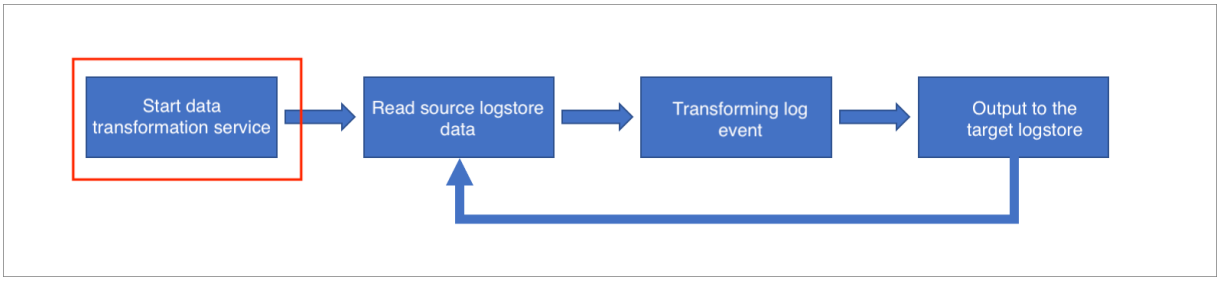
  In the preview phase, no consumer groups are created to consume data. Therefore, Log Service does not check the permissions of consumer groups.
- No error occurs even if a name error of the destination Logstore occurs in the transformation rule.

  Data is not written to the destination Logstore in the preview phase. Therefore, the system does not check whether the destination Logstore is correctly configured in the transformation rule.
- No error occurs even if configuration errors of the destination Logstore exist.
  - The configuration errors include invalid destination project, destination Logstore, and AccessKey pair configurations.
  - Data is not written to the destination Logstore in the preview phase. Therefore, the system does not check whether the configurations of the destination Logstore are correct.
- Only partial data is pulled in the preview phase.
  - By default, only 1,000 data records are pulled from the source Logstore for data transformation during preview.
  - If no transformation result is generated after the first 1,000 data records are transformed, Log Service continues to pull data for 5 minutes until a transformation result is generated.

16.3. How can I fix the startup errors of the data transformation engine?

This topic describes the causes of the startup errors that are related to the data transformation engine. This topic also provides methods that you can use to troubleshoot these errors.

Before Log Service runs a data transformation task based on a data transformation rule, the data transformation engine must be started first. If domain-specific language (DSL) rules fail the security check in the data transformation engine, errors may occur in this step.



Error log

If invalid Log Service DSL rules are detected during the startup process of the transformation engine, the following error message is returned:

```
{
  "errorMessage": "ETL config doesn't pass security check, detail: XXXXXX"
}
```

Note You can view error logs in the exception details of the diagnostic report for data transformation or in the internal-etl-log Logstore.

- If an error occurs during the transformation engine startup phase, Log Service retries the data transformation task until a retry is successful or until you manually stop retries.
- After you modify the data transformation rule and a retry is successful, the data transformation engine works as expected. No data is dropped or duplicated.

Troubleshoot common errors

- The basic syntax is invalid.

  The compiled data transformation rules do not conform to the Log Service DSL syntax. For example, the rules contain unpaired parentheses (), or commas (,) are incorrectly written as colons (:).

  - Error logs

    ```
    {
      "errorMessage": "ETL config doesn't pass security check, detail: invalid syntax"
    }
    {
      "errorMessage": "ETL config doesn't pass security check, detail: unexpected EOF while parsing"
    }
    ...
    ```

  - Troubleshooting method

    Locate the specific syntax error based on the `traceback` information in the related error log. For example, `e_set("test", v("status"))` is incorrectly written as `e_set("test": v("status"))`, as shown in the following figure.

- Invalid operators are used.

All operations in Log Service DSL must be specified by using the functions supported by Log Service DSL. For example, numerical operations or size comparisons must be specified by using the op_* function. You cannot use only operators.

- Error log

```
{
  "errorMessage": "ETL config doesn't pass security check, detail: invalid type detecte
d: <class `_ast.BinOp`> "
}
```

- Troubleshooting method

  Check Log Service DSL rules. Make sure that all operations such as numerical operations and size comparisons are specified by using the functions supported by Log Service DSL.

- Examples

```
e_set("b", v("a") - 10) # Invalid example
e_set("b", op_sub(v("a"), 10)) # Valid example
e_set("b", v("a") >= v("c")) # Invalid example
e_set("b", op_ge(v("a"), v("c"))) # Valid example
```

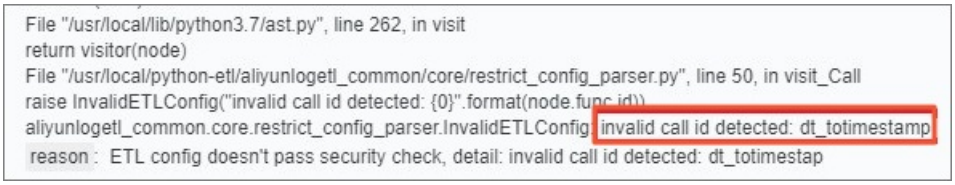
- If the type of the parameter that is passed to a function is invalid or the called function does not exist, an error occurs.

  If the type of the parameter that is passed to a function is different from the type of the parameter that is received by the function or the called function does not exist, an error occurs.

  - Error log

```
{
  "errorMessage": "ETL config doesn't pass security check, detail: invalid call in dete
cted: function_name"
}
```

  - Troubleshooting method
    - Check whether the called function exists and the function name is correct. If the function exists and the name is valid, check whether the type of the parameter that is passed to the function is valid.
    - Locate the error function based on the `traceback` information in the related error log. For example, if the returned error log indicates that the dt_totimestamp function is invalid, you must check whether the function exists, and then check whether the type of the parameter that is passed to the dt_totimestamp function is valid.



- Examples

  The type of the parameter received by the dt_totimestamp function is a datetime object, and `v("time1")` in the code is a string. An error occurs because the type of the parameter that is passed to the function is invalid.

  To fix the error, use the dt_parse function to convert the string to a datetime object before the parameter is passed to the dt_totimestamp function. You can also use the dt_parsetimestamp function that can receive strings instead of the dt_totimestamp function.

```
# Invalid examples
e_set("time1", "2019-06-03 2:41:26")
e_set("time2", dt_totimestamp(v("time1")))
# Valid examples
e_set("time1", "2019-06-03 2:41:26")
e_set("time2", dt_totimestamp(dt_parse(v("time1"))))
# Valid examples
e_set("time1", "2019-06-03 2:41:26")
e_set("time2", dt_parsetimestamp(v("time1")))
```

- Expression functions are globally called.

  The Log Service DSL syntax supports two types of functions: global operation functions and expression functions. Only global operation functions can be globally called in the data transformation process. If an expression function is globally called, an error occurs.

  - Error log

```
{
  "errorMessage": "ETL config doesn't pass security check, detail:  invalid type detect
ed: <class '_ast.Expr'>"
}
```

  - Troubleshooting method

    Check whether an expression function is globally called in the data transformation process.

  - Examples

```
# Invalid examples
op_add(v("a"), v("b"))
str_lower(v("name"))
# Valid examples
e_set("add", op_add(v("a"), v("b")))
e_set("lower", str_lower(v("name")))
```

- Parameters are specified by using variable values.

  The Log Service DSL syntax does not support value assignment by using variables. Variable values can only be passed in stateless mode.

  - Error log

```
{
  "errorMessage": "ETL config doesn't pass security check, detail: invalid assign detec
ted: variable_name"
}
```

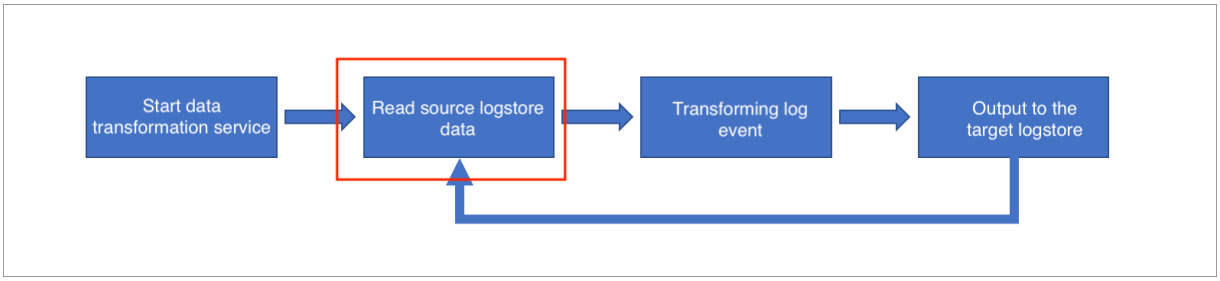
- Troubleshooting method
  - Check whether variables are used to assign values in the Log Service DSL rule.
  - Locate the error based on the `traceback` information in the related error log.
- Examples

```
# Invalid examples
sum_value = op_add(v("a"), v("b"))
e_set("sum", sum_value)
# Valid example
e_set("sum", op_add(v("a"), v("b")))
```

16.4. How can I fix errors that occur when the data transformation engine reads data from a source Logstore?

This topic describes the causes of the errors that occur when the data transformation engine reads data from a source Logstore. This topic also provides methods that you can use to troubleshoot these errors.

