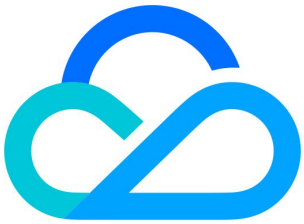
日志服务

API 文档

产品文档



腾讯云

文档目录

API 文档
请求签名

最近更新时间：2020-09-08 10:39:56

准备工作

1. 获取 SecretId 和 SecretKey。
   请在控制台 云 API 密钥 页面可获取。
2. 确定开发语言：
   根据不同开发语言，确定对应的 HMAC-SHA1 函数。日志服务 CLS 提供 C#，C++，Go，Java，Node.js，PHP，Python 开发语言的 签名计算 demo 供用户参考。

通过 API 对 CLS 发起的 HTTP 签名请求，使用标准的 HTTP Authorization 头部来传递，如下例所示：

```
GET /logset?logset_id=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx HTTP/1.1
Host: ap-shanghai.cls.tencentyun.com
Authorization: q-sign-algorithm=sha1&q-ak=AKIDc9YlmrBcFk4C8sbmXQ8i65XXXXXXXXXX&q-sign-time
=1510109254;1510109314&q-key-time=1510109254;1510109314&q-header-list=content-type;host&q-
url-param-list=logset_name&q-signature=e8b23b818caf4e33f196f895218bdabdbd1f1423
```

内外网域名

日志服务请求域名区分内外网形式：

- 内网域名格式形如${region}.cls.tencentyun.com ，内网域名仅对同地域访问生效，即云服务器或云产品通过内网域名访问相同地域的日志服务。
- 外网域名格式形如${region}.cls.tencentcs.com，访问源端接入 Internet 网后，正常情况均能访问日志服务外网域名。

region 字段是日志服务区域简称，例如北京区域填写 ap-beijing ，完整区域列表格式参考 地域列表。

```
ap-beijing - 北京
ap-shanghai - 上海
ap-guangzhou - 广州
ap-chengdu - 成都
...
```

键值描述

请求中的签名内容由多个 key=value 对，通过 “&” 连接而成，格式如下：

```
q-sign-algorithm=[Algorithm]&q-ak=[SecretId]&q-sign-time=[SignTime]&q-key-time=[KeyTime]&q-header-list=[SignedHeaderList]&q-url-param-list=[SignedParamList]&q-signature=[Signature]
```

上述组成签名内容的键值（Key=Value）描述如下：

键（Key）	值（Value）	含义
q-sign-algorithm	sha1	必需，签名算法，目前仅支持 sha1
q-ak	参数[SecretId]	必需，账户 API 密钥的 SecretId
q-sign-time	参数[SignTime]	必需，签名有效起止时间，Unix 时间戳，以秒为单位，使用;分隔。例如：1510109254;1510109314
q-key-time	参数[KeyTime]	必需，与 q-sign-time 值相同
q-header-list	参数[SignedHeaderList]	必需，需要加入签名的 Http 请求头部的 key，且 key 需转化为小写，并将多个 key 之间以字典顺序排序，例如有多组 key，用;分隔。若不想对任何 header 进行签名，可以写空字符串。
q-url-param-list	参数[SignedParamList]	必需，需要加入签名的 Http 请求 Uri 部分参数，且 key 需转化为小写，并将多个 key 之间以字典顺序排序，如有多组 key，用;分隔。若不想对任何 param 进行签名，可以写空字符串。
q-signature	参数[Signature]	必需，计算出的签名内容，小写字母

> **注意：**
> 关于 q-sign-time 和 q-key-time， 结束时间要大于起始时间，否则会导致签名马上过期。

计算方法

Signature 计算步骤：

1. 按照指定格式拼接 HTTP 请求的相关信息为字符串 HttpRequestInfo。
2. 对 HttpRequestInfo 使用 sha1 算法计算哈希值，与其他指定参数按照指定格式组成签名原串 StringToSign。
3. 使用 SecretKey 对 q-key-time 进行加密，得到 SignKey。
4. 使用 SignKey 对 StringToSign 进行加密，生成 Signature。

> **注意：**
> 特殊字符的 urlencode 结果要使用大写字符，例如 / 编码为 `%2F` ，不能使用 `%2f` 。

步骤1: 拼接 HttpRequestInfo

HttpRequestInfo 由 HTTP 请求中的 Method，Uri，Headers，Parameters 组成，具体方式如下：

```
HttpRequestInfo = Method + "\n"
+ Uri + "\n"
+ FormatedParameters + "\n"
+ FormatedHeaders + "\n"
```

上述中的 \n 表示换行转义字符，+表示字符串连接操作，其他各个参数定义如下：

字段名	含义
Method	HTTP 请求使用的方法，小写字母，例如get、post等
Uri	HTTP 请求的资源名称，不包含 query string 部分，例如/logset
FormatedParameters	HTTP 请求 query string 部分参数序列化的字符串，即 q-url-param-list 中指定的参数，若无指定，则使用空字符串。key 和 value 以=连接，不同键值对以&连接，需要以字典序排列，key 为小写字母，value 需要做 urlencode
FormatedHeaders	HTTP 请求的 header，即 q-header-list 中指定的 HTTP 头部，如无指定，则使用空字符串，key 和 value 以=连接，不同键值对以&连接， 需要以字典序排列，key 为小写字母，value 需要做 urlencode

获取 logset 信息，HTTP 请求如下：

```
GET /logset?logset_id=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx HTTP/1.1
Host: ap-shanghai.cls.tencentyun.com
```

对应的 HttpRequestInfo 如下：

有请求参数：

```
get\n/logset\nlogset_id=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx\nhost=ap-shanghai.cls.tencentyun.com\n
```

无请求参数：

```
get\n/logset\n\nhost=ap-shanghai.cls.tencentyun.com\n
```

> **注意：**
> 您需事先将 \n 转义为换行符，再对 HttpRequestInfo 进行 sha1 计算。

```
StringToSign = sha1\n1578973108;1578974918\n7be58ef9a64ecca66f96b79dc70d279bd93915cf\n
```

步骤3: 生成 SignKey

目前，API 只支持一种数字签名算法，即默认签名算法 hmac-sha1。其伪代码如下：

```
SignKey = Hexdigest(HMAC-SHA1(q-key-time, SecretKey))
```

其中 HMAC-SHA1 为加密算法， Hexdigest 为转换为16进制字符串的方法。有些语言的加密算法输出结果直接为16进制字符串，则无需转换。

对应的如下：

```
SignKey = Hexdigest(HMAC-SHA1(1578973108;1578974918, LUSE4nPK1d4tX5SHyXv6tZXXXXXXXXXX))
```

步骤4: 生成 Signature

目前，API 只支持一种数字签名算法，即默认签名算法 hmac-sha1。其伪代码如下：

```
Signature = Hexdigest(HMAC-SHA1(StringToSign, SignKey))
```

其中 HMAC-SHA1 为加密算法， Hexdigest 为转换为16进制字符串的方法。有些语言的加密算法输出结果直接为16进制字符串，则无需转换。

对应的签名如下：

```
Signature = Hexdigest(HMAC-SHA1(sha1\n1578973108;1578974918\n7be58ef9a64ecca66f96b79dc70d279bd93915cf\n, 100edfdb73b873dae3d94665a2a7505258475486))
```

举例说明

使用如下 SecretId 和 SecretKey 进行签名说明：

```
SecretId = "AKIDc9YlmrBcFk4C8sbmXQ8i65XXXXXXXXXX"
SecretKey = "LUSE4nPK1d4tX5SHyXv6tZXXXXXXXXXX"

StartTime = 1578976553
EndTime = 1578978363
```

例一：
获取 logset 信息，HTTP 请求如下：

```
GET /logset?logset_id=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx HTTP/1.1
Host: ap-shanghai.cls.tencentyun.com
Content-Type: application/json
```

对如上请求，请求头 Host 如果加入签名则生成的字符串为：

```
HttpRequestInfo=get\n/logset\nlogset_id=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx\ncontent-type=application%2Fjson&host=ap-shanghai.cls.tencentyun.com\n
```

根据 HttpRequestInfo 组成签名原串 为：

```
StringToSign = sha1\n1578976553;1578978363\ne2d0126b61269ef047d9d05b6c385cea0aea9799\n
```

使用 SecretKey 对 q-key-time 进行加密，得到为：

```
SignKey = f49255658de17084898d83beaa755b9f0301591f
```

使用 SignKey 对 StringToSign 进行加密，生成：

```
Signature = 315dfa0d0ce55582145f7800df5eb3e9c88d2f84
```

拼接最后的签名为：

```
Authorization = q-sign-algorithm=sha1&q-ak=AKIDc9YlmrBcFk4C8sbmXQ8i65XXXXXXXXXX&q-sign-time=1578976553;1578978363&q-key-time=1578976553;1578978363&q-header-list=content-type;host&q-url-param-list=logset_id&q-signature=315dfa0d0ce55582145f7800df5eb3e9c88d2f84
```

最终的请求内容为：

```
GET /logset?logset_id=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx HTTP/1.1
Host: ap-shanghai.cls.tencentyun.com
Content-Type: application/json
Authorization: q-sign-algorithm=sha1&q-ak=AKIDc9YlmrBcFk4C8sbmXQ8i65XXXXXXXXXX&q-sign-time=1578976553;1578978363&q-key-time=1578976553;1578978363&q-header-list=content-type;host&q-url-param-list=logset_id&q-signature=315dfa0d0ce55582145f7800df5eb3e9c88d2f84
```

例二：

修改 logset 信息，HTTP 请求如下：

```
PUT /logset HTTP/1.1
Host: ap-shanghai.cls.tencentyun.com
Content-Type: application/json
Content-Length: 50

{"logset_id":"xxxx-xx-xx-xx-xxxxxxxx","period":30}
```

对如上请求，请求头 Host 如果加入签名则生成的字符串为：

```
HttpRequestInfo = put\n/logset\n\ncontent-type=application%2Fjson&host=ap-shanghai.cls.tencentyun.com\n
```

> **说明：**
> uri 参数为空，所以是为空字符，但是不能省去 \n ， 所以生成 \n\n 。

根据 sha1(HttpRequestInfo) 组成签名原串为：

```
StringToSign = sha1\n1578976553;1578978363\ne86af9693f3de2047dd10dbe2898ecaf1df00de0\n
```

使用 SecretKey 对 q-key-time 进行加密，得到：

```
SignKey = f49255658de17084898d83beaa755b9f0301591f
```

使用 SignKey 对 StringToSign 进行加密，生成：

```
Signature = 600aeb5e646d385d7dd9da57ba9b2545cadfaa1c
```

