ALIBABA CLOUD

阿里云

日志服务

API 参考

文档版本：20210521

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.概览

日志服务除了通过管理控制台进行操作外，还提供了API（Application Programming Interface）方式写入、查询日志数据，管理项目及日志库等。

目前开放如下API:

对象	方法
服务操作	OpenSlsService、GetSlsService
Project（项目）	ListProject、CreateProject、DeleteProject、GetProject、UpdateProject
	GetProjectLogs（统计Project下所有日志）
Config（Logtail配置）	ListConfig、CreateConfig、DeleteConfig、GetConfig、UpdateConfig
	GetAppliedMachineGroups（查询应用到的机器组）
MachineGroup（机器组）	ListMachineGroup、CreateMachineGroup、DeleteMachineGroup、GetMachineGroup、UpdateMachineGroup
	ApplyConfigToMachineGroup、RemoveConfigFromMachineGroup（删除配置）
	GetAppliedConfigs（查询已应用配置列表）
Logstore（日志库）	ListLogstore、CreateLogstore、DeleteLogstore、GetLogstore、UpdateLogstore
	GetLogs（查询日志）、GetHistograms（查询日志分布）
Index（索引）	CreateIndex、UpdateIndex、DeleteIndex、GetIndex
Shard（分区）	ListShards、SplitShard、MergeShards
	PutLogs（写入日志）
	GetCursor（定位日志位置）
	PullLogs（消费日志）
Shipper（日志投递规则）	GetShipperStatus（查询日志投递任务状态）
	RetryShipperTask（重试失败投递任务）
ConsumerGroup（消费组）	CreateConsumerGroup、UpdateConsumerGroup、DeleteConsumerGroup、ListConsumerGroup
	HeartBeat（发送心跳）、GetCheckPoint、UpdateCheckPoint
Alert（告警）	CreateAlert、UpdateAlert、DeleteAlert、GetAlert、ListAlert
Savedsearch（保存查询）	CreateSavedsearch、UpdateSavedsearch、DeleteSavedsearch、GetSavedsearch、ListSavedsearch

通过API可以进行如下操作：

- 根据Logtail配置、Logtail机器组信息采集日志。
- 创建日志库，向日志库写入、读取日志。
- 对不同用户进行访问控制。

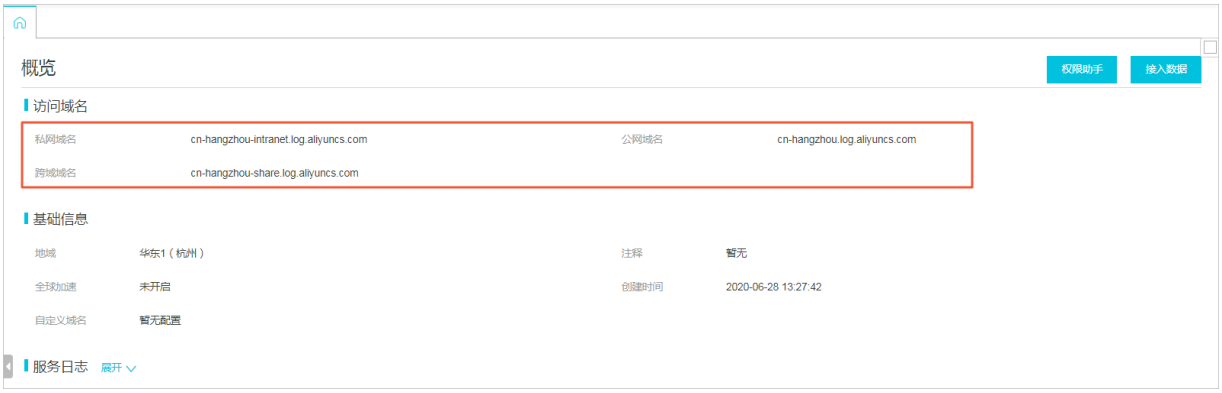
> 说明
>
> - 为更好的使用API，您需要了解服务入口。
> - API所有请求都需要做安全验证，请求签名解释了具体的API请求签名机制及流程。
> - 日志服务支持RAM、STS，RAM用户使用API时，除需要使用RAM用户的密钥进行签名验证外，其他和一般阿里云账号没有区别。STS临时身份除需要临时密钥信息外，还需要填写一个特殊的HTTP Header，请参见公共请求头，这个HTTP Header需要进行签名，请参见请求签名。

2.服务入口

本文介绍了日志服务不同网络类型的服务入口。

概述

日志服务提供各种内网、公网等网络，不同网络的接入方式请参见使用Logtail收集各网络日志数据。您可以在Project的**概览**页面，查看该Project所在地域的服务入口，如下图所示。



公网服务入口

日志服务入口是访问一个项目（Project）及其内部日志数据的URL。服务入口与Project所在的地域及Project名称相关。目前，日志服务已经在多个地域开通，在各地域内的公网服务入口如下表所示。

> **说明** 华北5（呼和浩特）和中国（香港）地域的公网服务入口已经支持通过IPv6进行访问，其他地域会陆续开放。

地域	服务入口
华东1（杭州）	cn-hangzhou.log.aliyuncs.com
华东1（杭州-金融云）	cn-hangzhou-finance.log.aliyuncs.com
华东2（上海）	cn-shanghai.log.aliyuncs.com
华东2（上海-金融云）	cn-shanghai-finance-1.log.aliyuncs.com
华北1（青岛）	cn-qingdao.log.aliyuncs.com
华北2（北京）	cn-beijing.log.aliyuncs.com
华北2 阿里政务云1	cn-north-2-gov-1.log.aliyuncs.com
华北3（张家口）	cn-zhangjiakou.log.aliyuncs.com
华北5（呼和浩特）	cn-huhehaote.log.aliyuncs.com
华北6（乌兰察布）	cn-wulanchabu.log.aliyuncs.com
华南1（深圳）	cn-shenzhen.log.aliyuncs.com
华南1（深圳-金融云）	cn-shenzhen-finance.log.aliyuncs.com
华南2（河源）	cn-heyuan.log.aliyuncs.com