After the data transformation engine is started, it reads data from the source Logstore in streaming mode. The data transformation engine continuously reads data from the source Logstore when the data is being transformed.



If the source Logstore cannot be accessed, errors may occur in this step. This issue may occur due to the following causes:

- The configurations of the source Logstore are invalid.
- The information of the source Logstore is changed.
- A network error occurs.

Error impact:

- If an error occurs when the data transformation engine reads data, Log Service retries the related data transformation task until a retry is successful or until you manually stop retries. If the retry succeeds, the data transformation task runs as expected.
- If an error occurs after some data is read, Log Service saves breakpoints and retries the data transformation task. After a retry is successful, the data transformation engine continues to read data from the last breakpoint. No data is dropped or duplicated.

Troubleshoot common errors

- An invalid AccessKey pair that consist of an AccessKey ID and an AccessKey secret is specified for the source Logstore.
  - Error logs

    ```
    {
      "errorCode": "Unauthorized",
      "errorMessage": "AccessKeyId not found: LTAIL3gUus8A****"
    }
    {
      "errorCode": "SignatureNotMatch",
      "errorMessage": "signature uJfAJbc0ji04gb+cXhh0qWt****= not match"
    }
    ```

  - Troubleshooting method

    Check whether the specified AccessKey ID and AccessKey secret exist and are valid.

- The information of the source Logstore is changed.

  The configurations of the source Logstore are valid and the related data transformation task is running as expected. However, the information of the source Logstore is changed during the data transformation process. In this case, the source Logstore cannot be accessed.

  - Error logs

    The information of the source Logstore is changed in the following two conditions:

    - The source Logstore is deleted. In this case, the following error message is returned:

      ```
      {
        "errorMessage": "Logstore [logstore_name] does not exist."
      }
      ```

    - The AccessKey ID or the AccessKey secret of the source Logstore is changed. In this case, the following error messages are returned:

      ```
      {
        "errorCode": "Unauthorized",
        "errorMessage": "AccessKeyId not found: LTAIL3gUus8A****"
      }
      {
        "errorCode": "SignatureNotMatch",
        "errorMessage": "signature uJfAJbc0ji04gb+cXhh0qWt****= not match"
      }
      ```

  - Troubleshooting method
    - Check whether the source Logstore is deleted.
    - Check whether the AccessKey ID or the AccessKey secret of the source Logstore is changed.

- A network error occurs.

- Error log

```
{
  "errorCode": "LogRequestError",
  "errorMessage": "HTTPConnectionPool(host='your_host', port=80): Max retries exceeded
with url: your_url (Caused by NewConnectionError: Failed to establish a new connection:
[Errno 11001] getaddrinfo failed'"
}
```

- Troubleshooting method

  Check whether the network is connected as required.

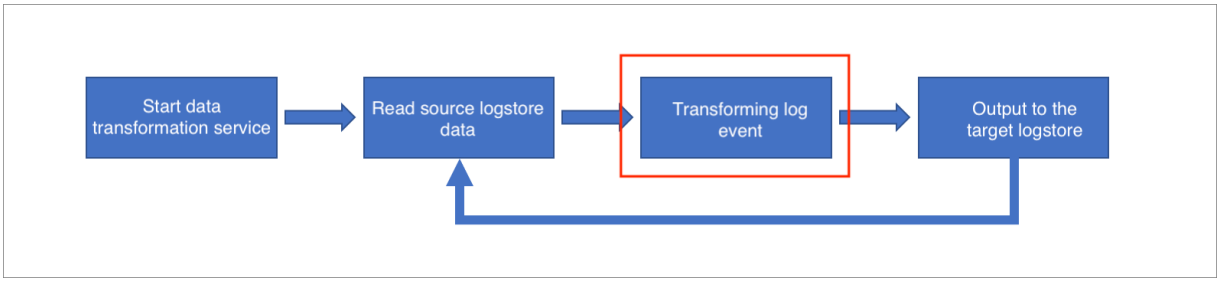
- No data is read from the source Logstore.

  No error message is returned. For more information, see Troubleshoot common errors.

16.5. How can I fix errors related to data transformation rules?

This topic describes the causes of errors that are related to data transformation rules and provides solutions.

After the data transformation engine reads data from a source Logstore, the engine starts to transform the log events of the Logstore.



- During the transformation process, logic errors may occur because the transformation rules may not apply to all log events.
- If the transformation rules involve data loading from external resources such as ApsaraDB RDS or Logstores, errors may also occur during the data loading or updating process.

This topic describes how to resolve logic errors. For information about how to resolve resource loading errors, see How can I fix data pull errors?.

Error impact

During the transformation process, log events that conflict with the transformation rules cause errors. These errors are divided into WARNING and ERROR levels. The error levels are identified by the logging.levelname field.

- For ERROR-level errors, the relevant log events are discarded. The transformation process continues and no retry is performed. The transformation result does not contain these log events.
- For WARNING-level errors, data is not transformed in the current domain-specific language (DSL). For example, if the log events do not match the specified regular expression, the next step is performed.

Solution

- Check the `logging.levelname` field of the error logs to determine the error levels.
- Check the `message` field to locate the error log events. For more information, see View error logs.
- Check the `reason` field of the error logs to identify the error causes.

Add logic to transform these error log events based on the error causes. You can use process control functions, such as the `e_if` and `e_switch` functions to identify and resolve the errors.

Error handling examples

- Log events contain abnormal values.

  Sample transformation rule 1:

  ```
  # The value of the b field in some log events is 0. This value is used as a divisor of a
  numeric operation and causes an error.
  e_set("c", op_div_floor(v("a"), v("b")))
  ```

  - Error log:

    ```
    {
      "reason": "error when calling : floordiv\nDetail: integer division or modulo by zero"
    ,
    }
    ```

  - Troubleshooting method:

    Check whether the value of the `b` field is *0* in the error logs. If the value of the b field is 0, this value is used as a divisor of a numeric operation and causes an error.

  - Solution:

    The error occurs only when the value of the `b` field is *0*. In this case, you can use the `e_if` function to capture the `b` field whose value is *0*.

    ```
    e_if_else(op_eq(v("b"), "0"), e_set("c", v("a")), e_set("c", op_div_floor(v("a"), v("b"
    )))
    ```

  Sample transformation rule 2:

  ```
  # The value of the a field in some log events is an invalid timestamp, which causes an er
  ror.
  e_set("b", dt_fromtimestamp(v("a")))
  ```

- Error log:

```
{
  "reason": "error when calling : int\nDetail: invalid literal for int() with base 10:
'invalid value'",
}
```

- Troubleshooting method:

  Check whether the value of the `a` field is a valid timestamp (numeric string).

- Solution:

  Add logic to check whether the value of the `a` field is a valid timestamp. If not, export the log event to `target2`.

```
e_if_else(str_isdigit(v("a")), e_set("b", dt_fromtimestamp(v("a"))), e_output("target2
"))
```

- The data type is not converted before a numeric operation.

  Sample transformation rule:

```
e_set("a", 10)
e_set("b", 10)
e_set("c", op_mul(v("a"), v("b")))
```

- Error log:

```
{
  "reason": "error when calling : mul\nDetail: can't mulltiply sequence by non-int of t
ype' str'",
}
```

- Error cause:

  The values of all fields in log events are stored as strings during the DSL transformation process. In the preceding sample transformation rule, the values of `v("a")` and `v("b")` are strings. If they are directly passed to the `op_mul` function, an error occurs.

- Troubleshooting method:

  Check the DSL rules to determine whether the data type is converted before the numeric operation.

- Solution:

  Use the `ct_int` function to convert the values from the string type to the integer type, and then pass them to the `op_mul` function.

```
e_set("a", 10)
e_set("b", 10)
e_set("c", op_mul(c_int(v("a")), c_int(v("b"))))
```

16.6. How can I fix data pull errors?

This topic describes the data pull errors that occur during data transformation and provides methods that you can use to troubleshoot these errors.