拼接最后的签名为：

```
Authorization = q-sign-algorithm=sha1&q-ak=AKIDc9YlmrBcFk4C8sbmXQ8i65XXXXXXXXXX&q-sign-time=1578976553;1578978363&q-key-time=1578976553;1578978363&q-header-list=content-type;host&q-url-param-list=&q-signature=600aeb5e646d385d7dd9da57ba9b2545cadfaa1c
```

最终的请求内容为：

```
PUT /logset HTTP/1.1
Host: ap-shanghai.cls.tencentyun.com
Content-Type: application/json
Content-Length: 50
Authorization: q-sign-algorithm=sha1&q-ak=AKIDc9YlmrBcFk4C8sbmXQ8i65XXXXXXXXXX&q-sign-time=1578976553;1578978363&q-key-time=1578976553;1578978363&q-header-list=content-type;host&q-url-param-list=&q-signature=600aeb5e646d385d7dd9da57ba9b2545cadfaa1c
{"logset_id":"xxxx-xx-xx-xx-xxxxxxxx","period":30}
```

API 概览

最近更新时间：2020-08-10 15:29:51

日志管理

接口名称	功能描述
上传结构化日志	上传日志到指定的日志主题
获取下载日志游标	获取并下载指定日志主题下的日志游标
搜索日志	根据指定的条件搜索日志内容
下载日志	使用游标下载日志

日志集管理

接口名称	功能描述
创建日志集	创建日志集，返回新日志集 ID
获取日志集信息	获取日志集信息
获取日志集列表	获取日志集信息列表
修改日志集	修改日志集
删除日志集	删除日志集

日志主题管理

接口名称	功能描述
创建日志主题	创建日志主题，返回新日志主题 ID
获取日志主题信息	获取日志主题信息
获取日志主题绑定的机器组	获取日志主题绑定的机器组信息
获取日志主题列表	获取日志主题信息列表
修改日志主题	修改日志主题
设置日志主题绑定的机器组	设置日志主题绑定的机器组信息
删除日志主题	删除日志主题

接口名称	功能描述
获取主题分区列表	获取主题分区信息列表
合并主题分区	合并一个读写态的主题分区
分裂主题分区	分裂一个读写态的主题分区

投递任务管理

接口名称	功能描述
创建投递任务	创建新的投递任务
获取投递配置	获取指定投递策略的详细信息
获取日志主题投递列表	获取指定日志主题的投递策略详细列表
获取投递任务列表	获取投递任务信息列表
修改投递任务	修改现有的投递任务
重试失败的任务	重试失败的投递任务
删除投递配置	删除投递配置

机器组管理

接口名称	功能描述
创建机器组	创建机器组，返回新机器组 ID
获取机器组信息	获取机器组信息
获取机器状态	获取指定机器组下的机器状态
获取机器组列表	获取机器组信息列表
修改机器组	修改机器组
删除机器组	删除机器组

消费管理

接口名称	功能描述
创建消费组	创建消费组

接口名称	功能描述
获取消费游标	获取消费游标
获取消费组游标	获取消费组游标
消费数据	用于消费读取日志
消费者心跳	用于消费者上传心跳
获取消费组列表	获取日志主题的消费组列表
修改消费组	修改消费组
修改消费组游标	修改消费组游标信息
删除消费组	删除消费组

索引管理

接口名称	功能描述
获取索引信息	获取指定索引策略的详细信息
修改索引任务	修改现有的索引任务

公共请求头部

最近更新时间：2020-05-27 18:23:27

概述

此文将介绍在使用 CLS API 时需要用到的公共请求头部（Request Header），下文提到的头部在之后具体 API 文档中不再赘述。

公共请求头部列表

HTTP Header 名称	描述
Host	请求的 host 名字，每个地域不同，以上海为例：ap-shanghai.cls.tencentyun.com
Authorization	签名内容，计算方式请参考 请求签名 文档
Content-Length	请求的 Body 长度，如果无 Body，可以不提供此头部
Content-Type	请求的 Body 格式，如果无 Body ，可以不提供此头部，根据接口详细文档确定，目前支持 application/json 、 application/x-protobuf
Content-MD5	请求的 Body 对应的 MD5 值，如果无 Body，可以不提供此头部，计算结果为小写字母
x-cls-compress-type	请求的 Body 使用的压缩方法，目前支持 lz4 压缩，只有上传日志接口需要，如果没有压缩则可以不提供此头部
x-cls-token	临时密钥凭证，请求 STS 返回的临时密钥 Token 信息。通过临时密钥访问日志服务时，必须携带此信息

公共响应头部

最近更新时间：2020-08-10 15:56:39

概述

本文将为您介绍在使用 CLS API 时的公共响应头部（Response Header），下文提到的头部在之后 API 文档中不再赘述。

响应头部列表

响应头的说明如下：

HTTP Header 名称	描述
Content-Length	响应的 Body 长度
Content-Type	响应的 Body 格式，目前支持 application/json
x-cls-requestid	服务端针对本次请求标示的唯一 ID

日志管理
上传结构化日志

最近更新时间：2020-10-28 10:02:00

功能描述

本接口用于将日志写入到指定的日志主题。

日志服务提供以下两种模式：

负载均衡模式

系统根据当前日志主题下所有可读写的分区，遵循负载均衡原则自动分配写入的目标分区。该模式适合消费不保序的场景。

示例

```
POST /structuredlog?topic_id=xxxxxxxx-xxxx-xxxx-xxxx HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
Content-Type: application/x-protobuf

<LogGroupList 的 PB 格式打包内容>
```

哈希路由模式

系统根据携带的哈希值（x-cls-hashkey）将数据写入到符合范围要求的目标分区。例如，可以将某个日志源端通过 hashkey 与某个主题分区强绑定，这样可以保证数据在该分区上写入和消费是严格保序的。

示例

```
POST /structuredlog?topic_id=xxxxxxxx-xxxx-xxxx-xxxx HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
Content-Type: application/x-protobuf
x-cls-hashkey: xxxxxxxxxxxxxxxxxxxxxxxx

<LogGroupList 的 PB 格式打包内容>
```

> 说明：
> PB 描述文件格式以及编译步骤，请参阅 PB 编译示例。

此外日志服务还为用户提供以下两种不同的日志上传模式：

压缩上传模式

使用压缩上传模式时，用户日志将在采集端先以 lz4 格式压缩，之后再将压缩完成后的日志数据上传到日志存储端。通过使用压缩上传模式，用户可以减少日志上传流量（日志写流量），从而节省费用。

示例

```
POST /structuredlog?topic_id=xxxxxxxx-xxxx-xxxx-xxxx HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
Content-Type: application/x-protobuf
x-cls-compress-type:lz4

<LogGroupList 的 PB 格式压缩包内容>
```

非压缩上传模式

使用非压缩上传模式时，用户日志将以原始大小上传，相比于压缩上传模式，非压缩模式将会产生更高的日志写流量费用。

示例

```
POST /structuredlog?topic_id=xxxxxxxx-xxxx-xxxx-xxxx HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
Content-Type: application/x-protobuf

<LogGroupList 的 PB 格式打包内容>
```

请求

请求行

```
POST /structuredlog
```

请求头

x-cls-hashkey 请求头表示日志根据 hashkey 路由写入到 CLS 对应范围的主题分区，以保证写入到的每个主题分区是严格有序的，便于消费时有序消费。

字段名	类型	位置	是否必须	含义
x-cls-hashkey	string	header	否	根据 hashkey 写入相应范围的主题分区

字段名	类型	位置	是否必须	含义

请求参数

字段名	类型	位置	必须	含义
topic_id	string	query	是	日志主题 ID ，表示数据上传的目标日志主题，可在 日志主题列表 查看日志主题 ID
logGroupList	message	pb	是	logGroup 列表，封装好的日志组列表内容，建议 logGroup 数量不要超过5个

LogGroup 说明：

字段名	是否必选	含义
logs	是	日志数组，表示有多个 Log 组成的集合，一个 Log 表示一条日志，一个 LogGroup 中 Log 个数不能超过10000
contextFlow	否	保持上下文的 UID，该字段目前暂无效用
filename	否	日志文件名
source	否	日志来源，一般使用机器 IP 作为标识
logTags	否	日志的标签列表

Log 说明：

字段名	是否必选	含义
time	是	日志时间（Unix 格式时间戳），支持秒、毫秒，建议采用毫秒
contents	否	key-value 格式的日志内容，表示一条日志里的多个 key-value 组合

Content 说明：

字段名	是否必选	含义
key	是	单条日志里某个字段组的 key 值，不能以_开头
value	是	单条日志某个字段组的 value 值，单条日志 value 不能超过1MB，LogGroup 中所有 value 总和不能超过5MB

LogTag 说明：

字段名	是否必选	含义
key	是	自定义的标签 key

字段名	是否必选	含义
value	是	自定义的标签 key 对应的 value 值

响应

响应示例

```
HTTP/1.1 200 OK
Content-Length: 0
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

无。

错误码

参见 错误码。

PB 编译示例

本示例将说明如何使用官方 protoc 编译工具将 PB 描述文件 编译生成为 C++ 语言可调用的上传日志接口。

> 说明：
> 目前 protoc 官方支持 Java、C++、Python 等语言的编译，详情请参见 protoc。

1. 安装 Protocol Buffer

下载 Protocol Buffer ，解压并安装。示例版本为 protobuf 2.6.1，环境为 Centos 7.3 系统。
解压 protobuf-2.6.1.tar.gz 压缩包至 /usr/local 目录并进入该目录，执行命令如下：

```
[root@VM_0_8_centos]# tar -zxvf protobuf-2.6.1.tar.gz -C /usr/local/ && cd /usr/local/protobuf-2.6.1
```

开始编译和安装，配置环境变量，执行命令如下：

```
[root@VM_0_8_centos protobuf-2.6.1]# ./configure
[root@VM_0_8_centos protobuf-2.6.1]# make && make install
```

```
[root@VM_0_8_centos protobuf-2.6.1]# export PATH=$PATH:/usr/local/protobuf-2.6.1/bin
```

编译成功后，您可以使用以下命令查看版本：

```
[root@VM_0_8_centos protobuf-2.6.1]# protoc --version
liprotoc 2.6.1
```

2. 创建 PB 描述文件

PB 描述文件是通信双方约定的数据交换格式，上传日志时须将规定的协议格式编译成对应语言版本的调用接口，然后添加到工程代码里，详情请参见 protoc 。

以日志服务所规定的 PB 数据格式内容为准， 在本地创建 PB 消息描述文件 cls.proto。

> ⚠ 注意：
> PB 描述文件内容不可更改，且文件名须以 .proto 结尾。

cls.proto 内容（PB 描述文件）如下：