地域	服务入口
华南3（广州）	cn-guangzhou.log.aliyuncs.com
西南1（成都）	cn-chengdu.log.aliyuncs.com
中国（香港）	cn-hongkong.log.aliyuncs.com
日本（东京）	ap-northeast-1.log.aliyuncs.com
新加坡	ap-southeast-1.log.aliyuncs.com
澳大利亚（悉尼）	ap-southeast-2.log.aliyuncs.com
马来西亚（吉隆坡）	ap-southeast-3.log.aliyuncs.com
印度尼西亚（雅加达）	ap-southeast-5.log.aliyuncs.com
阿联酋（迪拜）	me-east-1.log.aliyuncs.com
美国（硅谷）	us-west-1.log.aliyuncs.com
德国（法兰克福）	eu-central-1.log.aliyuncs.com
美国（弗吉尼亚）	us-east-1.log.aliyuncs.com
印度（孟买）	ap-south-1.log.aliyuncs.com
英国（伦敦）	eu-west-1.log.aliyuncs.com
俄罗斯（莫斯科）	rus-west-1.log.aliyuncs.com

当访问某个Project时，需要根据Project名称及其所在地域组合出最终访问地址。格式如下：

```
<project_name>.<region_endpoint>
```

例如，Project名为big-game，所在地域为华东1（杭州），则对应访问地址如下：

```
big-game.cn-hangzhou.log.aliyuncs.com
```

> 说明
> - 在创建日志服务项目时需要指定地域。
> - 当创建Project时指定了地域，该设置就不可以更改，且无法跨地域迁移。
> - 创建Project之后，必须选择与其所在地域相匹配的服务入口地址来组成该Project的访问地址，用做API请求的服务入口。

经典网络及VPC网络服务入口

如果在阿里云ECS机器环境使用日志服务API，还可以使用内网服务入口，各个地域服务入口如下表所示。

> 说明
> - 日志服务API在如下服务入口仅支持HTTP协议、HTTPS协议。
> - 使用内网服务入口访问日志服务不消耗ECS公网流量，可以节约ECS公网带宽。

地域	根服务入口
华东1（杭州）	cn-hangzhou-intranet.log.aliyuncs.com
华东1（杭州-金融云）	cn-hangzhou-finance-intranet.log.aliyuncs.com
华东2（上海）	cn-shanghai-intranet.log.aliyuncs.com
华东2（上海-金融云）	cn-shanghai-finance-1-intranet.log.aliyuncs.com
华北1（青岛）	cn-qingdao-intranet.log.aliyuncs.com
华北2（北京）	cn-beijing-intranet.log.aliyuncs.com
华北2 阿里政务云1	cn-north-2-gov-1-intranet.log.aliyuncs.com
华南1（深圳）	cn-shenzhen-intranet.log.aliyuncs.com
华南1（深圳-金融云）	cn-shenzhen-finance-intranet.log.aliyuncs.com
华南2（河源）	cn-heyuan-intranet.log.aliyuncs.com
华南3（广州）	cn-guangzhou-intranet.log.aliyuncs.com
华北3（张家口）	cn-zhangjiakou-intranet.log.aliyuncs.com
华北5（呼和浩特）	cn-huhehaote-intranet.log.aliyuncs.com
华北6（乌兰察布）	cn-wulanchabu-intranet.log.aliyuncs.com
西南1（成都）	cn-chengdu-intranet.log.aliyuncs.com
中国（香港）	cn-hongkong-intranet.log.aliyuncs.com
美国（硅谷）	us-west-1-intranet.log.aliyuncs.com
日本（东京）	ap-northeast-1-intranet.log.aliyuncs.com
新加坡	ap-southeast-1-intranet.log.aliyuncs.com
澳大利亚（悉尼）	ap-southeast-2-intranet.log.aliyuncs.com
马来西亚（吉隆坡）	ap-southeast-3-intranet.log.aliyuncs.com
印度尼西亚（雅加达）	ap-southeast-5-intranet.log.aliyuncs.com
阿联酋（迪拜）	me-east-1-intranet.log.aliyuncs.com
德国（法兰克福）	eu-central-1-intranet.log.aliyuncs.com
美国（弗吉尼亚）	us-east-1-intranet.log.aliyuncs.com
印度（孟买）	ap-south-1-intranet.log.aliyuncs.com
英国（伦敦）	eu-west-1-intranet.log.aliyuncs.com
俄罗斯（莫斯科）	rus-west-1-intranet.log.aliyuncs.com

例如，Project名为big-game，所在地域为华东1（杭州），则对应访问地址如下：

big-game.cn-hangzhou-intranet.log.aliyuncs.com

全球加速服务入口

日志服务在内网和公网基础上，新增全球加速公网的网络类型。相较于普通的公网访问，全球加速公网在延时和稳定性上具备显著优势，适用于对数据采集、消费延时、可靠性要求较高的场景。

全球加速公网的服务入口在所有地域一致，服务入口为：log-global.aliyuncs.com

> **说明** 全球加速功能默认为关闭状态，需要手动开启后才可使用。开启全球加速，请参见步骤一：开启全球加速服务。

3.访问密钥

阿里云访问密钥（AccessKey）是调用API访问云资源的安全口令，您可以使用AccessKey签名API请求内容以通过服务端的安全验证。本文介绍如何获取AccessKey。

背景信息

AccessKey包括AccessKey ID和AccessKey Secret。

- AccessKey ID：用于标识用户。
- AccessKey Secret：用于验证用户的密钥。AccessKey Secret必须保密。

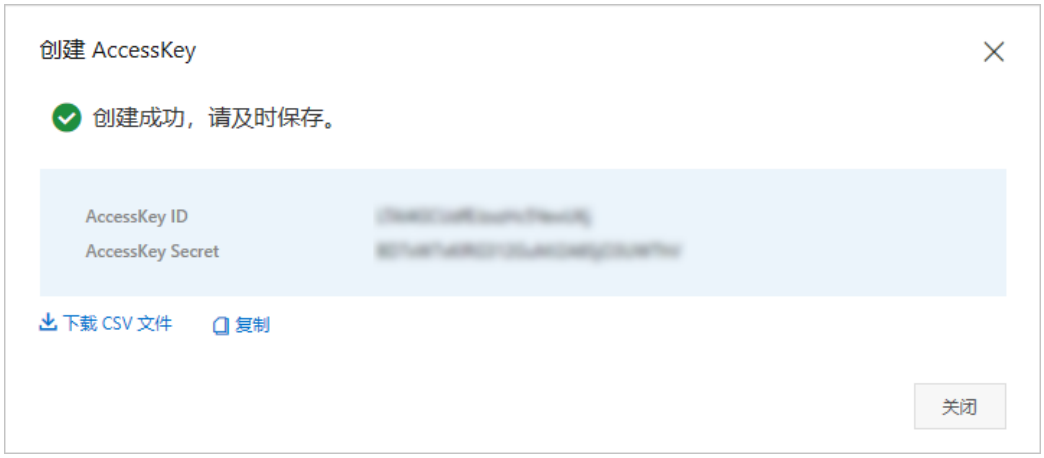
> 警告
> - 阿里云账号AccessKey泄露会威胁您所有资源的安全。建议您使用RAM用户AccessKey进行操作，有效降低AccessKey泄露的风险。
> - 如果某密钥对出现泄漏风险，建议及时删除该密钥对并生成新的密钥对。