Error handling

For more information about how to troubleshoot data pull errors, see res_log_logstore_pull, res_rds_mysql, and res_oss_file.

Troubleshoot common errors

If you use only one resource function, data pull errors occur.

- Sample transformation rules

```
res_log_logstore_pull(endpoint="cn-shenzhen.log.aliyuncs.com",ak_id="xxx",
        ak_secret="xxx",project="etl-test-shenzhen",
        fields=["__source__"]),field="processid",output_fields=["xx"]
```

```
res_rds_mysql(address="xx",username="xx",password="xx",database="xx")
```

```
res_oss_file(endpoint='xx',ak_id="xx",ak_key="xx",bucket='xx', file='xx',format='xx',ch
ange_detect_interval=0)
```

- Error log

```
aliyun.log.logexception.LogException: {"errorCode": "InvalidEtlConfig", "errorMessage": "
ETL config doesn't pass security check, detail: invalid type detected: <class '_ast.Expr'
>", "requestId": ""}
```

- Troubleshooting method

  The preceding error log indicates that the syntax of the specified function is invalid. This error log is returned because you use only the res_log_logstore_pull, res_rds_mysql, or res_oss_file function. You cannot use only one resource function.

- Solution

  You must use a resource function together with another function, such as e_table_map or e_search_table_map. For more information, see Resource functions.

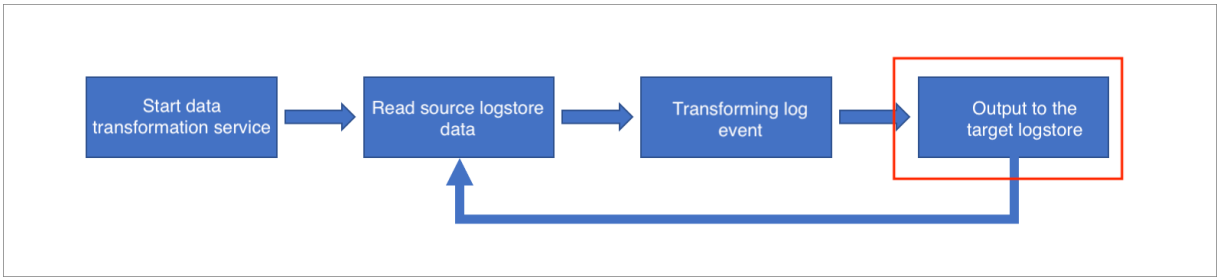
References

- For information about how to fix errors that occur when you pull data from another Logstore, see How can I fix errors that occur when I pull Logstore data (dimension table).
- For information about how to fix errors that occur when you pull data from Object Storage Service (OSS), see How can I fix errors that occur during data pulls from OSS.
- For information about how to fix errors that occur when you pull data from ApsaraDB RDS for MySQL, see How can I fix errors of the syntax used to pull data from ApsaraDB RDS for MySQL.

16.7. How can I fix errors that occur during data outputs to the target Logstore?

This topic describes how to fix errors that may occur when data transformation results are written to the target Logstore.

After the log events in the source Logstore are transformed, the transformation results are exported to the target Logstore. The data transformation engine reads a batch of source log records (up to 10,000 log records) at a time. Transformed data is stored in a cache pool and then exported to the target Logstore after all source log records are transformed.



Errors that may occur during the data output to the target Logstore are due to the following causes:

- The configurations of the target Logstore are incorrect.
- The information of the target Logstore is changed.
- A network connection error occurs.

Error impact:

- If an error occurs when data is exported to the target Logstore, the transformation task will retry until the data export succeeds or is manually stopped. If the retry succeeds, the transformation task continues and no data loss or redundancy occurs.
- If an error occurs after data is exported (for example, two target Logstores are specified. Data export from one Logstore succeeds, but data from the other Logstore fails), the transformation task saves the breakpoint and keeps retrying. After the retry succeeds, no data loss or redundancy occurs. If the transformation task is stopped and then restarted when the error occurs, the transformation task continues from the breakpoint. In this case, no data loss occurs, but data redundancy may occur.

Errors and troubleshooting methods

- The target Logstore is configured with an invalid

  AccessKey ID or AccessKey secret.

  - Error message

```
# The specified AccessKey ID is invalid.
{
  "errorCode": "Unauthorized",
  "errorMessage": "AccessKeyId not found: LTAIL3gUus8A****"
}
#The specified AccessKey secret is invalid.
{
  "errorCode": "SignatureNotMatch",
  "errorMessage": "signature uJfAJbc0ji04gb+cXhh0qWt****= not match"
}
```

  - Troubleshooting method

    Check the data transformation rule to make sure that the specified AccessKey ID and AccessKey secret of the target Logstore are valid and correct.

- The target project does not exist.
  - Error message

    ```
    {
      "errorCode": "ProjectNotExist",
      "errorMessage": "The Project does not exist : your_project_name"
    }
    ```

  - Error cause

    The error message `The Project does not exist` may be reported due to the following causes:

    - The target project name specified in the data transformation rule is incorrect.
    - The specified target project name is correct, but the project was deleted during the transformation task.
    - The target project is not in the same region as the project to which the transformation task corresponds.

      > **Note** The data transformation engine only supports data transmission between different projects in the same region.

  - Troubleshooting method
    - Check whether the target project name specified in the data transformation rule is correct.
    - Check whether the target project is deleted.
    - Check whether the target project is in the same region as the project to which the transformation task corresponds.
- The target project and Logstore do not exist.

  Transformation rule

  ```
  e_coutput("target1")
  ```

  - Error message

    ```
    {
      "errorMessage": "transform_data: output target target1 is not found in configurations
    "
    }
    ```

  - Error cause

    This error occurs because the target project and Logstore do not exist. The transformation rule exports log events to `target1`, but the error message `target1 is not found in configuratio ns` is reported. It means the project and Logstore corresponding to `target1` are not specified in the transformation rule.

  - Troubleshooting method

    Make sure that the corresponding project and Logstore are specified in the transformation rule.
- The information of the target Logstore is changed.

- Error cause

  This error occurred because the information of the specified target Logstore is changed during data transformation and the original configurations cannot be used to access the target Logstore.

- Error message

  The information of the target Logstore may have the following changes:

  - The target Logstore is deleted. In this case, the following error message is reported:

    ```
    {
      "errorMessage": "Logstore [logstore_name] does not exist."
    }
    ```

  - The AccessKey ID or AccessKey secret of the target Logstore is changed. In this case, the following error message is reported:

    ```
    # The AccessKey ID is invalid.
    {
      "errorCode": "Unauthorized",
      "errorMessage": "AccessKeyId not found: LTAIL3gUus8A****"
    }
    # The AccessKey secret is invalid.
    {
      "errorCode": "SignatureNotMatch",
      "errorMessage": "signature uJfAJbc0ji04gb+cXhh0qWt****= not match"
    }
    ```

- Troubleshooting method
  - Check whether the target Logstore is deleted.
  - Check whether the AccessKey ID or AccessKey secret of the target Logstore is changed.

- A network connection error occurs.
  - Error message

    ```
    {
      "errorCode": "LogRequestError",
      "errorMessage": "HTTPConnectionPool(host='your_host', port=80): Max retries exceeded
    with url: your_url (Caused by NewConnectionError: Failed to establish a new connection:
    [Errno 11001] getaddrinfo failed'"
    }
    ```

  - Troubleshooting method

    Check whether the network connection is normal.

16.8. How can I fix errors that occur when I pull Logstore data (dimension table)?

If the transformation rule of a Logstore also pulls data from another Logstore, data pull or update errors may occur. This topic describes these errors and the corresponding troubleshooting methods.

After reading data from a Logstore, the data transformation engine starts to transform log events in the Logstore. If the transformation rule involves data pulls from external resources such as Object Storage Service (OSS), ApsaraDB for RDS, and other Logstores, data pull or update errors may occur.

Data pull or update errors may have the following impacts:

- Log events that conflict with the transformation rule will cause errors and be dropped. These log events will not be included in the transformation result.
- Dropped log events will not be further transformed. The transformation task will continue to transform other log events.
- If a log event is split into multiple log events and a single split is dropped, all associated splits will be dropped.

> **Note** These log event splits are associated in a tree structure before they are exported.

Error 1: A required parameter is missing

- Transformation rule

```
e_table_map(res_log_logstore_pull(endpoint="cn-shenzhen.log.aliyuncs.com",ak_id="xxx",
        ak_secret="xxx",project="etl-test-shenzhen",
        fields=["__source__"]),field="processid",output_fields=["__source__"])
```

- Error message

```
error when calling : res_log_logstore_pull\nDetail: res_log_logstore_pull() missing 1 req
uired positional argument: 'logstore'", "requestId": "
```

- Error cause

  The error message `missing 1 required positional argument` is displayed because the required parameter `logstore` is missing.