```
package cls;

message Log
{
message Content
{
required string key = 1; // 每组字段的 key
required string value = 2; // 每组字段的 value
}
required int64 time = 1; // 时间戳，UNIX时间格式
repeated Content contents = 2; // 一条日志里的多个kv组合
}

message LogTag
{
required string key = 1;
required string value = 2;
}

message LogGroup
{
```

```
repeated Log logs = 1; // 多条日志合成的日志数组
optional string contextFlow = 2; // 目前暂无效用
optional string filename = 3; // 日志文件名
optional string source = 4; // 日志来源，一般使用机器IP
repeated LogTag logTags = 5;
}

message LogGroupList
{
repeated LogGroup logGroupList = 1; // 日志组列表
}
```

3. 编译生成

此例中，使用 proto 编译器生成 C++ 语言的文件，在 cls.proto 文件的同一目录下，执行如下编译命令：

```
protoc --cpp_out=./ ./cls.proto
```

> 说明：
> `--cpp_out=./` 表示编译成 cpp 格式并输出当前目录下，`./cls.proto` 表示位于当前目录下的 cls.proto 描述文件。

编译成功后，会输出对应语言的代码文件。此例会生成 cls.pb.h 头文件和 cls.pb.cc 代码实现文件，如下所示：

```
[root@VM_0_8_centos protobuf-2.6.1]# protoc --cpp_out=./ ./cls.proto
[root@VM_0_8_centos protobuf-2.6.1]# ls
cls.pb.cc cls.pb.h cls.proto
```

4. 调用

将生成的 cls.pb.h 头文件引入代码中，调用接口进行数据格式封装。

获取下载日志游标

最近更新时间：2020-08-10 15:52:55

功能描述

本接口用于获取指定日志主题下的日志游标，并进行下载。

请求

请求示例

```
GET /cursor?topic_id=xxxxxxxx-xxxx-xxxx-xxxx&start=2017-12-28%2014%3A13%3A00 HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
```

请求行

```
GET /cursor
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	必须	含义
topic_id	string	query	是	日志主题的 ID
start	string	query	是	日志的开始时间，精确到分钟，格式 YYYY-mm-dd HH:MM:SS

响应

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 23

{
"cursor": "1212ssssxxxxxx"
}
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

字段名	类型	必有	含义
cursor	string	是	游标

错误码

参见 错误码。

搜索日志

最近更新时间：2020-12-03 16:53:02

功能描述

本接口用于根据指定的条件搜索日志内容。

请求

请求示例

```
GET /searchlog?logset_id=xxxx-xx-xx-xx-xxxxxxxx&topic_ids=xxxx&start_time=2017-08-22+10%3A10%3A10&end_time=2017-08-23+10%3A10%3A10&query_string=&limit=10&context= HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
```

请求行

```
GET /searchlog
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	必须	含义
logset_id	string	query	是	要查询的 logset ID
topic_ids	string	query	是	要查询的 topic ID
start_time	string	query	是	要查询的日志的起始时间，格式 YYYY-mm-dd HH:MM:SS
end_time	string	query	是	要查询的日志的结束时间，格式 YYYY-mm-dd HH:MM:SS
query_string	string	query	是	查询语句，详情参考 检索语法与规则
limit	int	query	是	单次要返回的日志条数，单次返回的最大条数为100
context	string	query	否	加载更多使用，透传上次返回的 context 值，获取后续的日志内容，通过游标最多可获取10000条，请尽可能缩小时间范围
sort	string	query	否	按时间排序 asc（升序）或者 desc（降序），默认为 desc

响应

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 53
{
"context": "abcdefg",
"listover": false,
"results": [
{
"timestamp": "2017-07-14 20:43:00",
"topic_id": "xxxx-xx-xx-xx-xxxxxxxx",
"topic_name": "xxxxxxx",
"content": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
},
{
"timestamp": "2017-07-14 20:42:00",
"topic_id": "xxxx-xx-xx-xx-xxxxxxxx",
"topic_name": "xxxxxxx",
"content": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
}
]
}
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

字段名	类型	必有	含义
context	string	是	获取更多检索结果的游标
listover	bool	是	搜索结果是否已经全部返回
results	JsonArray（LogObject）	是	日志内容信息

LogObject 格式如下：

字段名	类型	必有	含义

字段名	类型	必有	含义
topic_id	string	是	日志属于的 topic ID
topic_name	string	是	日志主题的名字
timestamp	string	是	日志时间
content	string	是	日志内容
filename	string	是	采集路径
source	string	是	日志来源设备

错误码

请参见 错误码。

下载日志

最近更新时间：2020-08-10 15:53:35

功能描述

本接口用于使用游标下载日志。

请求

请求示例

```
GET /log?topic_id=xxxxxxxx-xxxx-xxxx-xxxx&cursor=xxxxxx&count=10 HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
```

请求行

```
GET /log
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	必须	含义
topic_id	string	query	是	日志主题的 ID
cursor	string	query	是	通过 获取日志游标 接口获取的游标
count	string	query	是	需要下载的日志条数，最大1000

响应

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/x-protobuf
Content-Length: 23
x-cls-cursor: xxxxxx
x-cls-count:10
```

```
<LogGroupList 的pb格式打包内容>
```

响应头

Header 名	含义
x-cls-cursor	当前的日志游标，供下次继续下载使用
x-cls-count	当前请求下载到的日志条数

响应参数

返回 LogGroupList 对象的打包内容，pb 文件描述请参见 结构化日志上传 接口。

错误码

请参见 错误码 文档。

日志集管理
创建日志集

最近更新时间：2020-08-10 15:51:14

功能描述

本接口用于创建日志集，返回新创建的日志集的 ID。

请求

请求示例

```
POST /logset HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
Content-Type: application/json

{"logset_name": "testname","period": 15}
```

请求行

```
POST /logset
```

请求头

除公共响应头部外，无特殊响应头部。

请求参数

字段名	类型	位置	必须	含义
logset_name	string	body	是	日志集的名字，不能重名
period	int	body	是	日志集的保存周期，单位天，最大 90

响应

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
Content-Length: 123

{"logset_id": "xxxx-xx-xx-xx-xxxxxxxx"}
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

字段名	类型	必有	含义
logset_id	string	是	日志集的 ID

错误码

参见 错误码。

获取日志集信息

最近更新时间：2020-08-10 15:49:10

功能描述

本接口用于获取日志集信息。

请求

请求示例

```
GET /logset?logset_id=xxxx-xx-xx-xx-xxxxxxxx HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
```

请求行

```
GET /logset
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	必须	含义
logset_id	string	query	是	查询的 logset ID

响应

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 123

{
"logset_id": "xxxx-xx-xx-xx-xxxxxxxx",
"logset_name": "testname",
"period": 15,
```

```
"create_time": "2017-08-08 12:12:12",
"assumer_uin": 1000088888,
"assumer_name": "xxxxxx",
"logset_modify_acl": 31
}
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

字段名	类型	必有	含义
logset_id	string	是	日志集的 ID
logset_name	string	是	日志集的名字
period	int	是	保存周期（天）
create_time	string	否	创建时间
assumer_uin	uint64	否	创建 logset 的服务的 uin（仅普通账号查看服务账号创建的 logset，才有此字段）
assumer_name	string	否	创建 logset 的服务的名称（仅普通账号查看服务账号创建的 logset，才有此字段）
logset_modify_acl	int	否	普通用户对 logset 的修改权限 modify_acl（0B00000=禁止修改，0B00001=允许修改基本信息）（仅普通账号查看服务账号创建的 logset，才有此字段）

错误码

参见 错误码。

获取日志集列表

最近更新时间：2020-08-10 15:47:30

功能描述

本接口用于获取日志集信息列表。

请求

请求示例

```
GET /logsets HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
```

请求行

```
GET /logsets
```

请求头

除公共头部外，无特殊请求头部。

请求参数

无。

响应

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 123

{
"logsets": [
{
"logset_id": "xxxx-xx-xx-xx-xxxxxxxx",
"logset_name": "testname",
"period": 15,
```

```
"create_time": "2017-08-08 12:12:12",
"assumer_uin": 1000088888,
"assumer_name": "xxxxxx",
"logset_modify_acl": 31
}
]
}
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

字段名	类型	必有	含义
logsets	JsonArray	是	日志集信息数组

LogsetInfo 格式如下：

字段名	类型	必有	含义
logset_id	string	是	日志集的 ID
logset_name	string	是	日志集的名字
period	int	是	保存周期（天）
create_time	string	否	创建时间
assumer_uin	uint64	否	创建 logset 的服务的 uin（仅普通账号查看服务账号创建的 logset，才有此字段）
assumer_name	string	否	创建 logset 的服务的名称（仅普通账号查看服务账号创建的 logset，才有此字段）
logset_modify_acl	int	否	普通用户对 logset 的修改权限 modify_acl（0B00000=禁止修改，0B00001=允许修改基本信息）（仅普通账号查看服务账号创建的 logset，才有此字段）

错误码

参见 错误码。

修改日志集

最近更新时间：2020-08-10 15:52:12

功能描述

本接口用于修改日志集。

请求

请求示例

```
PUT /logset HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
Content-Type: application/json

{"logset_id": "xxxx-xx-xx-xx-xxxxxxx","logset_name": "testname","period": 15}
```

请求行

```
PUT /logset
```

请求头

除公共响应头部外，无特殊响应头部。

请求参数

字段名	类型	位置	必须	含义
logset_id	string	body	是	要修改的日志集的 ID
logset_name	string	body	否	日志集的名字，不能重名
period	int	body	否	日志集的保存周期，单位天，最大90

> ⚠ 注意：
> logset_name 和 period 至少要提供一个。

响应

响应示例

```
HTTP/1.1 200 OK
Content-Length: 0
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

如果更新成功，无响应参数。

错误码

参见 错误码。

删除日志集

最近更新时间：2020-08-10 15:50:07

功能描述

本接口用于删除日志集。

请求

请求示例

```
DELETE /logset?logset_id=xxxx-xx-xx-xx-xxxxxxxx HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
```

请求行

```
DELETE /logset
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	必须	含义
logset_id	string	query	是	要删除的日志集的 ID

响应

响应示例

```
HTTP/1.1 200 OK
Content-Length: 0
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

如果删除成功，无响应参数。

错误码

参见 错误码。

日志主题管理
创建日志主题

最近更新时间：2020-08-10 15:46:17

功能描述

本接口用于创建日志主题，返回新创建的日志主题的 ID。

请求

请求示例

```
POST /topic HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
Content-Type: application/json

{
"logset_id": "xxxxxx-xx-xx-xx-xxxxxxxx",
"topic_name": "testname",
"partition_count": "1",
"path": "/data/nginx/log/access.log",
"wild_path":"/data/nginx/log/**/access.log",
"log_type": "delimiter_log",
"extract_rule": {
"time_key": "date",
"time_format": "%Y-%m-%d %H:%M:%S",
"delimiter": "|",
"log_regex": ".*",
"beginning_regex": "^",
"keys": ["date","","content"],
"filter_keys": [],
"filter_regex": []
}
}
```

请求行