操作步骤

1. 使用阿里云账号登录控制台。
2. 将鼠标置于页面右上方的账号图标，单击**AccessKey管理**。
3. 在**安全提示**对话框，选择使用阿里云账号AccessKey或RAM用户AccessKey。

   - 使用阿里云账号AccessKey
     a. 单击**继续使用AccessKey**。
     b. 在**AccessKey管理**页面，单击**创建AccessKey**。
     c. 获取验证码，单击**确定**。

d. 在**创建AccessKey**对话框，查看AccessKey ID和AccessKey Secret。可以单击**下载CSV文件**，下载AccessKey信息。或者单击**复制**，复制AccessKey信息。



- 使用RAM用户AccessKey
  a. 单击**开始使用子用户AccessKey**。
  b. 系统自动跳转到RAM控制台的**用户**页面，找到需要获取AccessKey的RAM用户。

> **说明** 如果没有RAM用户，请先创建RAM用户，详情请参见创建RAM用户。

  c. 单击用户登录名称。
  d. 在**认证管理**页签下的**用户AccessKey**区域，单击**创建AccessKey**。
  e. 获取验证码，单击**确定**。
  f. 在**创建AccessKey**页面，查看AccessKey ID和AccessKey Secret。可以单击**下载CSV文件**，下载AccessKey信息。或者单击**复制**，复制AccessKey信息。

> **说明**
> - RAM用户的AccessKey Secret只在创建时显示，不提供查询，请妥善保管。
> - 若AccessKey泄露或丢失，则需要创建新的AccessKey，最多允许为每个RAM用户创建2个AccessKey。

4.公共请求头

本文列举了日志服务API的公共请求头信息。

参数列表

日志服务API是基于HTTP协议的REST风格接口。它支持一组可以在所有API请求中使用的公共请求头，其详细说明如下：

Header名称	类型	是否必须	说明
Accept	字符串	否	客户端希望服务端返回的类型，目前支持application/json、application/x-protobuf两种。仅对GET请求有效，具体取值以各个接口定义为准。
Accept-Encoding	字符串	否	客户端希望服务端返回的压缩算法，目前支持lz4、deflate或空（不压缩）。仅对GET类请求有效，具体取值以各个接口定义为准。
Authorization	字符串	是	签名内容，更多细节请参考 请求签名。
Content-Length	数值	否	RFC 2616中定义的HTTP请求Body长度。如果请求无Body部分，则不需要提供该请求头。
Content-MD5	字符串	否	请求Body经过MD5计算后的字符串，计算结果为大写。如果没有Body部分，则不需要提供该请求头。
Content-Type	字符串	否	RFC 2616中定义的HTTP请求Body类型。目前日志服务API请求只支持application/x-protobuf类型。如果没有Body部分，则不需要提供该请求头。具体取值以各个接口定义为准。
Date	字符串	是	当前发送时刻的时间，参数目前只支持RFC 822格式，使用GMT标准时间。格式化字符串如下：%a,%d%b%Y %H:%M:%S GMT （如：Mon, 3 Jan 2010 08:33:47 GMT）。
Host	字符串	是	HTTP请求的完整HOST名称（不包括如 `http://` 这样的协议头）。例如，`big-game.cn-hangzhou.sls.aliyuncs.com` 。
x-log-apiversion	字符串	是	API的版本号，当前版本为0.6.0。
x-log-bodyrawsize	数值	否	请求的Body原始大小。当无Body时，该字段为0；当Body是压缩数据，则为压缩前的原始数据大小。该域取值范围为0~3x1024x1024。该字段只在压缩时需要。
x-log-compresstype	字符串	否	API请求中Body部分使用的压缩方式。目前支持lz4压缩类型和deflate压缩类型（RFC 1951，使用zlib格式，参考RFC 1950）。如果不压缩可以不提供该请求头。
x-log-date	字符串	否	当前发送时刻的时间，格式和Date头一致。如果请求中包含该公共请求头，它的值会取代Date标准头的值用于服务端请求验证（该字段不参与签名）。无论是否有x-log-date头，HTTP标准Date头都必须提供。
x-log-signaturemethod	字符串	是	签名计算方式，目前仅支持 `hmac-sha1` 。
x-acs-security-token	字符串	否	使用STS临时身份发送数据。当使用STS临时身份时必填，其他情况不要填写。

> 说明
>
> - 请求中Date所表示的时间与服务器接收到该请求的时间最大可接受误差为15分钟，如果超过15分钟服务器端会拒绝该请求。如果请求中设置了x-log-date头部，则该时间误差计算基于x-log-date头的值。
> - 如果请求指明了压缩算法（在x-log-compresstype中指定），则需要把原始数据压缩后放到HTTP Body部分，而对应的Content-Length、Content-MD5头部也是按照压缩后的Body部分计算。
> - 由于某些平台发送HTTP请求时无法指定Date头（由平台自身的库内部自动指定为发送当前时间），造成无法使用正确的Date值计算请求签名。在这种情况下，请指定x-log-date头并用该请求头的值参与请求签名计算。日志服务端在接收到API请求后会首先判断是否有x-log-date头。如果有，则用它的值来做签名验证，否则就用HTTP的标准头Date做签名验证。

示例

```
POST / HTTP/1.1
Authorization: LOG <yourAccessKeyId>:<yourSignature>
x-log-bodyrawsize: 0
User-Agent: sls-java-sdk-v-0.6.1
x-log-apiversion: 0.6.0
Host: my-project-test.cn-shanghai.log.aliyuncs.com
x-log-signaturemethod: hmac-sha1
Date: Sun, 27 May 2018 07:43:26 GMT
Content-Type: application/json
Content-MD5: A7967D81EFF5E3CD447FB6D8DF294E20
Content-Length: 80
Connection: Keep-Alive
```