- Troubleshooting method

  Check the orchestration syntax and specify the missing parameter.

Error 2: The delete_data parameter is not specified

- Transformation rule

```
e_table_map(res_log_logstore_pull(endpoint="xx",ak_id="xxx",
        ak_secret="xxx",project="etl-test-shenzhen",logstore="rds-mysql-test",
        fields=["__source__"],primary_keys="cid"),field="processid",output_fields=["__sou
rce__"])
```

- Error message

```
errorMessage\": \"when setting parameter primart_keys,need set delete_data\\nDetail: None
\
```

- Error cause

  This error occurs because the `delete_data` parameter is not specified when the `primary_keys` parameter is specified. The `primary_keys` and `delete_data` parameters must be set at the same time.

- Troubleshooting method

  Specify the primary_keys and delete_data parameters, and then restart the data transformation task.

Error 3: The target Logstore does not exist

- Transformation rule

```
e_table_map(res_log_logstore_pull(endpoint="xx",ak_id="xxx",
        ak_secret="xxx",project="etl-test-shenzhen",logstore="pull_logstore_test9900881",
        fields=["__source__"],primary_keys="cid"),field="processid",output_fields=["__sou
rce__"])
```

- Error message

```
message:  fetch data get errors:{"errorCode": "LogStoreNotExist", "errorMessage": "logsto
re pull_logstore_test9900881 does not exist", "requestId": "5D7227AA269948500404B777"},re
trytimes=2
```

- Error cause

  This error occurs because the target Logstore does not exist.

- Troubleshooting method

  Check the orchestration syntax and enter the correct Logstore name.

Error 4: The specified AccessKey pair is incorrect or invalid

- Transformation rule

```
e_table_map(res_log_logstore_pull(
        endpoint="cn-hangzhou.log.aliyuncs.com",
        ak_id="xx",
        ak_secret="xx",
        project="sls-test",
        logstore="pull_logstore_test",
        fields=[("id", "new_id"), "name", "status"],
        from_time="begin"), "name", "new_id")
```

- Error message

```
message:  fetch data get errors:{"errorCode": "SignatureNotMatch", "errorMessage": "signa
ture gdaL/nWSRtve5FOB+QqHO/sBdnA= not match", "requestId": "5D760261ED35D40AA4AB1953"},re
trytimes=1
```

```
message:  fetch data get errors:{"errorCode": "Unauthorized", "errorMessage": "AccessKeyI
d not found: xx", "requestId": "5D7602A01808F9EAA6EB0E2B"},retrytimes=3
```

- Error cause

  This error occurs because the ak_secret and ak_id parameters in the orchestration syntax do not match or the ak_id does not exist.

- Troubleshooting method

  Set the ak_secret and ak_id parameters correctly, and then restart the data transformation task.

Error 5: The current RAM user does not have sufficient permissions

- Transformation rule

```
e_table_map(res_log_logstore_pull(
        endpoint="cn-hangzhou.log.aliyuncs.com",
        ak_id="xx",
        ak_secret="xx",
        project="sls-test",
        logstore="pull_logstore_test",
        fields=[("id", "new_id"), "name", "status"],
        from_time="begin"), "name", "new_id")
```

- Error message

```
message:  fetch data get errors:{"errorCode": "Unauthorized", "errorMessage": "denied by
sts or ram, action: log:ListShards, resource: acs:log:cn-hangzhou:1654218965343050:...}
```

- Error cause

  The `Unauthorized` error message is displayed because the current RAM user does not have sufficient read and write permissions on the target Logstore.

- Troubleshooting method

  Grant sufficient permissions to the RAM user.

Error 6: The specified endpoint parameter is incorrect or invalid

- Transformation rule

```
e_table_map(res_log_logstore_pull(
        endpoint="xxx",
        ak_id="xx",
        ak_secret="xx",
        project="sls-test",
        logstore="pull_logstore_test",
        fields=[("id", "new_id"), "name", "status"],
        from_time="begin"), "name", "new_id")
```

- Error message

```
message:  fetch data get errors:{"errorCode": "ProjectNotExist", "errorMessage": "The Pro
ject does not exist : ali-sls-etl-regression-test", "requestId": "5D760AB12ECD0722AA1DD68
1"}
```

```
message:  fetch data get errors:{"errorCode": "LogRequestError", "errorMessage": "HTTPCon
nectionPool(host='ali-sls-etl-regression-test.xx', port=80): Max retries exceeded with ur
l: /logstores/pull_logstore_test/shards (Caused by NewConnectionError('<urllib3.connectio
n.HTTPConnection object at 0x7f968a298ef0>: Failed to establish a new connection: [Errno
-2] Name or service not known
```

- Error cause

  The `ProjectNotExist` or `LogRequestError` error message is displayed because the specified endpoint is incorrect or invalid or the required project does not exist in the specified endpoint.

- Troubleshooting method

  Correct the endpoint settings. For more information, see Endpoints.

16.9. How can I fix errors that occur during data pulls from OSS?

If the transformation rule of a Logstore also pulls data from Object Storage Service (OSS), data pull or update errors may occur. This topic introduces such errors and their corresponding troubleshooting methods.

After reading data from a Logstore, the data transformation engine starts to transform log events in the Logstore. If the transformation rule involves data pulls from external resources such as OSS, ApsaraDB for RDS, and other Logstores, data pull or update errors may occur.

Error impact

For more information, see Error impact.

Troubleshooting methods

For more information, see Solution.

Example errors

> **Note** In the following examples, AK_ID refers to AccessKey ID and AK_KEY refers to AccessKey secret.

- Incorrect object path or format

  A 404 error occurs if the "data" directory does not exist in OSS or the object format is incorrect.

  - Transformation rule

    ```
    # In this example, the correct directory is "test".
    e_set("oss_file",res_oss_file(endpoint, ak_id=res_local("AK_ID"),ak_key=res_local("AK_K
    EY"), bucket, 'data/test.txt', format='text', change_detect_interval=20))
    # The "text" format does not exist in the "test" directory.
    e_set("oss_file",res_oss_file(endpoint, ak_id=res_local("AK_ID"),ak_key=res_local("AK_K
    EY"), bucket, 'test/test.dat', format='text', change_detect_interval=20))
    ```

  - Error message

    ```
    message:  Exception: {'status': 404, 'x-oss-request-id': '5D49****878', 'details': {'Co
    de': 'NoSuchKey', 'Message': 'The specified key does not exist.', 'RequestId': '5D4***8
    878', 'HostId': 'lo***g.oss-cn-hangzhou.aliyuncs.com', 'Key': 'oss/test.txt'}}
    ```

  - Error cause

    This error occurs because the specified object path is incorrect. If a *404* error occurs and the error message involves the object, the object path is incorrect.

  - Troubleshooting method

    Correct the `file` settings in the orchestration syntax of the `res_oss_file` function.

    ```
    # In this example, the correct directory is "test".
    e_set("oss_file",res_oss_file(endpoint, ak_id=res_local("AK_ID"),ak_key=res_local("AK_K
    EY"), bucket, 'test/test.txt', format='text', change_detect_interval=20))
    ```

- Object decoding error

  A decoding error occurs if an object pulled from OSS fails to be decoded or a nonexistent decoding method is used.

  - Transformation rule

    ```
    e_set("oss_file",res_oss_file(endpoint, ak_id=res_local("AK_ID"),ak_key=res_local("AK_KEY"), bucket, 'test/test.txt', format='text', change_detect_interval=20, encoding='unicode'))
    ```

  - Error message

    ```
    {
      "reason": "LookupError: unknown encoding: unicode"
    }
    ```

  - Error cause

    This error occurs because the specified `encoding` is incorrect. If `unknown encoding: unicode` error message is reported, it means the specified parameter is incorrect.

  - Troubleshooting method

    Correct the `encoding` settings in the orchestration syntax of the `res_oss_file` function.

    ```
    # In this example, the decoding format is UTF-8.
    e_set("oss_file",res_oss_file(endpoint, ak_id=res_local("AK_ID"),ak_key=res_local("AK_KEY"), bucket, 'test/test.txt', format='text', change_detect_interval=20,encoding='utf8'))
    ```

- Endpoint errors

  An endpoint error occurs if the specified `endpoint` in the `res_oss_file` function is incorrect.