```
POST /topic
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	是否必须	含义
logset_id	string	body	是	日志主题归属的日志集的 ID
topic_name	string	body	是	日志主题的名字
partition_count	int	body	否	主题分区 partition个数，不传参默认创建1个，最大创建允许10个，分裂/合并操作会改变分区数量，整体上限50个
path	string	body	否	旧版日志主题需要采集的日志路径，不采集无需设置
wild_path	string	body	否	新版通配符日志采集路径，以/**/分隔文件目录和文件名，和旧版path只会存在一个
log_type	string	body	否	采集的日志类型，json_log代表 json 格式日志，delimiter_log代表分隔符格式日志，minimalist_log代表单行全文格式，multiline_log代表多行日志，fullregex_log代表完整正则，默认为minimalist_log
extract_rule	JsonObject	body	否	提取规则，如果设置了 extract_rule，则必须设置 log_type

extract_rule 格式如下：

字段名	类型	是否必须	含义
time_key	string	否	时间字段的 key 名字，time_key 和 time_format 必须成对出现
time_format	string	否	时间字段的格式，参考 C 语言的strftime函数对于时间的格式说明
delimiter	string	否	分隔符类型日志的分隔符，只有log_type为delimiter_log时有效
log_regex	string	否	整条日志匹配规则，只有log_type为fullregex_log时有效

字段名	类型	是否必须	含义
beginning_regex	string	否	行首匹配规则，只有log_type为multiline_log时有效
keys	JsonArray（string）	否	提取的每个字段的 key 名字，为空的 key 代表丢弃这个字段，只有log_type为delimiter_log时有效，json_log的日志使用 json 本身的 key
filter_keys	JsonArray（string）	否	需要过滤日志的 key，最多5个
filter_regex	JsonArray（string）	否	上述字段 filter_keys 对应的值，个数与 filter_keys 相同，一一对应，采集匹配的日志

响应

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 123

{"topic_id": "xxxx-xx-xx-xx-xxxxxxxx"}
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

字段名	类型	是否必须	含义
topic_id	string	是	日志主题的 ID

错误码

参见 错误码。

获取日志主题信息

最近更新时间：2020-08-10 15:45:01

功能描述

本接口用于获取日志主题信息。

请求

请求示例

```
GET /topic?topic_id=xxxx-xx-xx-xx-yyyyyyyy HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
```

请求行

```
GET /topic
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	是否必须	含义
topic_id	string	query	是	查询的 topic ID

响应

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 123

{
"logset_id": "xxxx-xx-xx-xx-xxxxxxxx",
"topic_id": "xxxx-xx-xx-xx-yyyyyyyy",
"topic_name": "testname",
```

```
"partition_count": "1",
"path": "/abc/log/test.log",
"wild_path": "/data/nginx/log/**/access.log",
"collection": true,
"index": true,
"log_type": "delimiter_log",
"extract_rule": {
"time_key": "date",
"time_format": "%Y-%m-%d %H:%M:%S",
"delimiter": "|",
"log_regex": ".*",
"beginning_regex": "^",
"keys": ["date", "", "content"],
"filter_keys": [],
"filter_regex": []
},
"assumer_uin": 1000088888,
"assumer_name": "xxxxxx",
"topic_modify_acl": 31,
"topic_show_acl": 31,
"create_time": "2017-08-08 12:12:12"
}
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

字段名	类型	是否必须	含义
logset_id	string	是	日志集的 ID
topic_id	string	是	日志主题的 ID
topic_name	string	是	日志主题的名字
partition_count	int	是	主题分区 partition 的数量
path	string	是	旧版日志文件路径
wild_path	string	是	新版通配符日志文件路径，以/**/分隔文件目录和文件名，和旧版 path 只会存在一个
collection	bool	是	是否开启采集

字段名	类型	是否必须	含义
index	bool	是	是否开启索引
log_type	string	是	采集的日志类型，json_log代表 json 格式日志，delimiter_log代表分隔符格式日志，minimalist_log代表单行全文格式，multiline_log代表多行日志，fullregex_log代表完整正则
extract_rule	JsonObject	是	提取规则
machine_group	JsonObject	否	采集机器组信息
assumer_uin	uint64	否	创建主题的服务的 uin（仅普通账号查看服务账号创建的主题，才有此字段）
assumer_name	string	否	创建主题的服务的名称（仅普通账号查看服务账号创建的主题，才有此字段）
topic_modify_acl	int	否	普通用户对主题的修改权限 modify_acl（0B00000=禁止修改，0B00001=允许修改基本信息，0B00010=允许修改采集信息，0B00100=允许修改索引信息，0B01000=允许修改投递信息，0B10000=允许修改消费信息）（仅普通账号查看服务账号创建的主题，才有此字段）
topic_show_acl	int	否	主题展示给普通用户的权限 show_acl（0B00000=全部不展示，0B00001=展示基本信息，0B00010=展示采集信息，0B00100=展示索引信息，0B01000=展示投递信息，0B10000=展示消费信息）
create_time	string	否	创建时间

extract_rule 格式如下：

字段名	类型	是否必须	含义
time_key	string	否	时间字段的 key 名字
time_format	string	否	时间字段的格式，参考 C 语言的strftime函数对于时间的格式说明
delimiter	string	否	分隔符类型日志的分隔符
log_regex	string	否	完全正则类型的日志匹配规则
beginning_regex	string	否	多行日志类型的 行首匹配规则
keys	JsonArray(string)	否	提取的每个字段的 key 名字

字段名	类型	是否必须	含义
filter_keys	JsonArray(string)	否	需要过滤日志的 key
filter_regex	JsonArray(string)	否	上述字段 filter_keys 对应的值，个数与 filter_keys 相同，一一对应

错误码

参见 错误码。

获取日志主题绑定的机器组

最近更新时间：2020-07-31 11:23:58

功能描述

本接口用于获取日志主题绑定的机器组信息。

请求

请求示例

```
GET /topic/machinegroup?topic_id=xxxx-xx-xx-xx-xxxxxxxx HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <Authorization String>
```

请求行

```
GET /topic/machinegroup
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	是否必须	含义
topic_id	string	query	是	查询的日志主题 ID

响应

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 123
{
"machine_groups": [
{
"group_id": "xxxx-xx-xx-xx-yyyyyyyy",
"group_name": "testname"},
```

```
{"group_id": "xxxx-xx-xx-xx-zzzzzzzz",
"group_name": "testname1"}
]
}
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

字段名	类型	是否必须	含义
machine_groups	JsonArray	是	日志主题绑定的机器组数组

machine_groups 格式如下：

字段名	类型	是否必须	含义
group_id	string	是	机器组的 ID
group_name	string	是	机器组的名字

错误码

请查看 错误码 文档。

获取日志主题列表

最近更新时间：2020-08-10 15:44:09

功能描述

本接口用于获取日志主题信息列表。

请求

请求示例

```
GET /topics?logset_id=xxxx-xx-xx-xx-xxxxxxxx HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
```

请求行

```
GET /topics
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	是否必须	含义
logset_id	string	query	是	查询的 logset ID

响应

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 123

{
"topics": [{
"logset_id": "xxxx-xx-xx-xx-xxxxxxxx",
"topic_id": "xxxx-xx-xx-xx-yyyyyyyy",
```

```
"topic_name": "testname",
"partition_count": "1",
"path": "/abc/log/test.log",
"wild_path": "/data/nginx/log/**/access.log",
"collection": true,
"index": true,
"log_type": "delimiter_log",
"extract_rule": {
"time_key": "date",
"time_format": "%Y-%m-%d %H:%M:%S",
"delimiter": "|",
"log_regex": ".*",
"beginning_regex": "^",
"keys": ["date", "", "content"],
"filter_keys": [],
"filter_regex": []
},
"assumer_uin": 1000088888,
"assumer_name": "xxxxxx",
"topic_modify_acl": 31,
"topic_show_acl": 31,
"create_time": "2017-08-08 12:12:12"
}]
}
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

字段名	类型	是否必须	含义
topics	JsonArray	是	日志主题信息数组

TopicInfo 格式如下：

字段名	类型	是否必须	含义
logset_id	string	是	日志集的 ID
topic_id	string	是	日志主题的 ID
topic_name	string	是	日志主题的名字

字段名	类型	是否必须	含义
partition_count	int	是	主题分区 partition 的数量
path	string	是	旧版日志文件路径
wild_path	string	否	新版通配符日志采集路径，以/**/分隔文件目录和文件名，和旧版 path 只会存在一个
collection	bool	是	是否开启采集
index	bool	是	是否开启索引
log_type	string	是	采集的日志类型，json_log代表json格式日志，delimiter_log代表分隔符格式日志，minimalist_log代表极简日志，egex_log、multiline_log代表多行日志，fullregex_log代表完整正则
extract_rule	JsonObject	是	提取规则
machine_group	JsonObject	否	采集机器组信息
assumer_uin	uint64	否	创建主题的服务的 uin（仅普通账号查看服务账号创建的主题，才有此字段）
assumer_name	string	否	创建主题的服务的名称（仅普通账号查看服务账号创建的主题，才有此字段）
topic_modify_acl	int	否	普通用户对主题的修改权限 modify_acl（0B00000=禁止修改，0B00001=允许修改基本信息，0B00010=允许修改采集信息，0B00100=允许修改索引信息，0B01000=允许修改投递信息，0B10000=允许修改消费信息）（仅普通账号查看服务账号创建的主题，才有此字段）
topic_show_acl	int	否	主题展示给普通用户的权限 show_acl（0B00000=全部不展示，0B00001=展示基本信息，0B00010=展示采集信息，0B00100=展示索引信息，0B01000=展示投递信息，0B10000=展示消费信息）
create_time	string	否	创建时间

extract_rule 格式如下：

字段名	类型	是否必须	含义
time_key	string	否	时间字段的 key 名字

字段名	类型	是否必须	含义
time_format	string	否	时间字段的格式，参考 C 语言的strftime函数对于时间的格式说明
delimiter	string	否	分隔符类型日志的分隔符
log_regex	string	否	多行日志类型的 整条日志匹配规则
beginning_regex	string	否	多行日志类型的行首匹配规则
keys	JsonArray(string)	否	提取的每个字段的 key 名字
filter_keys	JsonArray(string)	否	需要过滤日志的 key
filter_regex	JsonArray(string)	否	上述字段 filter_keys 对应的值，个数与 filter_keys 相同，一一对应

错误码

参见 错误码。

修改日志主题

最近更新时间：2020-09-02 16:23:00

功能描述

本接口用于修改日志主题。

请求

请求示例

```
PUT /topic HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
Content-Type: application/json