5.公共响应头

本文列举了日志服务的API的公共响应头信息。

参数列表

Log Service API是基于HTTP协议的Rest风格接口。所有的Log Service API响应都提供一组公共响应头，其详细定义如下：

Header名称	类型	说明
Content-Length	数值	RFC 2616中定义的HTTP响应内容长度。
Content-MD5	字符串	RFC 2616中定义的HTTP响应内容的MD5值。Body经过MD5计算后的字符串，为大写字符串。
Content-Type	字符串	RFC 2616中定义的HTTP响应内容类型。目前Log Service服务端响应类型支持application/json、application/x-protobuf两种类型。
Date	字符串	当前返回时刻的时间，参数目前只支持RFC 822格式，使用GMT标准时间。格式化字符串如下：%a, %d%b%Y %H:%M:%S GMT，如：Mon, 3 Jan 2010 08:33:47 GMT。
x-log-requestid	字符串	服务端产生的标示，该请求的唯一ID。该响应头与具体应用无关，主要用于跟踪和调查问题。如果用户希望调查出现问题的API请求，可以通过工单向日志服务团队提供该ID。

示例

```
HTTP/1.1 200
Server: nginx
Content-Length: 0
Connection: close
Access-Control-Allow-Origin: *
Date: Sun, 27 May 2018 07:43:27 GMT
x-log-requestid: 5B0A619F205DC3F30EDA9322
```

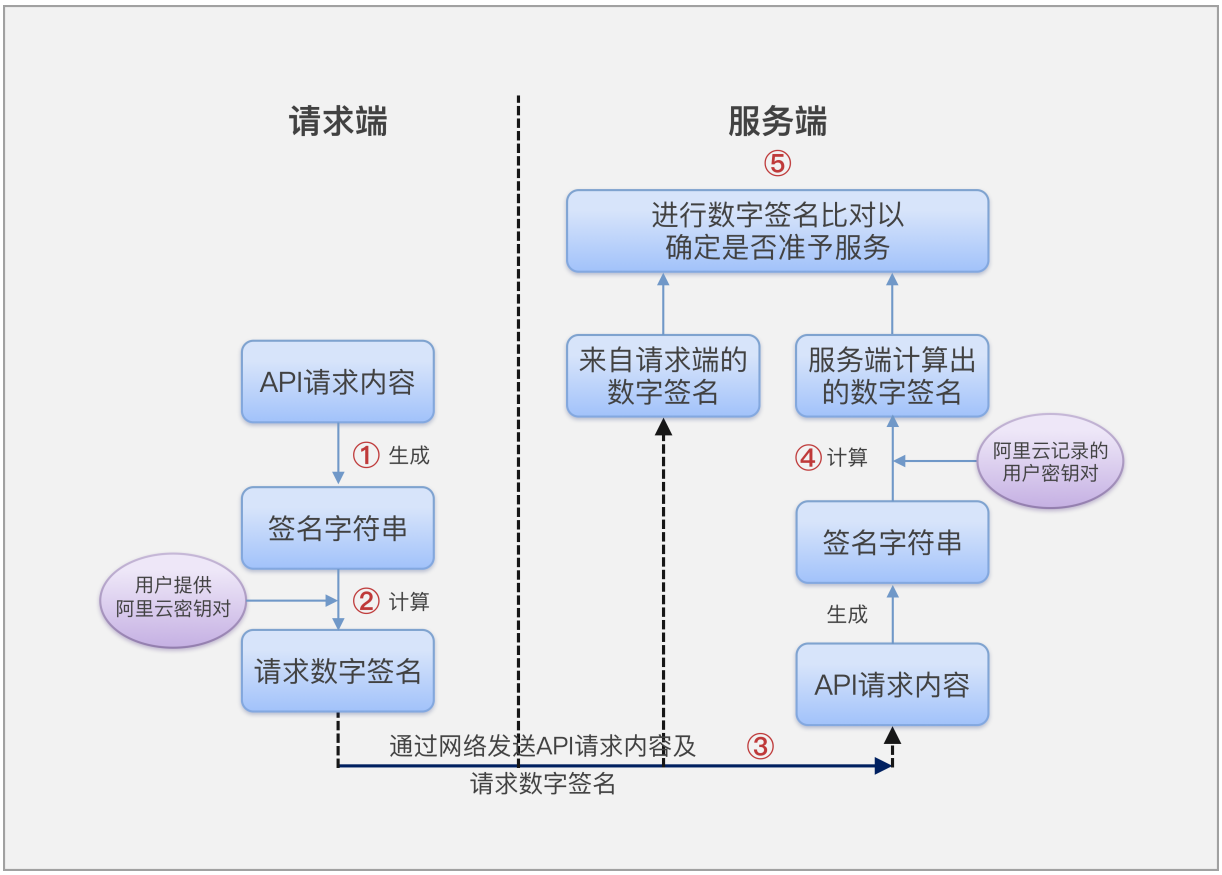
6.请求签名

为保证用户日志数据的安全，日志服务API的所有HTTP请求都必须经过安全验证。

验证流程

安全验证基于阿里云的访问密钥，使用对称加密算法完成。

验证流程图如下：



验证操作流程如下：

1. 请求端根据API请求内容（包括HTTP Header和Body）生成签名字符串。
2. 请求端使用阿里云的访问密钥对（AccessKeyID和AccessKeySecret），对步骤1生成的签名字符串进行签名，形成该API请求的数字签名。
3. 请求端把API请求内容和数字签名一同发送给服务端。
4. 服务端在接到请求后会重复如上的步骤1和步骤2的工作，并在服务端计算出该请求期望的数字签名。

   > **说明** 服务端会在后台取得该请求使用的用户访问密钥对。

5. 服务端用期望的数字签名和请求端发送过来的数字签名做比对，如果一致则认为该请求通过安全验证，否则直接拒绝该请求。

安全验证能达到以下目的：

- 确认哪位用户在做API请求。

  因为在发送请求前需要用户指定生成数字签名的密钥对，在服务端即可通过该密钥对确定用户身份，进而可做访问权限管理。

- 确认用户请求在网络传输过程中是否被篡改。

因为服务端会对接收到的请求内容重新计算数字签名，若请求内容在网络上被篡改，则无法通过数字签名比对。

签名API请求

为了通过API请求的安全验证，用户需要在客户端对其API请求进行签名（即生成正确的数字签名），并且使用HTTP头Authorization在网络上传输该请求的数字签名。Authorization头示例如下：

```
Authorization = "LOG" + AccessKeyId + ":" + Signature
```