  - The specified endpoint does not exist.
    - Transformation rule

      ```
      e_set("oss_file",res_oss_file("https://oss-cn-asd.aliyuncs.com", ak_id=res_local("AK_ID"),ak_key=res_local("AK_KEY"), 'your bucket', 'test/test.txt', format='text', change_detect_interval=20))
      ```

    - Error message

      ```
      message:  get_oss_bytes get errors:{'status': -2, 'x-oss-request-id': '', 'details': "RequestError: HTTPSConnectionPool(host='log-etl-staging.oss-cn-asd.aliyuncs.com', port=443)
      ```

    - Error cause

      This error occurs because the specified endpoint does not exist. If the status value is `-2` and the error message involves `RequestError`, it means the specified endpoint is invalid.

    - Troubleshooting method

      Correct the `endpoint` settings in the orchestration syntax of the `res_oss_file` function.

- The specified OSS endpoint does not match the bucket name.

  In this example, the AccessKey pair owner has permissions to manage resources in the China (Hangzhou) region, but the specified endpoint corresponds to the China (Shanghai) region.

  - Transformation rule

    ```
    e_set("oss_file",res_oss_file("https://oss-cn-shanghai.aliyuncs.com", ak_id=res_local("AK_ID"),ak_key=res_local("AK_KEY"), 'your bucket', 'test/test.txt', format='text', change_detect_interval=20))
    ```

  - Error message

    ```
    message:  get_oss_bytes get errors:{'status': 403, 'x-oss-request-id': '5D7219353A90A2852B234D14', 'details': {'Code': 'AccessDenied', 'Message': 'The bucket you are attempting to access must be addressed using the specified endpoint. Please send all future requests to this endpoint.', 'RequestId': '5D7**14', 'HostId': 'log-**.oss-cn-shanghai.aliyuncs.com', 'Bucket': 'log-**', 'Endpoint': 'oss-cn-hangzhou.aliyuncs.com'}}
    ```

  - Error cause

    This error occurs because the specified `endpoint` and the AccessKey pair do not belong to the same region. If a 403 error occurs and the error message involves the specified bucket and endpoint, it means the endpoint and AccessKey pair belong to different regions.

  - Troubleshooting method

    Correct the `encoding` settings in the orchestration syntax of the `res_oss_file` function.

- Permission errors
  - The specified AccessKey ID or AccessKey secret is incorrect.

    In this example, the specified AccessKey ID in the `res_oss_file` function is incorrect.

    - Transformation rule

      ```
      e_set("oss_file",res_oss_file(endpoint, ak_id=res_local("AK_ID"),ak_key=res_local("AK_KEY"), 'your bucket', 'test/test.txt', format='text', change_detect_interval=20))
      ```

    - Error message

      ```
      message: Exception: {'status': 403, 'x-oss-request-id': '5D***BEB6', 'details': {'Code': 'InvalidAccessKeyId', 'Message': 'The OSS Access Key Id you provided does not exist in our records.', 'RequestId': '5D4***BEB6', 'HostId': 'lo***g.oss-cn-hangzhou.aliyuncs.com', 'OSSAccessKeyId': 'LT***Ai'}}
      ```

    - Error cause

      This error occurs because the specified AccessKey ID or AccessKey secret is incorrect. If a 403 error occurs and the error message involves the AccessKey ID or AccessKey secret, it means the AccessKey ID or AccessKey secret is incorrect.

    - Troubleshooting method

      Correct the AK_ID and AK_KEY settings in the orchestration syntax of the `res_oss_file` function.

- The AccessKey pair owner has no permission to access the specified bucket.

  In this example, a bucket access error occurs when the `res_oss_file` function is used.

  - Transformation rule

    ```
    e_set("oss_file",res_oss_file(endpoint='http://oss-cn-hangzhou.aliyuncs.com',ak_id=os
    .getenv("SLS_ACCESS_KEY_ID"),
                        ak_key=os.getenv("SLS_ACCESS_KEY_SECRET"),
                        bucket='log', file='test.txt',
                        change_detect_interval=30, encoding='utf8'))
    ```

  - Error message

    ```
    message: Exception: {'status': 403, 'x-oss-request-id': '5D674CE8BE0EBC45166026C5', '
    details': {'Code': 'AccessDenied', 'Message': 'You have no right to access this objec
    t because of bucket acl.', 'RequestId': '5D4***BEB6', 'HostId': 'log.oss-cn-hangzhou.
    aliyuncs.com'}}
    ```

  - Error cause

    This error occurs because the AccessKey pair owner has no permission to access the specified bucket. If a 403 error occurs and the error message involves the specified bucket, it means the AccessKey pair owner has no permission to access the bucket.

  - Troubleshooting method

    Correct the bucket settings in the orchestration syntax of the `res_oss_file` function.

- The specified bucket does not exist.

  In this example, a bucket error occurs when the `res_oss_file` function is used. The error occurs because the specified bucket does not exist in OSS.

  - Transformation rule

    ```
    e_set("oss_file",res_oss_file(endpoint='http://oss-cn-hangzhou.aliyuncs.com',ak_id=os
    .getenv("SLS_ACCESS_KEY_ID"),
                        ak_key=os.getenv("SLS_ACCESS_KEY_SECRET"),
                        bucket='twiss', file='test.txt',
                        change_detect_interval=30, encoding='utf8'))
    ```

  - Error message

    ```
    message: Exception: {'status': 404, 'x-oss-request-id': '5D75F6E9BB4097C678A381EF', '
    details': {'Code': 'NoSuchBucket', 'Message': 'The specified bucket does not exist.',
    'RequestId': '5D75F6E9BB4097C678A381EF', 'HostId': 'twiss.oss-cn-hangzhou.aliyuncs.co
    m', 'BucketName': 'twiss'}}
    ```

  - Error cause

    This error occurs because the specified bucket does not exist in OSS. If a 403 error occurs and the error message involves the bucket, it means the bucket does not exist.

  - Troubleshooting method

    Correct the bucket settings in the orchestration syntax of the `res_oss_file` function.

- Object update errors

The object update error messages are included in logs. The following examples describe three of the error messages.

- The following error message indicates that the first data pull from OSS fails due to network problems and the transformation rule retries the data pull.

```
{
  "reason":"Failed to pull data from oss for the first time and it is preparing to re-p
ull ..."
}
```

- The following error message indicates that data of the specified object is updated at regular intervals, but the resource function fails to pull new data from OSS and retries the data pull.

```
{
  "reason":"get_oss_byte get errors,begin retry ..."
}
```

- The following error message indicates that the retry fails and the resource function starts another retry. A maximum of three retries can be performed.

```
{
  "reason":"get_oss_byte get errors,refresh_interval ..."
}
```

16.10. How do I fix errors in the syntax used to load data from ApsaraDB RDS for MySQL?

If a transformation rule requires data to be loaded from ApsaraDB RDS for MySQL, errors may occur when the data is loaded or updated. This topic describes the errors and provides solutions.

After the data transformation engine reads data from a Logstore, the engine transforms the data. If the transformation rule requires data to be loaded from external resources such as Object Storage Service (OSS), ApsaraDB RDS, and other Logstores, errors may occur when the data is loaded or updated.

Incorrect use of resource functions in the Log Service console

- Transformation rule

```
res_rds_mysql(address="xx",username="xx",password="xx",database="xx")
```

- Error log

```
aliyun.log.logexception.LogException: {"errorCode": "InvalidEtlConfig", "errorMessage": "
ETL config doesn't pass security check, detail: invalid type detected: <class '_ast.Expr'
>", "requestId": ""}
```

- Troubleshooting

The error occurs because the syntax is invalid. The error is common when the resource function res_rds_mysql, res_log_logstore_pull, or res_oss_file function is independently used in the Log Service console.

- Solution

  Use the resource functions in combination with other functions such as e_table_map or e_search_table_map because the resource functions cannot be independently used in the Log Service console.

Invalid parameter settings

- Transformation rule

```
e_table_map(res_rds_mysql(address="xx",username="xx",password="xx",database="xx"),field="
processid",output_fields=["name","xixi"])
```

- Error log

```
"errorMessage\": \"When sql is not set, table must be set\\nDetail: None\"
```

- Troubleshooting

  Check whether the table or sql parameter is configured.

- Solution

  If the table parameter is not configured, you must specify an SQL statement to find the required table. If you do not specify the SQL statement in this situation, the required table cannot be found, and the data cannot be loaded. You must configure one of the table and sql parameters.

Invalid output fields

- Transformation rule

```
e_table_map(res_rds_mysql(address="x",username="xx",password="xx",database="xx",table="te
st"),field="processid",output_fields=["name","xixi"])
```

- Error log

```
"errorMessage\": \"trans_comp_lookup: output field xixi doesn't exist in lookup table\\nD
etail: None\
```

- Troubleshooting

  The fields in the output_fields parameter do not exist in the required table.