{
"topic_id": "xxxxxx-xx-xx-xx-xxxxxxxx",
"topic_name": "testname",
"path": "/data/nginx/log/access.log",
"wild_path":"/data/nginx/log/**/access.log",
"collection": false,
"log_type": "delimiter_log",
"extract_rule": {
"time_key": "date",
"time_format": "%Y-%m-%d %H:%M:%S",
"delimiter": "|",
"log_regex": ".*",
"beginning_regex": "^",
"keys": ["date","","content"],
"filter_keys": [],
"filter_regex": []
}
}
```

请求行

```
PUT /topic
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	是否必须	含义
topic_id	string	body	是	日志主题的 ID
topic_name	string	body	否	日志主题的名字
path	string	body	否	旧版日志主题需要采集的日志路径
wild_path	string	body	否	新版通配符日志采集路径，以/**/分隔文件目录和文件名，和旧版 path 只会存在一个
collection	bool	body	否	是否开启采集
log_type	string	body	否	采集的日志类型： • json_log代表 json 格式日志 • delimiter_log代表分隔符格式日志 • minimalist_log代表单行全文格式 • multiline_log代表多行日志 • fullregex_log代表完整正则
extract_rule	JsonObject	body	否	提取规则

> ⚠ 注意：
> topic_name、path、group_id、collection、（log_type+extract_rule）至少要提供一个。

extract_rule 格式如下：

字段名	类型	是否必须	含义
time_key	string	否	时间字段的 key 名字，time_key 和 time_format 必须成对出现
time_format	string	否	时间字段的格式，参考 C 语言的strftime函数对于时间的格式说明
delimiter	string	否	分隔符类型日志的分隔符，只有log_type为delimiter_log时有效
log_regex	string	否	整条日志匹配规则，只有log_type为fullregex_log时有效

字段名	类型	是否必须	含义
beginning_regex	string	否	行首匹配规则，在log_type为fullregex_log或multiline_log的时候有效
keys	JsonArray(string)	否	提取的每个字段的 key 名字，为空的 key 代表丢弃这个字段，只有log_type为delimiter_log时有效，json_log的日志使用 json 本身的 key
filter_keys	JsonArray(string)	否	需要过滤日志的 key，最多5个
filter_regex	JsonArray(string)	否	上述字段 filter_keys 对应的值，个数与 filter_keys 相同，一一对应，采集匹配的日志

响应

响应示例

```
HTTP/1.1 200 OK
Content-Length: 0
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

无。

错误码

参见 错误码。

设置日志主题绑定的机器组

最近更新时间：2020-07-31 11:23:15

功能描述

本接口用于设置日志主题绑定的机器组信息。

请求

请求示例

```
PUT /topic/machinegroup?topic_id=xxxx-xx-xx-xx-xxxxxxxx HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <Authorization String>
Content-Type: application/json
{
"machine_groups": ["xxxxxx-xx-xx-xx-yyyyyyyy"]
}
```

请求行

```
PUT /topic/machinegroup
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	是否必须	含义
topic_id	string	query	是	设置的日志主题 ID
machine_groups	JsonArray（string）	body	是	日志主题绑定的机器组 ID 数组

响应

响应示例

```
HTTP/1.1 200 OK
Content-Length: 0
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

无。

错误码

请查看 错误码 文档。

删除日志主题

最近更新时间：2020-08-10 15:45:38

功能描述

本接口用于删除日志主题。

请求

请求示例

```
DELETE /topic?topic_id=xxxx-xx-xx-xx-xxxxxxxx HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
```

请求行

```
DELETE /topic
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	必须	含义
topic_id	string	query	是	要删除的日志主题的 ID

响应

响应示例

```
HTTP/1.1 200 OK
Content-Length: 0
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

无。

错误码

参见 错误码。

分区管理
获取主题分区列表

最近更新时间：2020-07-31 11:01:22

功能描述

本接口用于获取主题分区信息列表。

请求

请求示例

```
GET /partitions?topic_id=xxxx-xx-xx-xx-xxxx HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
```

请求头

除公共头部外，无特殊请求头部。

请求参数

参数名	类型	位置	是否必须	描述
topic_id	string	query	是	日志主题 ID

响应

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 21

{
"partitions":[
{
"partition_id": 1,
"status": "readwrite",
"inclusive_begin_key": "00000000000000000000000000000000",
"exclusive_end_key": "a0000000000000000000000000000000",
```

```
"create_time": "2019-01-14 19:19:41"
},
{
"partition_id": 2,
"status": "readwrite",
"inclusive_begin_key": "a00000000000000000000000000000000000",
"exclusive_end_key": "ffffffffffffffffffffffffffffffffffff",
"create_time": "2019-01-14 19:19:41"
}
]
}
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

字段名	类型	是否必须	说明
partition_id	int	是	主题分区编号
status	string	是	主题分区状态： • readwrite：读写态 • readonly：只读态
inclusive_begin_key	string	是	主题分区范围的起始位置
exclusive_end_key	string	是	主题分区范围的结束位置
create_time	string	是	主题分区的创建时间

错误码

参见 错误码。

合并主题分区

最近更新时间：2020-07-31 11:00:43

功能描述

本接口用于合并一个读写态的主题分区，合并时指定一个主题分区 ID，日志服务会自动合并范围右相邻的分区。

请求

请求示例

```
POST /partitions?topic_id=xxxx-xx-xx-xx-xxxx&partition_id=2&action=merge HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	是否必须	说明
topic_id	string	query	是	分区所属的日志主题 ID
partition_id	int	query	是	需合并的主题分区编号，日志服务会自动合并范围右相邻的分区。 例如，若2、3是两个相邻的 readwrite 分区，paritition_id=2时，会将2、3分区合并，同时2、3分区类型变成 readonly，合并的后的分区类型为 readwrite，partition_id=4
action	string	query	是	操作类型，action 需要设置为：merge

响应

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 21

{
```

```
"partitions":[
{
"partition_id": 2,
"status": "readonly",
"inclusive_begin_key": "00000000000000000000000000000000",
"exclusive_end_key": "7fffffffffffffffffffffffffffffff",
"create_time": "2019-01-14 19:25:41"
},
{
"partition_id": 3,
"status": "readonly",
"inclusive_begin_key": "7fffffffffffffffffffffffffffffff",
"exclusive_end_key": "ffffffffffffffffffffffffffffffff",
"create_time": "2019-01-14 19:25:41"
},
{
"partition_id": 4,
"status": "readwrite",
"inclusive_begin_key": "00000000000000000000000000000000",
"exclusive_end_key": "ffffffffffffffffffffffffffffffff",
"create_time": "2019-01-14 19:33:41"
}
]
}
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

字段名	类型	说明
partition_id	int	主题分区编号
status	string	主题分区状态： • readwrite：读写态 • readonly：只读态
inclusive_begin_key	string	主题分区范围的起始位置
exclusive_end_key	string	主题分区范围的结束位置
create_time	string	主题分区的创建时间

错误码

参见 错误码。

分裂主题分区

最近更新时间：2020-10-15 15:26:50

功能描述

本接口用于分裂一个读写态的主题分区。

请求

请求示例

```
POST /partitions?topic_id=xxxx-xx-xx-xx-xxxx&partition_id=1&split_key=7fffffffffffffffffffffffffffffff&action=split HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	是否必须	说明
topic_id	string	query	是	分区所属的日志主题 ID
partition_id	int	query	是	需分裂的主题分区编号
action	string	query	是	操作类型，action 需要设置为 split
split_key	string	query	否	分裂成两个时，可以指定主题分区的分裂位置，32位16进制字符串（不含0x部分）;当分裂成3个及以上时，按平均方式分裂，此参数不生效
number	int	query	否	分裂的个数，默认值为2

响应

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 21
```

```
{
"partitions":[
{
"partition_id": 1,
"status": "readonly",
"inclusive_begin_key": "00000000000000000000000000000000",
"exclusive_end_key": "ffffffffffffffffffffffffffffffff",
"create_time": "2019-01-14 19:19:41"
},
{
"partition_id": 2,
"status": "readwrite",
"inclusive_begin_key": "00000000000000000000000000000000",
"exclusive_end_key": "7fffffffffffffffffffffffffffffff",
"create_time": "2019-01-14 19:25:41"
},
{
"partition_id": 3,
"status": "readwrite",
"inclusive_begin_key": "7fffffffffffffffffffffffffffffff",
"exclusive_end_key": "ffffffffffffffffffffffffffffffff",
"create_time": "2019-01-14 19:25:41"
}
]
}
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

字段名	类型	说明
partition_id	int	主题分区编号
status	string	主题分区状态： • readwrite：读写态 • readonly：只读态
inclusive_begin_key	string	主题分区范围的起始位置
exclusive_end_key	string	主题分区范围的结束位置

字段名	类型	说明
create_time	string	主题分区的创建时间

错误码

参见 错误码。

投递任务管理
创建投递任务

最近更新时间：2020-05-14 16:49:50

功能描述

本接口用于创建新的投递任务，如果使用此接口，需要自行处理 CLS 对指定 Bucket 的写权限。

请求

请求示例

```
POST /shipper HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
Content-Type: application/json

{
"topic_id": "xxxx-xx-xx-xx-xxxxxxxx",
"bucket": "test-1250000001",
"prefix": "test",
"shipper_name": "myname",
"interval": 300,
"max_size": 100,
"partition": "%Y%m%d",
"compress": {
"format": "none"
},
"content": {
"format": "csv",
"csv_info": {
"print_key": true,
"keys": ["key1", "key2"],
"delimiter": "|",
"escape_char": "'",
"non_existing_field": "null"
}
}
}
```

请求行

```
POST /shipper
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	是否必须	含义
topic_id	string	body	是	创建的 Shipper 属于的 topic ID
bucket	string	body	是	创建的 Shipper 投递的 bucket，格式：{bucketName}-{appid}
prefix	string	body	是	创建的 Shipper 投递目录的前缀
shipper_name	string	body	是	投递规则的名字
interval	int	body	否	投递的时间间隔，单位：秒，默认300，取值范围为：[300，360，420，480，540，600，660，720，780，840，900]（即整数分钟级别数值）
max_size	int	body	否	投递的文件的最大值，单位：MB，默认100，范围100 - 256
partition	string	body	否	投递日志的分区规则，支持strftime的时间格式表示
compress	object	body	否	投递日志的压缩配置
content	object	body	否	投递日志的内容格式配置

compress 格式如下：

字段名	类型	是否必须	含义
format	string	是	压缩格式，支持gzip、lzop和none不压缩

content 格式如下：

字段名	类型	是否必须	含义
format	string	是	内容格式，支持json、csv
csv_info	object	否	内容格式为csv时设置

csv_info 格式如下：

字段名	类型	是否必须	含义
print_key	bool	是	csv 首行是否打印 key
keys	array(string)	是	每列 key 的名字
delimiter	string	是	各字段间的分隔符
escape_char	string	是	若字段内容中包含分隔符，则使用该转义符包裹改字段
non_existing_field	string	是	对于上面指定的不存在字段使用该内容填充