如上格式所示，Authorization头的值包含用户访问密钥对中的AccessKeyId，且与之对应的AccessKeySecret将用于Signature值的构造。下面将详细解释如何构造该Signature值。

1. 准备阿里云访问密钥。为API请求生成签名，需使用一对访问密钥。您可以使用已经存在的访问密钥对，也可以创建新的访问密钥对，但需要保证使用的密钥对为**启用**状态。
2. 生成请求的签名字符串。日志服务API的签名字符串由HTTP请求中的Method、Header和Body信息一同生成，具体方式如下：

```
SignString = VERB + "\n"
        + CONTENT-MD5 + "\n"
        + CONTENT-TYPE + "\n"
        + DATE + "\n"
        + CanonicalizedLOGHeaders + "\n"
        + CanonicalizedResource
```

上面公式中的 `\n` 表示换行转义字符，加号（ `+` ）表示字符串连接操作，其他各个部分定义如下所示。

签名字符串定义

名称	说明	示例
`VERB`	HTTP请求的方法名称。	PUT、GET、POST等
`CONTENT-MD5`	HTTP请求中Body部分的MD5值（必须为大写字符串）。	875264590688CA6171F6228AF5BBB3D2
`CONTENT-TYPE`	HTTP请求中Body部分的类型。	application/x-protobuf
`DATE`	HTTP请求中的标准时间戳头（遵循RFC 1123格式，使用GMT标准时间）。	Mon,3 Jan 2010 08:33:47 GMT
`CanonicalizedLOGHeaders`	由HTTP请求中以 `x-log` 和 `x-acs` 为前缀的自定义头构造的字符串（具体构造方法见下面详述）。	x-log-apiversion:0.6.0\nx-log-bodyrawsize:50\nx-log-signaturemethod:hmac-sha1
`CanonicalizedResource`	由HTTP请求资源构造的字符串（具体构造方法见下面详述）。	/logstores/app_log

CONTENT-TYPE

对于部分无Body的HTTP请求，其CONTENT-MD5和CONTENT-TYPE两个域为空字符串，这时整个签名字符串的生成方式如下：

```
SignString = VERB + "\n"
        + "\n"
        + "\n"
        + DATE + "\n"
        + CanonicalizedLOGHeaders + "\n"
        + CanonicalizedResource
```

如公共请求头中的描述，日志服务API中引入了一个自定义请求头 `x-log-date` 。如果您在请求中指定了该请求头，则其值会替代HTTP标准请求头Date加入签名计算。

- CanonicalizedLOGHeaders的构造方式如下：
  a. 将所有以 `x-log` 和 `x-acs` 为前缀的HTTP请求头的名字转换成小写字母。
  b. 将上一步得到的所有LOG自定义请求头按照字典顺序进行升序排序。
  c. 删除请求头和内容之间分隔符两端出现的任何空格。
  d. 将所有的头和内容用 `\n` 分隔符组合成最后的CanonicalizedLOGHeader。
- CanonicalizedResource的构造方式如下：
  a. 将CanonicalizedResource设置为空字符串 `""` 。
  b. 放入要访问的LOG资源，如*/logstores/logstorename*（如果没有 `logstorename` 则可不填写）。
  c. 如果请求包含查询字符串 `QUERY_STRING` ，则在CanonicalizedResource字符串尾部添加 `?` 和查询字符串。

  `QUERY_STRING` 是URL中请求参数按字典顺序排序后的字符串，其中参数名和值之间用 `=` 相隔组成字符串，并对参数名-值对按照字典顺序升序排序，然后以 `&` 符号连接构成字符串。其公式化描述如下：

  ```
  QUERY_STRING = "KEY1=VALUE1" + "&" + "KEY2=VALUE2"
  ```

3. 生成请求的数字签名。目前，日志服务API只支持一种数字签名算法，即默认签名算法 `hmac-sha1` 。其完整签名公式如下：

   ```
   Signature = base64(hmac-sha1(UTF8-Encoding-Of(SignString)，AccessKeySecret))
   ```

   - 签名的方法用RFC 2104中定义的HMAC-SHA1方法。如上公式用的AccessKeySecret必须和最终的Authorization头中使用的AccessKeyId相对应。否则，请求将无法通过服务端验证。
   - 在计算出数字签名后，使用该值按本节最前面描述的Authorization头格式构建完整的Log Service API请求安全验证头，并填入HTTP请求中即可发送。
   - 签名的字符串必须为 `UTF-8` 格式。含有中文字符的签名字符串必须先进行 `UTF-8` 编码，再与 `AccessKeySecret` 计算最终签名。
   - `CONTENT-MD5` 和 `CONTENT-TYPE` 在请求中不是必须的，但是如果请求需要签名验证，空值的话以换行符 `\n` 代替。

请求签名过程示例

为了帮助您更好地理解整个请求签名的流程，我们用两个示例来演示整个过程。首先，假设您用做日志服务API签名的访问密钥对如下：

```
AccessKeyId = "bq2sjzesjmo86kq*********"
AccessKeySecret = "4fdO2fTDDnZPU/L7CHNd********"
```

- 示例一

  您需要发送如下GET请求列出ali-test-project项目下的所有Logstore，其HTTP请求如下：

  ```
  GET /logstores?logstoreName=&offset=0&size=1000 HTTP 1.1
  Mon, 09 Nov 2015 06:11:16 GMT
  Host: ali-test-project.regionid.example.com
  x-log-apiversion: 0.6.0
  x-log-signaturemethod: hmac-sha1
  ```

  如上Log Service API请求生成的签名字符串为：

  ```
  GET\n\n\nMon, 09 Nov 2015 06:11:16 GMT\nx-log-apiversion:0.6.0\nx-log-signaturemethod:hmac-sha1\n/logstores?logstoreName=&offset=0&size=1000
  ```

  由于是GET请求，该请求无任何HTTP Body，所以生成的签名字符串中CONTENT-TYPE与CONTENT-MD5域为空字符串。如果以前面指定的AccessKeySecret做签名运算后得到的签名为：

```
jEYOTCJs2e88o+y5F4/S5IsnBJQ=
```