- Solution

  Confirm the fields that are contained in the table and replace the invalid fields in the output_fields parameter.

Invalid parameter type settings

- Transformation rule

```
e_table_map(res_rds_mysql(address="xxx",username=1234,password="xx",database="xx",table="
xx"),field="processid",output_fields=["name","xixi"])
```

- Error log

```
"errorMessage\": \"username not a string type\\nInvalid Settings
```

- Troubleshooting

  The value of the username parameter is not of the string type.

- Solution

  Specify a value of the string type.

Network or permission errors

- Transformation rule

```
e_table_map(res_rds_mysql(address="xxx",username=xxx,password="xx",database="test999",tab
le="xx"),field="processid",output_fields=["name","xixi"])
```

- Error log 1

```
message:  Database connection failed, cause: (1045, "Access denied for user 'root'@'192.0
.2.1' (using password: YES)")
```

- Error log 2

```
message:  Database connection failed, cause: (1049, "Unknown database 'test999')
```

- Troubleshooting

  Check whether the configurations are valid, the network is connected, or the required whitelist is configured on the required ApsaraDB RDS for MySQL instance. If a connection error occurs, the cause field in the error log contains the detailed cause of the error. You can troubleshoot the error based on the cause. Common errors include lack of permissions, incorrect passwords, and invalid addresses.

- Solution

  Reconfigure the functions and restart the data transformation task.

SQL syntax errors

- Transformation rule

```
e_table_map(res_rds_mysql(address="xxx",username=xxx,password="xx",database="xx",sql="ins
et into test values(1,"aini")",field="processid",output_fields=["name","xixi"]))
```

- Error log

```
"errorMessage\": \"The sql_query field only supports database query syntax\\nInvalid Sett
ings \\\"insert into test values(1,aini)\\\"\\nDetail: None\", \"requestId\": \"\"}
```

- Troubleshooting

  The SQL syntax is invalid. Check that the SQL syntax used to read data from and write data to databases is valid. If an error is found in the SQL syntax used to write data to databases, the error `fetch data error` may occur. In this situation, you must analyze the cause of the error.

- Solution

  Correct the SQL syntax. Resource functions support only SELECT.

Errors during continuous transformation

- Transformation rule

```
e_table_map(res_rds_mysql(address="xxx",username=xxx,password="xx",database="xx",table="t
est,field="processid",output_fields=["name","xixi"],refresh_interval=30))
```

- Error log

```
"errorMessage\": \"Database connection failed, cause: (2003, \\\"Can't connect to MySQL s
erver on 'rm-wz9z68i4itrk4v8d9yo.mysql.rds.aliyuncs.com' (timed out))
```

- Troubleshooting

  The error occurs because a specific whitelist of the required ApsaraDB RDS for MySQL instance is cleared. If an error such as a network interruption occurs during continuous transformation, the data transformation task is automatically retried. If an error such as revoked permissions or incorrectly deleted tables occurs, you must manually grant the permissions or restore the tables.

- Solution

  Check whether the tables or permissions are changed in the databases on the ApsaraDB RDS for MySQL instance based on the cause of the error.

16.11. How do I dynamically construct a field?

This topic describes how to dynamically construct fields, package existing logs as a whole, and add these logs to the new fields.

Example: The first function in the following transformation rule renames the content field in the following raw log entry to k1_content_copy, the name field to k2_name_copy, and the School field to k3_school_copy. The second function constructs a field named __extract_others__, adds the field to the transformed log entry, and drops the k1_content_copy and k3_school_copy fields.

- Transformation rule:

```
e_set("k1_content_copy", v("content"), "k2_name_copy", v("name"), "k3_school_copy", v("Sc
hool"))
e_set("__extract_others__", dct_delete(KEEP,"k1_content_copy","k3_school_copy"))
```

- Raw log entry:

```
School: CMU
__source__: 192.168.1.1
__tag__:__client_ip__: 192.168.1.2
_tag__:__receive_time__: 1591755799
__topic__:
content: test concent
name: Twish
```

- Result:

```
School:  CMU
__extract_others__:  {"__time__": "1591755799", "__topic__": "", "__source__": "192.168.1
.1", "__tag__:__client_ip__": "192.168.1.2", "__tag__:__receive_time__": "1591755799", "c
ontent": "test content", "name": "Twish", "School": "CMU", "k2_name_copy": "Twish"}
__source__:  192.168.1.1
__tag__:__client_ip__:  192.168.1.2
__tag__:__receive_time__:  1591755799
__topic__:
content:  test content
k2_name_copy:  Twish
name:  Twish
```

16.12. How do I send a log entry to different storage targets with different field sets?

To send a log entry to different storage targets with different field sets, use the following example transformation rule. In this example, the fields in the raw log entry are f1, f2, f3, f4, and f5. The f1 and f2 fields are dropped when the raw log entry is sent to storage target 1. The f3 and f4 fields are dropped when the raw log entry is sent to storage target 2.

- Raw log entry:

```
__time__ : 1591754815
f1: GET
f2: https
f3: aliyun
f4: 200
f5: standard
```

- Transformation rule:

```
e_set("tag", "target1, target2")
e_split("tag")
e_if(e_search("tag==target1"), e_compose(e_drop_fields("f1", "f2", regex=False), e_output
("target1")))
e_drop_fields("f3", "f4", regex=False)
e_output("target2")
```

- Result:
  - Storage target 1:

```
__time__ : 1591754815
f3: aliyun
f4: 200
f5: standard
```

- Storage target 2:

```
__time__ : 1591754815
f1: GET
f2: https
f5: standard
```

If you use the following transformation rule, the log entry is sent to storage target 1 as expected. However, the f1 and f2 fields are dropped when the raw log entry is sent to storage target 2.

```
e_drop_fields("f1", "f2", regex=False)
e_coutput("target1")
e_drop_fields("f3", "f4", regex=False)
e_output("target2")
```

16.13. What do I do if the destination Logstores contain no data?

After you distribute the transformed data to the destination Logstores, the Logstores may contain no data. This topic describes how to troubleshoot the issue in this situation.

Scenario 1: The storage destinations specified in the transformation statements are inconsistent with those configured in the Create Data Transformation Rule panel

The source Logstore is website_log. It contains 1,000 log entries whose SourceIP is 192.0.2.54, 1,000 log entries whose SourceIP is 192.0.2.28, 1,000 log entries whose SourceIP is 192.0.2.136, and 2,000 log entries whose SourceIP is different from these IP addresses. The log entries are transformed. Then, the log entries that meet specific conditions are distributed to the following destination Logstores: 54_log_target, 28_log_target, and 136_log_target.