响应

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 0
{
"shipper_id": "xxxx-xx-xx-xx-xxxxxxxx",
}
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

字段名	类型	是否必须	含义
shipper_id	string	是	新创建的投递的 ID

错误码

参见 错误码。

获取投递配置

最近更新时间：2020-11-03 09:51:56

功能描述

本接口用于获取指定投递策略的详细信息。

请求

请求示例

```
GET /shipper?shipper_id=xxxx-xx-xx-xx-xxxxxxxx HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
```

请求行

```
GET /shipper
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	是否必须	含义
shipper_id	string	query	是	查询的 shipper ID

响应

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 123

{
"shipper_id": "xxxx-xx-xx-xx-xxxxxxxx",
"topic_id": "yyyy-yy-yy-yy-yyyyyyyy",
"bucket": "test-1250000001",
```

```
"prefix": "test",
"shipper_name": "myname",
"interval": 300,
"max_size": 100,
"effective": true,
"partition": "%Y%m%d",
"compress": {
"format": "none"
},
"content": {
"format": "json"
},
"create_time": "2017-12-12 12:12:12"
}
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

字段名	类型	是否必须	含义
shipper_id	string	是	投递的 ID
topic_id	string	是	投递规则属于的 topic ID
bucket	string	是	投递的 bucket 地址
prefix	string	是	投递的前缀目录
shipper_name	string	是	投递规则的名字
interval	int	是	投递的时间间隔，单位秒
max_size	int	是	投递的文件的最大值，单位 MB
effective	bool	是	是否生效
create_time	string	是	投递日志的创建时间
partition	string	是	投递日志的分区规则，支持strftime的时间格式表示
compress	object	是	投递日志的压缩配置
content	object	是	投递日志的内容格式配置

compress 格式如下：

字段名	类型	是否必须	含义
format	string	是	压缩格式，支持gzip、lzop和none不压缩

content 格式如下：

字段名	类型	是否必须	含义
format	string	是	内容格式，支持json、csv
csv_info	object	否	内容格式为csv时返回

csv_info 格式如下：

字段名	类型	是否必须	含义
print_key	bool	是	csv 首行是否打印 key
keys	array（string）	是	每列 key 的名字
delimiter	string	是	各字段间的分隔符
escape_char	string	是	若字段内容中包含分隔符，则使用该转义符包裹改字段
non_existing_field	string	是	对于上面指定的不存在字段使用该内容填充

错误码

参见 错误码。

获取日志主题投递列表

最近更新时间：2020-11-03 09:51:17

功能描述

本接口用于获取指定日志主题的投递策略详细列表。

请求

请求示例

```
GET /shippers?topic_id=xxxx-xx-xx-xx-xxxxxxxx HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <Authorization String>
```

请求行

```
GET /shippers
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	是否必须	含义
topic_id	string	query	是	查询的 Shipper 属于的 topic ID
offset	int	query	否	查询的起始位置，默认0
count	int	query	否	查询的个数，默认50，最大1000

响应

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 123

{
```

```
"shippers": [
{
"shipper_id": "xxxx-xx-xx-xx-xxxxxxxx",
"topic_id": "yyyy-yy-yy-yy-yyyyyyyy",
"bucket": "test-1250000001",
"prefix": "test",
"shipper_name": "myname",
"interval": 300,
"max_size": 100,
"effective": true,
"partition": "%Y%m%d",
"compress": {
"format": "none"
},
"content": {
"format": "json"
},
"create_time": "2017-12-12 12:12:12"
}
]
}
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

字段名	类型	是否必须	含义
shippers	JsonArray	是	投递信息数组

ShipperInfo 格式如下：

字段名	类型	是否必须	含义
shipper_id	string	是	投递的 ID
topic_id	string	是	投递规则属于的 topic ID
bucket	string	是	投递的 bucket 地址
prefix	string	是	投递的前缀目录

字段名	类型	是否必须	含义
shipper_name	string	是	投递规则的名字
interval	int	是	投递的时间间隔，单位秒
max_size	int	是	投递的文件的最大值，单位 MB
effective	bool	是	是否生效
create_time	string	是	投递日志的创建时间
partition	string	是	投递日志的分区规则，支持 strftime 的时间格式表示
compress	object	是	投递日志的压缩配置
content	object	是	投递日志的内容格式配置

compress格式如下：

字段名	类型	是否必须	含义
format	string	是	压缩格式，支持 gzip、lzop 和 none 不压缩

content 格式如下：

字段名	类型	是否必须	含义
format	string	是	内容格式，支持 json、csv
csv_info	object	否	内容格式为 csv 时返回

csv_info 格式如下：

字段名	类型	是否必须	含义
print_key	bool	是	csv 首行是否打印 key
keys	array（string）	是	每列 key 的名字
delimiter	string	是	各字段间的分隔符
escape_char	string	是	若字段内容中包含分隔符，则使用该转义符包裹改字段
non_existing_field	string	是	对于上面指定的不存在字段使用该内容填充

错误码

请参见 错误码 文档。

获取投递任务列表

最近更新时间：2020-08-10 15:43:09

功能描述

本接口可用于获取投递任务信息列表。

请求

请求示例

```
GET /tasks?shipper_id=xx-xx-xx-xxxx&start_time=2017-10-10+00%3A00%3A00&end_time=2017-10-10
+23%3A59%3A59 HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
```

请求行

```
GET /tasks
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	必须	含义
shipper_id	string	query	是	查询的投递规则 ID
start_time	string	query	是	查询的开始时间，支持最近3天的查询
end_time	string	query	是	查询的结束时间

响应

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 123
```

```
{
"tasks": [
{
"task_id": "xxxxx-xx-xx-xx",
"shipper_id": "xxxxx-xx-xx-xx",
"topic_id": "xxxxx-xx-xx-xx",
"range_start": "2017-10-17 10:10:10",
"range_end": "2017-10-17 10:10:10",
"start_time": "2017-10-17 10:10:10",
"end_time": "2017-10-17 10:10:10",
"status": "success",
"message": "success",
}
]
}
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

字段名	类型	必有	含义
tasks	JsonArray	是	投递任务信息数组

TaskInfo 格式如下：

字段名	类型	必有	含义
task_id	string	是	投递任务的 ID
shipper_id	string	是	投递规则的 ID
topic_id	string	是	日志主题的 ID
range_start	string	是	本批投递的日志的开始时间
range_end	string	是	本批投递的日志的结束时间
start_time	string	是	本次投递任务的开始时间
end_time	string	是	本次投递任务的结束时间
status	string	是	本次投递的结果，"success"，"running"，"failed"，"wait"

字段名	类型	必有	含义
message	string	是	结果的详细信息

错误码

参见 错误码。

修改投递任务

最近更新时间：2020-08-18 15:30:36

功能描述

本接口可用于修改现有的投递任务，客户如果使用此接口，需要自行处理 CLS 对指定 Bucket 的写权限。

请求示例

```
PUT /shipper HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
Content-Type: application/json

{
"shipper_id": "xxxx-xx-xx-xx-xxxxxxxx",
"bucket": "test-1250000001",
"prefix": "test",
"shipper_name": "myname",
"interval": 300,
"max_size": 100,
"effective": true,
"partition": "%Y%m%d",
"compress": {
"format": "none"
},
"content": {
"format": "json"
}
}
```

请求行

```
PUT /shipper
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	是否必须	含义
shipper_id	string	body	是	修改的 Shipper 的 ID
bucket	string	body	否	Shipper 投递的新的 bucket，格式：{bucketName}-{appid}
prefix	string	body	否	Shipper 投递的新的目录前缀
shipper_name	string	body	否	投递规则的名字
interval	int	body	否	投递的时间间隔，单位：秒，默认300，取值范围为：[300，360，420，480，540，600，660，720，780，840，900]（即整数分钟级别数值）
max_size	int	body	否	投递的文件的最大值，单位：MB，默认100，范围100 - 256
effective	bool	body	否	Shipper 的开关状态
partition	string	body	否	投递日志的分区规则，支持strftime的时间格式表示
compress	object	body	否	投递日志的压缩配置
content	object	body	否	投递日志的内容格式配置

compress 格式如下：

字段名	类型	是否必须	含义
format	string	是	压缩格式，支持gzip、lzop和none不压缩

content 格式如下：

字段名	类型	是否必须	含义
format	string	是	内容格式，支持json、csv
csv_info	object	否	内容格式为csv时设置

csv_info 格式如下：

字段名	类型	是否必须	含义
print_key	bool	是	csv 首行是否打印 key
keys	array(string)	是	每列 key 的名字

字段名	类型	是否必须	含义
delimiter	string	是	各字段间的分隔符
escape_char	string	是	若字段内容中包含分隔符，则使用该转义符包裹改字段
non_existing_field	string	是	对于上面指定的不存在字段使用该内容填充

> **注意：**
> 其中 bucket、prefix、shipper_name、interval、max_size、effective、compress 字段至少要有一个。

响应

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 0
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

无。

错误码

参见 错误码。

重试失败的任务

最近更新时间：2020-08-10 15:41:23

功能描述

本接口可用于重试失败的投递任务。

请求

请求示例

```
PUT /task HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
Content-Type: application/json

{
"shipper_id": "xxxxxx-xx-xx-xx-xxxxxxxx",
"task_id": "xxxxxx-xx-xx-xx-xyyyyyyy"
}
```

请求行

```
PUT /task
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	必须	含义
shipper_id	string	body	是	投递规则的 ID
task_id	string	body	是	投递任务的 ID

响应

响应示例

```
HTTP/1.1 200 OK
Content-Length: 0
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

无。

错误码

参见 错误码。

删除投递配置

最近更新时间：2020-08-10 15:42:27

功能描述

本接口用于删除投递配置。

请求

请求示例

```
DELETE /shipper?shipper_id=xxxx-xx-xx-xx-xxxxxxxx HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
```

请求行

```
DELETE /shipper
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	必须	含义
shipper_id	string	query	是	要删除的投递配置的 ID

响应

响应示例

```
HTTP/1.1 200 OK
Content-Length: 0
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

无

错误码

参见 错误码。

机器组管理
创建机器组

最近更新时间：2020-10-14 09:49:25

功能描述

本接口用于创建 机器组，返回新创建的机器组的 ID。

请求

请求示例

```
POST /machinegroup HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
Content-Type: application/json

{
"group_name": "testname",
"type" : "label",
"labels": ["defined_label_1", "defined_label_2"]
}
```