最后发送经数字签名的HTTP请求内容如下：

```
GET /logstores?logstoreName=&offset=0&size=1000 HTTP 1.1
Mon, 09 Nov 2015 06:11:16 GMT
Host: ali-test-project.regionid.example.com
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
Authorization: LOG bq2sjzesjmo86kq35behupbq:jEYOTCJs2e88o+y5F4/S5IsnBJQ=
```

- 示例二

  您需要给同上例ali-test-project项目中名为test-logstore的Logstore写入下面的日志：

  ```
  topic=""
  time=1447048976
  source="10.10.10.1"
  "TestKey": "TestContent"
  ```

  为此，按照Log Service API定义需要构建如下HTTP请求：

  ```
  POST /logstores/test-logstore HTTP/1.1
  Date: Mon, 09 Nov 2015 06:03:03 GMT
  Host: test-project.regionid.example.com
  x-log-apiversion: 0.6.0
  x-log-signaturemethod: hmac-sha1
  Content-MD5: 1DD45FA4A70A9300CC9FE7305AF2C494
  Content-Length: 52
  x-log-apiversion:0.6.0
  x-log-bodyrawsize:50
  x-log-compresstype:lz4
  x-log-signaturemethod:hmac-sha1
  <日志内容序列化成ProtoBuffer格式的字节流>
  ```

  在这个HTTP请求中，写入的日志内容首先被序列化成ProtoBuffer格式（请参见数据编码方式了解该格式的更多细节）后作为请求Body。所以该请求的Content-Type头的值指定为application/x-protobuf。类似，Content-MD5头的值是请求body对应的MD5值。按照上面的签名字符串构造方式，这个请求对应的签名字符串为：

  ```
  POST\n1DD45FA4A70A9300CC9FE7305AF2C494\napplication/x-protobuf\nMon, 09 Nov 2015 06:03:03 GMT\nx-log-apiversion:0.6.0\nx-log-bodyrawsize:50\nx-log-compresstype:lz4\nx-log-signaturemethod:hmac-sha1\n/logstores/test-logstore
  ```

  同样，以前面示例中的AccessKeySecret做签名运算，得到的最终签名为：

  ```
  XWLGYHGg2F2hcfxWxMLiNkGki6g=
  ```

  最后发送经数字签名的HTTP请求内容如下：

```
POST /logstores/test-logstore HTTP/1.1
Date: Mon, 09 Nov 2015 06:03:03 GMT
Host: test-project.regionid.example.com
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
Content-MD5: 1DD45FA4A70A9300CC9FE7305AF2C494
Content-Length: 52
x-log-apiversion:0.6.0
x-log-bodyrawsize:50
x-log-compresstype:lz4
x-log-signaturemethod:hmac-sha1
Authorization: LOG bq2sjzesjmo86kq35behupbq:XWLGYHGg2F2hcfxWxMLiNkGki6g=
<日志内容序列化成ProtoBuffer格式的字节流>
```

7.通用错误码

本文介绍调用API接口发生错误时的通用错误码。

当API请求发生错误的时候，服务端会返回错误信息，包括HTTP的Status Code和响应Body中的具体错误细节。其中响应Body中的错误细节为如下格式：

```
{
"errorCode" : <ErrorCode>,
"errorMessage" : <ErrorMessage>
}
```

在服务端返回的错误信息中，适用于大部分API接口，但存在部分错误信息为某些API所独有的情况。下表描述API错误响应信息中的通用错误码，它们会在多个API错误响应信息中出现。每个API所独有的错误码会在对应API文件文档中单独描述。

通用错误码

HTTP状态码（Status Code）	错误码（Error Code）	错误消息（Error Message）	描述（Description）
400	MissingContentType	Content-Type does not exist in http header when body is not empty.	请求Body不为空时，缺少Content-Type。请您检查请求头，确认存在Content-Type。
400	MissingBodyRawSize	x-log-bodyrawsize does not exist in header when it is necessary.	压缩场景下没有提供必须的x-log-bodyrawsize请求头。请您检查请求头，确认存在x-log-bodyrawsize。
400	InvalidBodyRawSize	x-log-bodyrawsize is invalid.	x-log-bodyrawsize的值无效。请您检查请求头，确认其取值为有效数字。
400	InvalidCompressType	x-log-compresstype *type* is unsupported.	不支持x-log-compresstype指定的压缩方式。请您检查请求头，确认其值为lz4或Deflate。
400	MissingHost	Host does not exist in http header.	请求头缺少Host。请您检查请求头，确认存在Host。
400	MissingDate	Date does not exist in http header.	请求头缺少Date。请您检查请求头，确认存在Date。
400	InvalidDateFormat	Date *date* must follow RFC822.	请求头中Date的值不符合RFC822标准。请您检查请求头，确认Date取值符合RFC822标准。
400	MissingAPIVersion	x-log-apiversion does not exist in http header.	请求头缺少x-log-apiversion。请您检查请求头，确认存在x-log-apiversion。
400	InvalidAPIVersion	x-log-apiversion *version* is unsupported.	HTTP请求头x-log-apiversion的值不支持。请您检查请求头，确认支持API版本。
400	MissAccessKeyId	x-log-accesskeyid does not exist in header.	没有在Authorization头部提供AccessKey ID。请检查请求头，确认Authorization头部存在AccessKey ID。

HTTP状态码（Status Code）	错误码（Error Code）	错误消息（Error Message）	描述（Description）
400	MissingSignatureMethod	x-log-signaturemethod does not exist in http header.	没有提供HTTP请求头x-log-signaturemethod。请您检查请求头，确认存在x-log-signaturemethod。
400	InvalidSignatureMethod	signature method *method* is unsupported.	x-log-signaturemethod头部指定的签名方法不支持。请您检查请求头，确认支持的签名方法。
400	RequestTimeTooSkewed	Request time exceeds server time more than 15 minutes.	请求的发送时间超过当前服务处理时间前后15分钟的范围。
401	SignatureNotMatch	Signature *signature* is not matched.	请求的数字签名不匹配。请您重试或更换Accesskey后重试。
401	Unauthorized	The AccessKeyId is unauthorized.	提供的AccessKey ID值未授权。请确认您的AccessKey ID有访问日志服务权限。
		The security token you provided is invalid.	STS Token不合法。请检查您的STS接口请求，确认STS Token是合法有效的。
		The security token you provided has expired.	STS Token已经过期。请重新申请STS Token后发起请求。
		AccessKeyId not found: *AccessKey ID*	AccessKey ID不存在。请检查您的AccessKey ID，重新获取后再发起请求。
		AccessKeyId is disabled: *AccessKey ID*	AccessKey ID是禁用状态。请检查您的AccessKey ID，确认为已启动状态后重新发起请求。
		Your SLS service has been forbidden.	日志服务已经被禁用。请检查您的日志服务状态，例如是否已欠费。
401	InvalidAccessKeyId	The access key id you provided is invalid: *AccessKey ID*.	AccessKey ID不合法。请检查您的AccessKey ID，确认AccessKey ID是合法有效的。
		Your SLS service has not opened.	日志服务没有开通。请登录日志服务控制台或者通过API开通日志服务后，重新发起请求。
403	WriteQuotaExceed	Write quota is exceeded.	超过写入日志限额。请您优化写入日志请求，减少写入日志数量。
403	ReadQuotaExceed	Read quota is exceeded.	超过读取日志限额。请您优化读取日志请求，减少读取日志数量。
404	ProjectNotExists	Project *name* does not exist.	日志项目（Project）不存在。请您检查Project名称，确认已存在该Project。
411	MissingContentLength	Content-Length does not exist in http header when it is necessary.	请求头中缺少Content-Length。请您检查请求头，确认存在Content-Length。
415	InvalidContentType	Content-Type *type* is unsupported.	不支持该类型的Content-Type。请您检查Content-Type定义是否正确。