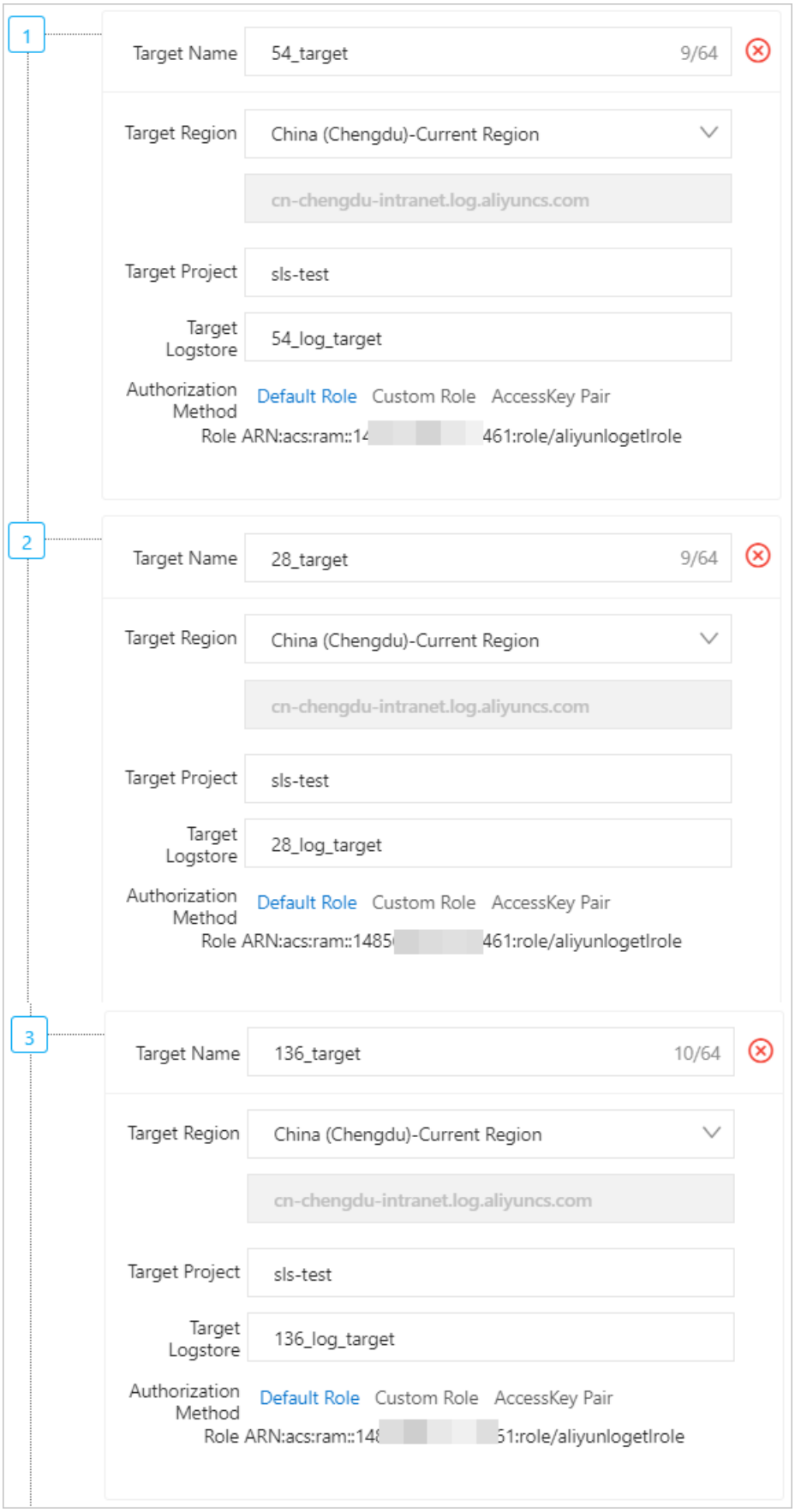
- Transformation requirements
  - Distribute the log entries whose SourceIP is 192.0.2.54 to the 54_log_target Logstore.
  - Distribute the log entries whose SourceIP is 192.0.2.28 to the 28_log_target Logstore.
  - Distribute the log entries whose SourceIP is 192.0.2.136 to the 136_log_target Logstore.
  - Discard all other log entries.
- Transformation statements

  The three destination Logstores are 54_log, 28_log, and 136_log.

```
e_if(e_search("SourceIP==192.0.2.54"),
  e_output(name="54-target",
             project="sls-test",
             logstore="54_log"))
e_if(e_search("SourceIP==192.0.2.28"),
    e_output(name="28-target",
             project="sls-test",
             logstore="28_log"))
e_if(e_search("SourceIP==192.0.2.136"),
    e_output(name="136-target",
             project="sls-test",
             logstore="136_log"))
e_drop()
```

- Storage Target

  The three destination Logstores are 54_log_target, 28_log_target, and 136_log_target.



- Transformation results

The 54_log_target, 28_log_target, and 136_log_target destination Logstores contain no data.

- Cause

  In the e_output functions of the transformation statements, the project and logstore parameters are set differently from **Destination Project** and **Destination Logstore** in the **Create Data Transformation Rule** panel. The project and logstore parameters specify the destination project and Logstores in transformation statements. The transformation statements are domain-specific language (DSL) statements. If the DSL statements use different parameter settings from those specified in the **Create Data Transformation Rule** panel, the parameter settings in the DSL statements overwrite those in the **Create Data Transformation Rule** panel. In this situation, only the authorization settings for the storage destinations in the panel are used for data transformation. As a result, the log entries are distributed to the 54_log, 28_log, and 136_log destination Logstores.

  > **Notice** If you use the e_output or e_coutput function, make sure that the storage destinations are consistent between the functions and the **Create Data Transformation Rule** panel. For more information, see e_output and e_coutput.

- Solution

  Modify the DSL statements to make sure that the destination Logstores are consistent between the statements and the **Create Data Transformation Rule** panel.

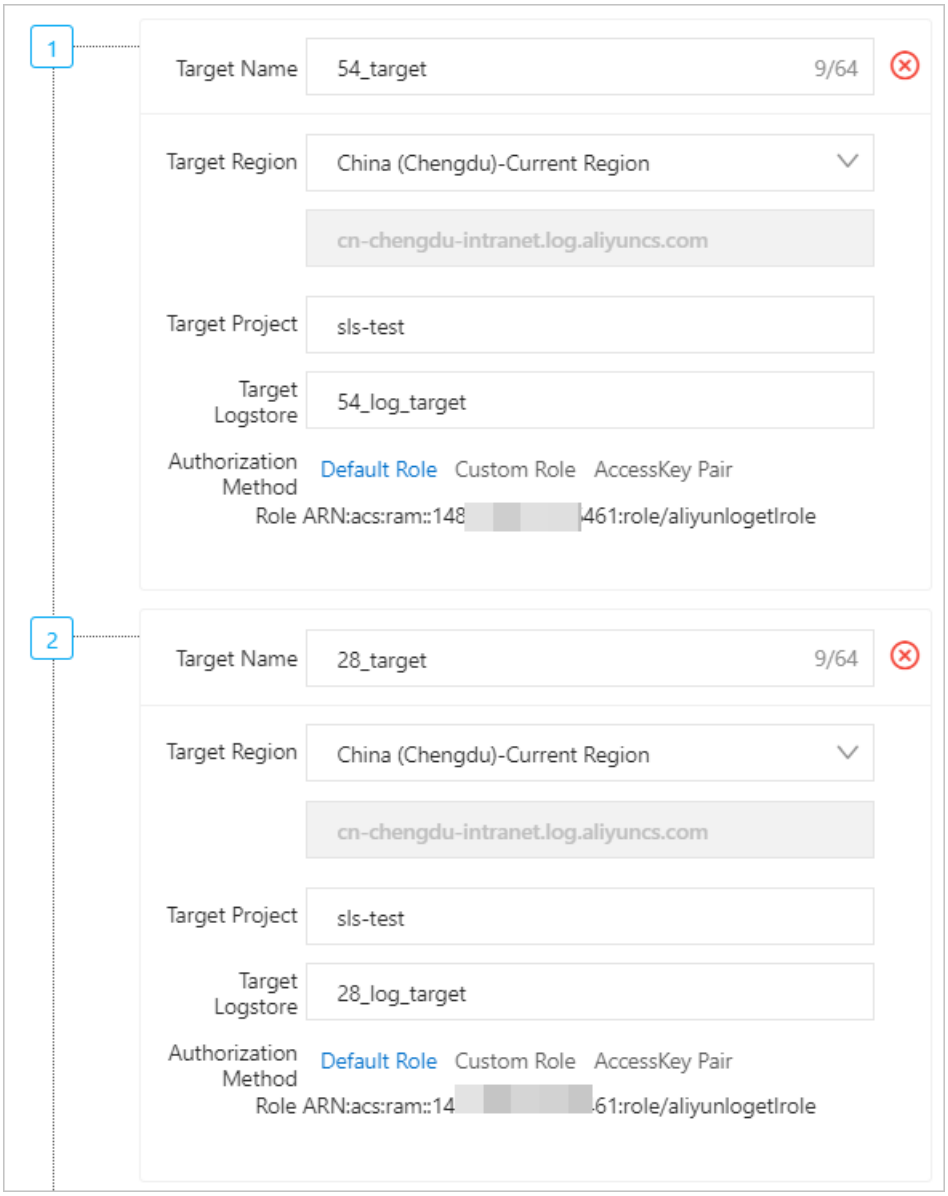
```
e_if(e_search("SourceIP==192.0.2.54"),
  e_output(name="54-target",
            project="sls-test",
            logstore="54_log_target"))
e_if(e_search("SourceIP==192.0.2.28"),
    e_output(name="28-target",
            project="sls-test",
            logstore="28_log_target"))
e_if(e_search("SourceIP==192.0.2.136"),
    e_output(name="136-target",
            project="sls-test",
            logstore="136_log_target"))
e_drop()
```

Scenario 2: No transformation statements are entered

- Transformation statements

  None.

- Storage Target



- Transformation results

  Only the 54_log_target destination Logstore contains data.

- Cause

  No transformation statements are entered, and multiple storage destinations are configured in the **Create Data Transformation Rule** panel. In this situation, all log entries are distributed to the destination Logstore in the storage destination that is numbered 1. By default, the storage destination that is numbered 1 is used.