请求行

```
POST /machinegroup
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	必须	含义
group_name	string	body	是	机器组的名字，不能重名
type	string	body	否	机器组类型，只支持 ip 和 label 两种类型，默认是 ip
ips	JsonArray	body	否	ip 机器组下的 IP 列表
labels	JsonArray	body	否	label 机器组的标签表

响应

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 123

{"group_id": "xxxx-xx-xx-xx-xxxxxxxx"}
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

字段名	类型	必有	含义
group_id	string	是	机器组的 ID

错误码

参见 错误码。

获取机器组信息

最近更新时间：2020-10-14 09:48:13

功能描述

本接口用于获取机器组信息。

请求

请求示例

```
GET /machinegroup?group_id=xxxx-xx-xx-xx-xxxxxxxx HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
```

请求行

```
GET /machinegroup
```

请求头

除公共响应头部外，无特殊响应头部。

请求参数

字段名	类型	位置	必须	含义
group_id	string	query	是	查询的 group ID

响应

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 123

{
"group_id": "xxxx-xx-xx-xx-xxxxxxxx",
"group_name": "testname",
"type":"label",
```

```
"labels": [
"defined_label_1",
"defined_label_2"
],
"create_time": "2017-08-08 12:12:12"
}
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

字段名	类型	必有	含义
group_id	string	是	机器组的 ID
group_name	string	是	机器组的名字
type	string	是	机器组类型
ips	JsonArray	否	ip 机器组下的 IP 列表
labels	JsonArray	否	label 机器组的标签表
create_time	string	否	创建时间

> **注意：**
> ips 和 labels 根据 type 至少返回一个。

错误码

参见 错误码。

获取机器状态

最近更新时间：2020-08-10 15:33:51

功能描述

本接口用于获取指定机器组下的机器状态。

请求

请求示例

```
GET /machines?group_id=xxxx-xx-xx-xx-xxxxxxxx HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
```

请求行

```
GET /machines
```

请求头

除公共响应头部外，无特殊响应头部。

请求参数

字段名	类型	位置	必须	含义
group_id	string	query	是	查询的 group ID

响应

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 123

{
"machines": [
{"ip": "10.10.10.10","status": 0},
{"ip": "10.10.10.11","status": 1}
```

```
]
}
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

字段名	类型	必有	含义
machines	JsonArray	是	机器信息数组

MachineInfo 格式如下：

字段名	类型	必有	含义
ip	string	是	机器的 IP
status	int	是	0：异常 1：正常

错误码

参见 错误码。

获取机器组列表

最近更新时间：2020-08-10 15:32:12

功能描述

本接口用于获取机器组信息列表。

请求

请求示例

```
GET /machinegroups HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
```

请求行

```
GET /machinegroups
```

请求头

除公共头部外，无特殊请求头部。

请求参数

无。

响应

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 123

{
"machine_groups": [
{
"group_id": "xxxx-xx-xx-xx-xxxxxxxx",
"group_name": "testname",
"create_time": "2017-08-08 12:12:12"
```

```
}
]
]
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

字段名	类型	必有	含义
machine_groups	JsonArray	是	机器组信息数组

MachineGroupInfo 格式如下：

字段名	类型	必有	含义
group_id	string	是	机器组的 ID
group_name	string	是	机器组的名字
create_time	string	否	创建时间

错误码

参见 错误码。

修改机器组

最近更新时间：2020-10-14 09:48:52

功能描述

本接口用于修改机器组。

请求

请求示例

```
PUT /machinegroup HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
Content-Type: application/json

{
"group_id": "xxxx-xx-xx-xx-xxxxxxx",
"group_name": "testname",
"type" :  "ip",
"ips": ["10.10.10.10", "10.10.10.11"]
}
```

请求行

```
PUT /machinegroup
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	必须	含义
group_id	string	body	是	要修改的机器组的 ID
group_name	string	body	否	机器组的名字，不能重名
type	string	body	否	机器组类型，只支持 ip 和 label 两种类型，默认是 ip
ips	JsonArray	body	否	ip 机器组下的 IP 列表

字段名	类型	位置	必须	含义
labels	JsonArray	body	否	label 机器组的标签表

> **注意：**
> group_name 、 ips 和 labels 至少要提供一个。

响应

响应示例

```
HTTP/1.1 200 OK
Content-Length: 0
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

无。

错误码

参见 错误码 文档。

删除机器组

最近更新时间：2020-08-10 15:34:44

功能描述

本接口用于删除机器组。

请求

请求示例

```
DELETE /machinegroup?group_id=xxxx-xx-xx-xx-xxxxxxxx HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
```

请求行

```
DELETE /machinegroup
```

请求头

除公共响应头部外，无特殊响应头部。

请求参数

字段名	类型	位置	必须	含义
group_id	string	query	是	要删除的机器组的 ID

响应

响应示例

```
HTTP/1.1 200 OK
Content-Length: 0
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

无。

错误码

参见 错误码。

消费管理
创建消费组

最近更新时间：2020-07-31 11:09:31

功能描述

本接口用于创建消费组。

请求

请求示例

```
POST /consumergroup?topic_id=xxxx-xx-xx-xx-xxxx HTTP/1.1
Host: <Region>.cls.tencentyun.com
Content-Type: application/json
Authorization: <AuthorizationString>

{"consumer_group": "cls_demo_consumer_group", "timeout": 3600, "order": false}
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	是否必须	含义
topic_id	string	query	是	日志主题 ID，在该日志主题下创建消费组
consumer_group	string	body	是	消费组名称，字符长度为1至255个字符，允许的字符为a-z、A-Z、0-9、_、-
timeout	int	body	是	消费组超时时间（单位：秒），超过 timeout 秒后未收到任何心跳，系统会删除消费组
order	bool	body	是	该配置会影响主题分区分裂/合并时的消费行为： • true：按顺序消费 • false：不按顺序消费

响应

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 0
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

无。

错误码

参见 错误码。

获取消费游标

最近更新时间：2020-07-31 11:11:06

功能描述

本接口根据时间获取对应主题分区的游标（cursor），该游标用于获取对应主题分区上的日志数据。

请求

请求示例

```
GET /cursor?topic_id=xxxxxxxx-xxxx-xxxx-xxxx&partition_id=1&from=end HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
```

请求头

除公共头部外，无特殊请求头部。

请求参数

参数名	类型	位置	是否必须	描述
topic_id	string	query	是	日志主题 ID
partition_id	int	query	是	主题分区编号
from	string	query	是	from 用于标识实时消费的开始时间，支持三种类型： 1. "UNIX 时间戳（秒）"，表示从指定 UNIX 时间开始消费日志 2. "start"，表示从主题分区生命周期的开始时间开始消费日志 3. "end"，表示从主题分区生命周期的结束时间（当前时间）消费日志

主题分区生命周期说明

主题分区的数据生命周期由 CLS 后台系统设置，不低于1天（不同的日志主题下的主题分区数据生命周期会不同）。

例如，当前时间为2019-10-10 12:00:00，则每个主题分区中数据可以被消费的时间范围（以服务端时间为准）为：[2019-10-09 12:00:00, 2019-10-10 12:00:00)。

通过 from 可以在主题分区中定位实时消费的起始位置，假设主题分区的生命周期为 [start_time, end_time)：

- 当from（UNIX 时间戳）≤ start_time 或 from = "start"，接口返回时间点为 start_time 所对应的游标位置。
- 当from（UNIX 时间戳）≥ end_time 或 from = "end"，接口返回在当前时间点下，下一条将被写入的游标位置（当前该游标位置上无数据）。

- 当from（UNIX 时间戳）> start_time and from（UNIX 时间戳）< end_time，接口返回第一个服务端接收时间大于等于from（UNIX 时间戳）的数据包对应的游标位置。

响应

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 23

{"cursor": "MTQ0NzI5OTYwNjg5NjYzMjM1Ng=="}
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

字段名	类型	是否必有	含义
cursor	string	是	返回的游标值

错误码

参见 错误码。

获取消费组游标

最近更新时间：2020-07-31 11:05:26

功能描述

本接口用于获取消费组游标。

请求

请求示例

```
GET /consumergroupcursor?topic_id=xxxx-xx-xx-xx-xxxx&consumer_group=cls_demo_consumer_grou
p&partition_id=1 HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	是否必须	含义
topic_id	string	query	是	消费组所属的日志主题 ID
consumer_group	string	query	是	消费组名称
partition_id	int	query	否	主题分区编号（如果不指定，则返回该主题下所有分区的游标）

响应

响应示例

未指定 partition_id 响应回包：

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 123

{
"cursors":[
```

```
{
"consumer_id":"cls-demo_consumer_id",
"cursor":"FAjUjMtmELBovQRogYkBuq",
"partition_id":1,
"update_time":1573645058
},
{
"consumer_id":"cls-demo_consumer_id",
"cursor":"FAjUjMtmELBovQRogYkBuqg",
"partition_id":2,
"update_time":1573645058
}
]
}
```

指定 partition_id 响应回包：

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 123

{
"consumer_id":"cls-demo_consumer_id",
"cursor":"FAjUjMtmELBovQRog",
"partition_id":1,
"update_time":1573645058
}
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

字段名	类型	是否必有	含义
partition_id	string	是	主题分区编号
cursor	string	是	游标值
update_time	long long	是	游标更新的时间
consumer_id	string	是	该分区被分配的消费者 ID

错误码

参见 错误码。

消费数据

最近更新时间：2020-07-31 11:10:19

功能描述

本接口用于消费读取日志。根据游标（cursor）、数量（count）获取对应主题分区上的日志数据。

请求

请求示例

```
GET /pulllogs?topic_id=xxxxxxxx-xxxx-xxxx-xxxx&partition_id=1&cursor=xxxxxxxxx&count=10 HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
```

请求头

除公共头部外，无特殊请求头部。

请求参数

参数名	类型	位置	是否必须	描述
topic_id	string	query	是	日志主题 ID
partition_id	int	query	是	消费的主题分区编号
cursor	string	query	是	游标值，base64 编码，表示从当前位置开始读取数据
count	int	query	是	单次消费的 LogGroup 个数，最多1000个（一个 LogGroup 会包含多条日志，LogGroup 定义参考 上传结构化日志 文档）

响应

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/x-protobuf
Content-Length: 23
x-cls-cursor: xxxxxx
x-cls-count: 10
```

```
<LogGroupList 的pb格式打包内容>
```

响应头

Header名	含义
x-cls-cursor	游标值，base64 编码，表示下次从当前位置开始读取数据，供继续消费使用
x-cls-count	当前请求返回的 LogGroup 个数（一个 LogGroup 会包含多条日志，LogGroup 定义参考 上传结构化日志 文档）