HTTP状态码（Status Code）	错误码（Error Code）	错误消息（Error Message）	描述（Description）
500	InternalServerError	Internal server error message.	服务器内部错误。请您稍后重试。
503	ServerBusy	The server is busy, please try again later.	服务器正忙，请稍后再试。

错误消息中斜体部分为出错相关的具体信息。例如，ProjectNotExists的错误消息中*name*，表示错误消息中该部分会被具体的Project名称来替换。

8.服务操作接口

8.1. OpenSlsService

调用OpenSlsService接口开通日志服务。只有开通服务后，才能进行日志服务的操作。

请求语法

```
POST https://sls.aliyuncs.com/
?AccessKeyId=yourAccessKeyId
&Action=OpenSlsService
&Format=Format
&SignatureMethod=HMAC-SHA1
&SignatureNonce=SignatureNonce
&SignatureVersion=SignatureVersion
&Timestamp=Timestamp
&Version=Version
&Signature=Signature
```

请求参数

- 请求头

  OpenSlsService接口无特有请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
Action	String	是	OpenSlsService	API名称，该接口中固定配置为OpenSlsService。
AccessKeyId	String	是	test-key	阿里云账号的AccessKey ID。
Signature	String	是	xxxxxx	您的签名，详情请参见签名机制。
SignatureMethod	String	是	HMAC-SHA1	签名方式。
SignatureVersion	String	是	1.0	签名算法版本。
SignatureNonce	String	是	dt712rl9d	签名唯一的随机数。用于防止网络重放攻击，建议您每一次请求都使用不同的随机数。
Timestamp	String	是	2018-01-01T12:00:00Z	请求的时间戳。根据ISO8601标准表示时间（UTC格式），格式为yyyy-MM-ddTHH:mm:ssZ。例如2018-01-01T12:00:00Z表示北京时间2018年01月01日20点00分00秒。
Version	String	是	2019-10-23	API版本号，格式为YYYY-MM-DD。
Format	String	否	JSON	返回参数的语言类型。可选值：JSON、XML。默认值：XML。

签名机制

1. 构造规范化请求字符串。
   i. 将参数按照字母序进行排序，不包含Signature参数。
   ii. 使用UTF-8字符集按照RFC3986规则编码参数。
      - A~Z、a~z、0~9、短划线（-）、下划线（_）、英文句点（.）、和波浪线（~）不编码。
      - 其它字符编码为 `%XY` 格式，其中 `XY` 是字符对应ASCII码的十六进制数。例如半角双引号（"）对应的编码为 `%22` 。
      - 使用等号（=）连接编码后的请求参数和参数取值。
      - 使用and（&）连接编码后的请求参数，该参数排序按照字母序进行排序，不包含Signature参数。
   iii. 获取规范化的请求字符串（CanonicalizedQueryString）。
2. 构造签名字符串。
   i. 构造待签名的字符串StringToSign。

```
StringToSign=
 HTTPMethod + "&" +              //HTTPMethod表示发送请求的HTTP方法，例如GET。
 percentEncode("/") + "&" +      //percentEncode("/")表示正斜线（/）经过UTF-8编码获得的值，即%2F。
 percentEncode(CanonicalizedQueryString) //您在步骤1中获取的规范化请求字符串。
```

   ii. 按照RFC2104定义，计算待签名字符串StringToSign的HMAC-SHA1值。

```
Signature = Base64( HMAC-SHA1( AccessKeySecret + "&", UTF-8-Encoding-Of(StringToSign) ) )
```

   iii. 将通过RFC3986规则编码后的参数Signature添加到规范化请求字符串URL中。

Python代码示例如下所示：

```
import base64
import datetime
import hmac
import random
from hashlib import sha1
from urllib import parse
import pytz
url = 'https://sls.aliyuncs.com/?'
accessKeyId = 'test-key'
accessKeySecret = 'test-secret'
params = {
  'AccessKeyId': accessKeyId,
  'Action': 'OpenSlsService',
  'Format': 'JSON',
  'Version': '2019-10-23',
  'SignatureMethod': 'HMAC-SHA1',
  'SignatureNonce': str(int(random.random() * 1000000)),
  'SignatureVersion': '1.0',
  'Timestamp': parse.quote(
    datetime.datetime.now(pytz.utc).strftime('%Y-%m-%dT%H:%M:%SZ')
  )
}
def create_url():
  kvs = sorted(params.items(), key=lambda x: x[0])
  paramStr = '&'.join([k + '=' + v for k, v in kvs])
  sigStr = 'GET&%2F&' + parse.quote(paramStr, safe='')
  hashed = hmac.new((accessKeySecret + '&').encode(), sigStr.encode(), sha1)
  signature = base64.encodebytes(
    hashed.digest()).decode('utf-8').rstrip('\n')
  return url + paramStr + '&Signature=' + signature
print(create_url())
```

返回数据

- 响应头

  OpenSlsService接口无特有响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功。

参数名称	数据类型	示例值	描述
RequestId	String	1CCC2B8E-4FF3-4755-A96C-8CE2E4BF27DF	当次请求的Request ID。
Success	Boolean	true	是否请求成功。 ○ true：请求成功。 ○ false：请求失败。
Code	String	200	返回状态码。
Message	String	您已开通，请前往控制台使用。	返回响应描述。