Note

- If transformation statements are entered without the e_drop() function and multiple storage destinations are configured, the system distributes all the log entries that do not meet the conditions in the statements and are not discarded to the destination Logstore in the storage destination that is numbered 1.
- If no transformation statements are entered and multiple storage destinations are configured, the system distributes all log entries to the destination Logstore in the storage destination that is numbered 1. Other storage destinations are not used.
- If transformation statements are entered, the statements include only the e_drop() function, and a single storage destination is configured, the system discards all log entries. The destination Logstore contains no data.
- If no transformation statements are entered and a single storage destination is configured, the system only replicates the log entries to the destination Logstore.

Scenario 3: Data transformation encounters latency

If the settings in the transformation rule are correct but the destination Logstores contain no data, data transformation may encounter latency.

In this situation, we recommend that you perform the following operations:

- Modify the numbers of shards in the source and destination Logstores. For more information, see Performance guide.
- Optimize the logic of the transformation statements. For example, optimize regular expressions.

16.14. What do I do if the destination Logstores contain unexpected data?

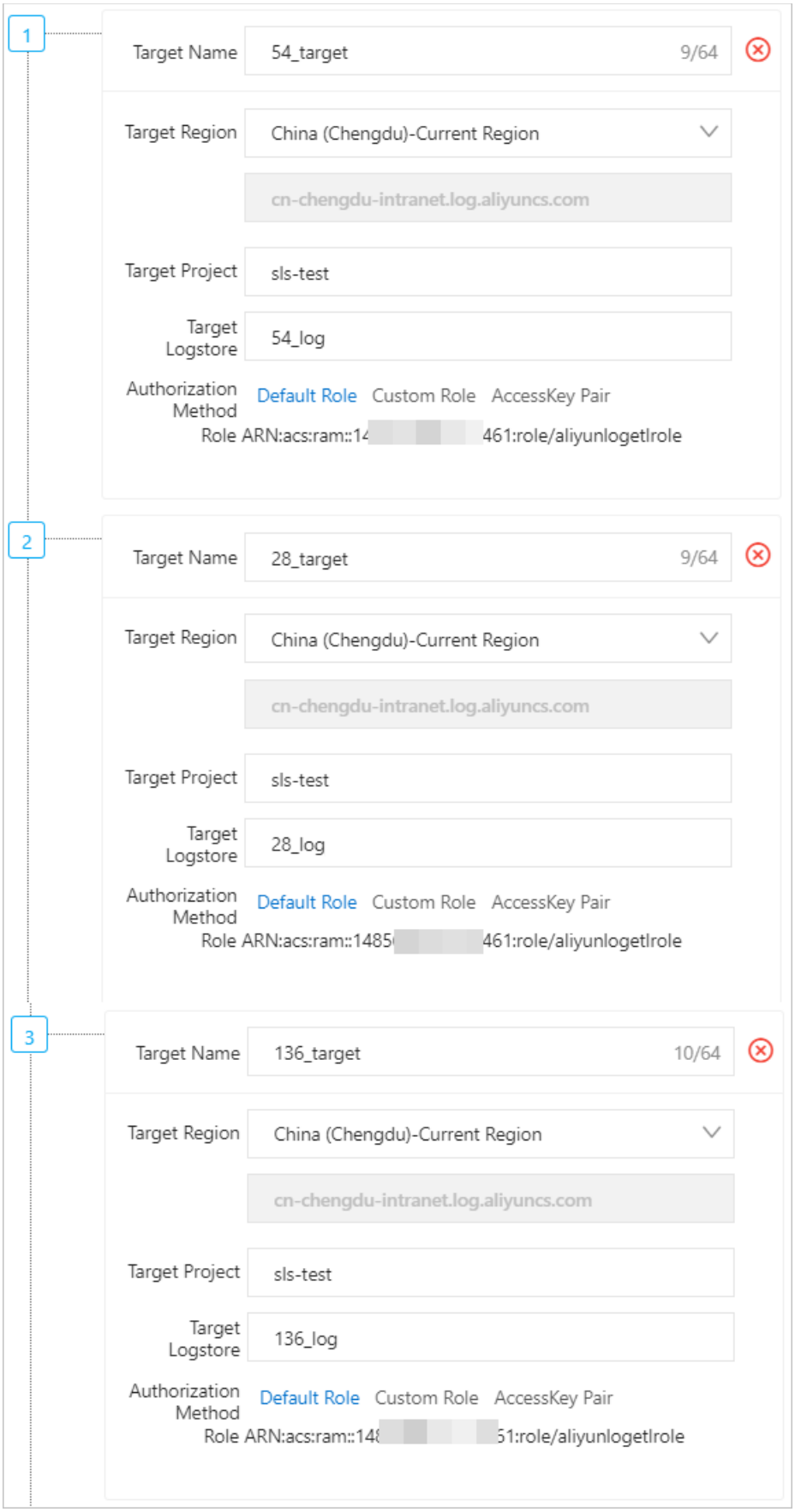
After you distribute the transformed data to the destination Logstores, the Logstores may contain unexpected data. This topic describes how to troubleshoot the issue in this situation.

The source Logstore is website_log. It contains 1,000 log entries whose SourceIP is 192.0.2.54, 1,000 log entries whose SourceIP is 192.0.2.28, 1,000 log entries whose SourceIP is 192.0.2.136, and 2,000 log entries whose SourceIP is different from these IP addresses. The log entries are transformed. Then, the log entries that meet specific conditions are distributed to different destination Logstores.

- Transformation requirements
  - Distribute the log entries whose SourceIP is 192.0.2.54 to the 54_log Logstore.
  - Distribute the log entries whose SourceIP is 192.0.2.28 to the 28_log Logstore.
  - Distribute the log entries whose SourceIP is 192.0.2.136 to the 136_log Logstore.
- Expected results
  - The 54_log Logstore contains 1,000 log entries whose SourceIP is 192.0.2.54.
  - The 28_log Logstore contains 1,000 log entries whose SourceIP is 192.0.2.28.
  - The 136_log Logstore contains 1,000 log entries whose SourceIP is 192.0.2.136.
- Transformation statements

```
e_if(e_search("SourceIP==192.0.2.54"),
  e_output(name="54-target",
             project="sls-test",
             logstore="54_log"))
e_if(e_search("SourceIP==192.0.2.28"),
    e_output(name="28-target",
             project="sls-test",
             logstore="28_log"))
e_if(e_search("SourceIP==192.0.2.136"),
    e_output(name="136-target",
             project="sls-test",
             logstore="136_log"))
```

- Storage Target



- Transformation results

- The 54_log Logstore contains 3,000 log entries whose SourceIP is 192.0.2.54 or an IP address other than 192.0.2.26 and 192.0.2.136. This does not meet the expectation.
- The 28_log Logstore contains 1,000 log entries whose SourceIP is 192.0.2.28. This meets the expectation.
- The 136_log Logstore contains 1,000 log entries whose SourceIP is 192.0.2.136. This meets the expectation.

- Cause

  When Log Service distributes the log entries after transformation, the log entries that meet the condition for each e_output function are distributed to the specified destination Logstore. The log entries that do not meet the conditions and are not discarded are all distributed to the destination Logstore in the storage destination that is numbered 1. In this example, these log entries are distributed to the 54_log Logstore. By default, the storage destination that is numbered 1 is used.

- Solution

  Include the e_drop() function in the transformation statements to discard the log entries that do not meet the conditions specified in the statements.

```
e_if(e_search("SourceIP==192.0.2.54"),
  e_output(name="54-target",
             project="sls-test",
             logstore="54_log"))
e_if(e_search("SourceIP==192.0.2.28"),
    e_output(name="28-target",
             project="sls-test",
             logstore="28_log"))
e_if(e_search("SourceIP==192.0.2.136"),
    e_output(name="136-target",
             project="sls-test",
             logstore="136_log"))
e_drop()
```

16.15. What do I do if I cannot use context query in a destination Logstore?

After you distribute transformed data to the destination Logstore, you may fail to use context query in the Logstore. To use context query in this situation, you must disable the filtering of context information in the configuration of the data transformation task.

- Cause

  When fields are added to or deleted from raw logs during data transformation, log groups are reassembled, and the context information of logs in the log groups may become invalid. In this case, Log Service filters out all context information of logs. As a result, you cannot use context query in the destination Logstore.

- Solution

When you create a data transformation task or modify an existing data transformation task, specify `system.control.filter_pack_id:false` in **Advanced Parameter Settings** to disable the filtering of context information.

- Parameter description: specifies whether to filter out the context information of logs.
- Valid values:
  - true: This is the default value. If you use this value, Log Service filters out the context information, and you cannot use context query in the destination Logstore.
  - false: If you use this value, Log Service tries to retain context information as it was, and you can use context query in the destination Logstore. If a data transformation task is used only to replicate data, the context information of raw logs is retained as it was.

For more information, see Context query.