响应参数

返回 LogGroupList 对象的打包内容，pb 文件描述详见 上传结构化日志 接口。

错误码

参见 错误码。

消费者心跳

最近更新时间：2020-11-05 09:27:55

功能描述

本接口用于消费者上传心跳。

请求

请求示例

```
POST /consumerheartbeat?topic_id=xxxx-xx-xx-xx-xxxx HTTP/1.1
Host: <Region>.cls.tencentyun.com
Content-Type: application/json
Authorization: <AuthorizationString>

{"consumer_group": "cls_demo_consumer_group", "consumer_id": "consumer_id", "partition_id_
list": []}
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	是否必须	含义
topic_id	string	query	是	消费组所属日志主题的 ID
consumer_group	string	body	是	消费组名称
consumer_id	string	body	是	消费者名称
partition_id_list	array	body	是	消费者消费的分区列表

响应

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 123
```

```
{
"partition_id_list":[
4,
5
]
}
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

字段名	类型	是否必有	含义
partition_id_list	array	是	该消费者可消费的分区列表

错误码

参见 错误码。

获取消费组列表

最近更新时间：2020-07-31 11:03:54

功能描述

本接口用于获取日志主题的消费组列表。

请求

请求示例

```
GET /consumergroups?topic_id=xxxx-xx-xx-xx-xxxxxxxx HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	是否必须	含义
topic_id	string	query	是	查询日志主题的 ID

响应

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 123

{
"consumer_groups":[
{
"consumer_group":"cls-demo_consumer_group",
"order":true,
"timeout":3600
}
]
}
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

字段名	类型	是否必有	含义
consumer_group	string	是	消费组名称
timeout	int	是	消费组超时时间，超过 timeout 秒后未收到任何心跳，系统会删除消费组
order	bool	是	是否按顺序消费

错误码

参见 错误码。

修改消费组

最近更新时间：2020-07-31 11:03:01

功能描述

本接口用于修改消费组。

请求

请求示例

```
PUT /consumergroup?topic_id=xxxx-xx-xx-xx-xxxxxxxx&consumer_group=xxxxx HTTP/1.1
Host: <Region>.cls.tencentyun.com
Content-Type: application/json
Authorization: <AuthorizationString>

{"timeout": 3600, "order": true}
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	是否必须	含义
topic_id	string	query	是	消费组所属的日志主题 ID
consumer_group	string	query	是	消费组名称
timeout	int	body	否	消费组超时时间，超过 timeout 秒未收到任何心跳，系统将删除消费组
order	bool	body	否	是否按顺序消费

响应

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 0
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

无。

错误码

参见 错误码。

修改消费组游标

最近更新时间：2020-07-31 11:02:18

功能描述

本接口用于更新消费组游标。

请求

请求示例

```
PUT /consumergroupcursor?topic_id=xxxx-xx-xx-xx-xxxx&consumer_group=cls_demo_consumer_grou
p&partition_id=1 HTTP/1.1
Host: <Region>.cls.tencentyun.com
Content-Type: application/json
Authorization: <AuthorizationString>

{"consumer_id": "cls_demo_consumer_1", "cursor": "FAjUjMtmELBo"}
```

请求行

```
PUT /consumergroupcursor
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	是否必须	含义
topic_id	string	query	是	消费组所属的日志主题 ID
consumer_group	string	query	是	消费组名称
partition_id	int	query	是	主题分区编号
consumer_id	string	body	否	消费者名称
cursor	string	body	是	游标值

响应

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 123
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

无。

错误码

参见 错误码。

删除消费组

最近更新时间：2020-07-31 11:06:04

功能描述

本接口用于删除消费组。

请求

请求示例

```
DELETE /consumergroup?topic_id=xxxx-xx-xx-xx-xxxx&consumer_group=cls_demo_consumer_group H
TTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	是否必须	含义
topic_id	string	query	是	消费组所属的日志主题的 ID
consumer_group	string	query	是	消费组名称

响应

响应示例

```
HTTP/1.1 200 OK
Content-Length: 0
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

无。

错误码

参见 错误码。

索引管理
获取索引信息

最近更新时间：2020-08-10 15:30:41

功能描述

本接口用于获取指定索引策略的详细信息。

请求

请求示例

```
GET /index?topic_id=xxxx-xx-xx-xx-xxxxxxxx HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
```

请求行

```
GET /index
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	必须	含义
topic_id	string	query	是	查询的 index 所属的 topic ID

响应

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 153

{
"topic_id": "yyyy-yy-yy-yy-yyyyyyyy",
```

```
"effective": true,
"rule": {
"full_text": {
"case_sensitive": false
},
"key_value": {
"case_sensitive": false,
"keys": ["age","name"],
"types": ["long","text"]
}
}
}
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

字段名	类型	必有	含义
topic_id	string	是	索引规则属于的 topic ID
effective	bool	是	是否生效
rule	object	否	索引规则，当 effective 为 true 时返回

rule 内容说明：

字段名	类型	必有	含义
full_text	object	否	全文索引的相关配置
key_value	object	否	kv 索引的相关配置

full_text 内容说明：

字段名	类型	必有	含义
case_sensitive	bool	是	是否大小写敏感

key_value 内容说明：

字段名	类型	必有	含义

字段名	类型	必有	含义
case_sensitive	bool	是	是否大小写敏感
keys	array(string)	是	需要建索引的 key 的名字
types	array(string)	是	上面 key 对应的类型，一一对应，目前支持`long double text`

错误码

参见 错误码。

修改索引任务

最近更新时间：2020-08-10 15:31:33

功能描述

本接口用于修改现有的索引任务。

请求

请求示例

```
PUT /index HTTP/1.1
Host: <Region>.cls.tencentyun.com
Authorization: <AuthorizationString>
Content-Type: application/json

{
"topic_id": "xxxx-xx-xx-xx-xxxxxxxx",
"effective": true,
"rule": {
"full_text": {
"case_sensitive": false,
"tokenizer": "{^&%"
},
"key_value": {
"case_sensitive": false,
"keys": ["age","name"],
"types": ["long","text"],
"tokenizers": ["","-"]
}
}
}
```

请求行

```
PUT /index
```

请求头

除公共头部外，无特殊请求头部。

请求参数

字段名	类型	位置	必须	含义
topic_id	string	body	是	修改的 index 属于的 topic ID
effective	bool	body	是	index 的开关状态
rule	object	body	否	索引规则，当 effective 为 true 时必需

rule 内容说明：

字段名	类型	必须	含义
full_text	object	否	全文索引的相关配置
key_value	object	否	kv 索引的相关配置

> 注意：
> 设置 rule 时，full_text、key_value 两者至少要设置一个。

full_text 内容说明：

字段名	类型	必须	含义
case_sensitive	bool	是	是否大小写敏感
tokenizer	string	否	全文索引的分词符，不允许为空，建议设置为`!@#%^&*()-_='",<>/?|\;:\n\t\r[]{}`

key_value 内容说明：

字段名	类型	必须	含义
case_sensitive	bool	是	是否大小写敏感
keys	array(string)	是	需要建索引的 key 的名字
types	array(string)	是	需要建索引 的 key 对应的类型，一一对应，目前支持`long double text`
tokenizers	array(string)	否	上面 key 对应的分词符，一一对应，只对`text`类型设置，其他类型为空字符串

响应

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 0
```

响应头

除公共响应头部外，无特殊响应头部。

响应参数

无。

错误码

参见 错误码。

错误码

最近更新时间：2020-08-05 15:02:32

功能说明

本文为您介绍 API 请求出错时返回的错误码和对应错误信息。您可以根据 HTTP 的 StatusCode 和 Body 来确定问题。其中 Body 的格式如下：

```
{
"errorcode" : "<ErrorCode>",
"errormessage" : "<ErrorMessage>"
}
```

错误码列表

HTTP 状态码（StatusCode）	错误码（ErrorCode）	描述（ErrorMessage）
400	InvalidAuthorization	签名串格式不合法
400	InvalidCompressType	指定的 x-cls-compress-type 不支持
400	InvalidContent	消息体错误，解压失败或者解析失败
400	InvalidContentType	指定的 Content-Type 不支持
400	InvalidParam	缺少必要参数或者个别参数不合法
400	MissingAgentIp	缺少 x-cls-agent-ip
400	MissingAgentVersion	缺少 x-cls-agent-version
400	MissingAuthorization	缺少 Authorization
400	MissingContent	消息体是空的
400	MissingContentType	缺少 Content-Type
400	TopicClosed	指定的日志主题已经关闭收集功能
400	IndexRuleEmpty	指定的日志主题没有设置索引规则
400	LogsetNotEmpty	指定的日志集非空，含有日志主题
400	SyntaxError	检索语法错误

HTTP 状态码（StatusCode）	错误码（ErrorCode）	描述（ErrorMessage）
400	LogsetEmpty	指定的日志集为空，不包含任何日志主题
401	AuthFailure.SecretIdNotFound	密钥不存在。 请在控制台检查密钥是否已被删除或者 禁用，如状态正常，请检查密钥是否填写正确，注意前后不得有空格
401	AuthFailure.SignatureFailure	签名错误。 签名计算错误，请对照调用方式中的签名 方法文档检查签名计算过程
401	AuthFailure.SignatureExpire	签名过期
401	AuthFailure.MFAFailure	MFA 错误
401	AuthFailure.UnauthorizedOperation	请求未授权。请参考 CAM 文档对鉴权的说明
401	AuthFailure.InvalidSecretId	密钥非法（不是云 API 密钥类型）
401	AuthFailure.TokenFailure	token 错误
401	AuthFailure.Unauthorized	鉴权内部错误
403	LogsetExceed	日志集数量超出限制，最多20个
403	LogSizeExceed	提交的日志超出最大限制，最大5MB
403	MachineGroupExceed	机器组数量超出限制，最多200个
403	NotAllowed	不允许此操作
403	TopicExceed	日志主题数量超出限制，最多10个
403	ShipperExceed	投递规则数量超出限制，最多10个
403	TaskReadOnly	只有失败的投递任务才能重启，其他状态时不能被修改
403	AccountArrears	账户欠费
403	ServiceNotActivated	日志服务未开通
404	CursorNotExist	指定的位置没有可以下载的日志
404	TaskNotExist	指定的投递任务不存在
404	IndexNotExist	指定的索引规则不存在

HTTP 状态码（StatusCode）	错误码（ErrorCode）	描述（ErrorMessage）
404	LogsetNotExist	指定的日志集不存在
404	MachineGroupNotExist	指定的机器组不存在
404	ShipperNotExist	指定的投递规则不存在
404	TopicNotExist	指定的日志主题不存在
404	ConsumerNotExist	指定的日志主题不存在消费任务
405	NotSupported	不支持此操作
409	IndexConflict	相同的索引规则已存在
409	LogsetConflict	相同的日志集已存在
409	MachineGroupConflict	相同的机器组已存在
409	ShipperConflict	相同的投递规则已存在
409	TopicConflict	相同的日志主题已存在
409	ConsumerConflict	日志主题的消费任务已存在
429	SpeedQuotaExceed	请求过于频繁
500	InternalError	内部错误