示例

- 请求示例

```
https://sls.aliyuncs.com/
?AccessKeyId=test-key
&Action=OpenSlsService
&Format=JSON
&SignatureMethod=HMAC-SHA1
&SignatureNonce=222856
&SignatureVersion=1.0
&Timestamp=2020-09-15T13%3A01%3A26Z
&Version=2019-10-23
&Signature=xxxxxxx
```

- 正常返回示例

```
{
 "RequestId": "1CCC2B8E-4FF3-4755-A96C-8CE2E4BF27DF",
 "Message": "您已开通，请前往控制台使用。",
 "Code": "200",
 "Success": true
}
```

错误码

HTTP状态码	错误码	错误信息	描述
400	PermissionDenied	No permission to open SLS service.	没有开通日志服务的权限。
500	GetSpecificationsFailed	Failed to get specifications of commodity.	查询商品规格失败。
500	CreateOrderFailed	Failed to create an order.	订单创建失败。

更多错误码，请参见通用错误码。

8.2. GetSlsService

调用GetSlsService接口获取日志服务的开通状态。

请求语法

```
GET https://sls.aliyuncs.com/
?AccessKeyId=yourAccessKeyId
&Action=GetSlsService
&Format=Format
&SignatureMethod=HMAC-SHA1
&SignatureNonce=SignatureNonce
&SignatureVersion=SignatureVersion
&Timestamp=Timestamp
&Version=Version
&Signature=Signature
```

请求参数

- 请求头

  GetSlsService接口无特有请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
Action	String	是	GetSlsService	API名称。该接口中固定配置为GetSlsService。
AccessKeyId	String	是	test-key	阿里云账号的AccessKey ID。
Signature	String	是	xxxxxx	您的签名，详情请参见签名机制。
SignatureMethod	String	是	HMAC-SHA1	签名方式。
SignatureVersion	String	是	1.0	签名算法版本。
SignatureNonce	String	是	dt712rl9d	签名唯一的随机数。用于防止网络重放攻击，建议您每一次请求都使用不同的随机数。
Timestamp	String	是	2018-01-01T12:00:00Z	请求的时间戳。根据ISO8601标准表示时间（UTC格式），格式为yyyy-MM-ddTHH:mm:ssZ。例如2018-01-01T12:00:00Z表示北京时间2018年01月01日20点00分00秒。
Version	String	是	2019-10-23	API版本号，格式为YYYY-MM-DD。
Format	String	否	JSON	返回参数的语言类型。可选值：JSON、XML。默认值：XML。

返回数据

- 响应头

  GetSlsService接口无特有响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功。

参数名称	数据类型	示例值	描述
RequestId	String	5ACE2FA0-1C36-412A-BA9F-3B0E17A146EA	当次请求的Request ID。
Success	Boolean	true	是否请求成功。 ○ true：请求成功。 ○ false：请求失败。
Code	String	200	返回状态码。
Message	String	successful	返回响应描述。
Enabled	Boolean	true	是否开通服务。 ○ true：已开通服务。 ○ false：未开通服务。

示例

- 请求示例

```
https://sls.aliyuncs.com/
?AccessKeyId=test-key
&Action=GetSlsService
&Format=JSON
&SignatureMethod=HMAC-SHA1
&SignatureNonce=222856
&SignatureVersion=1.0
&Timestamp=2020-09-15T13%3A01%3A26Z
&Version=2019-10-23
&Signature=xxxxxxx
```

- 正常返回示例

```
{
 "RequestId": "5ACE2FA0-1C36-412A-BA9F-3B0E17A146EA",
 "Message": "successful",
 "Enabled": true,
 "Code": "200",
 "Success": true
}
```

错误码

HTTP状态码	错误码	错误信息	描述
400	PermissionDenied	No permission to get SLS service status.	没有查询日志服务开通状态的权限。
500	DescribeUserBusinessStatusFailed	Failed to describe user business status.	查询用户业务状态失败。

更多错误码，请参见通用错误码。

9.日志项目接口

9.1. CreateProject

调用CreateProject接口创建一个Project。

请求语法

```
POST / HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignature
x-log-bodyrawsize: 0
User-Agent: UserAgent
x-log-apiversion: 0.6.0
Host: ProjectName.Endpoint
x-log-signaturemethod: hmac-sha1
Date: GMT Date
Content-Type: application/json
Content-MD5: Content-MD5
Content-Length: ContentLength
Connection: Keep-Alive
{
 "projectName": ProjectName,
 "description": Description
}
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  CreateProject接口无特有请求头，关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	ali-project-test	Project名称。其命名规则如下： ○ Project名称必须全局唯一。 ○ 只能包括小写字母、数字和短划线（-）。 ○ 必须以小写字母或者数字开头和结尾。 ○ 长度为3-63字节。
description	String	是	Description of my-project-test	Project描述。

返回数据

- 响应头

  CreateProject接口无特有响应头，关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

HTTP状态码返回200，表示请求成功。

示例

- 请求示例

```
POST / HTTP/1.1
Header :
{
Content-Type: application/json
x-log-bodyrawsize: 54
Content-Length: 54
Content-MD5: 6EF60FBBCE512F05F71CC2C4B1AADFCF
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
Host: ali-project-test.cn-shanghai.log.aliyuncs.com
Date: Sun, 27 May 2018 08:25:04 GMT
Authorization: LOG yourAccessKeyId:yourSignature
x-log-date: Sun, 27 May 2018 08:25:04 GMT
}
Body :
{
 "projectName": "my-project-test",
 "description": "Description of my-project-test"
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header :
{
Server: nginx
Content-Type: application/json
Content-Length: 0
Connection: close
Access-Control-Allow-Origin: *
Date: Sun, 27 May 2018 08:25:04 GMT
x-log-requestid: 5B0A6B60BB6EE39764D458B5
}
```

错误码

HTTP状态码	错误码	错误信息	描述
400	ProjectAlreadyExist	Project *ProjectName* already exist.	项目已存在。
400	ParameterInvalid	The body is not valid json string.	无效的参数。
500	InternalServerError	Specified Server Error Message.	内部服务调用错误。

更多错误码，请参见通用错误码。

9.2. DeleteProject

调用DeleteProject接口删除一个指定的Project。

请求语法

```
DELETE / HTTP/1.1
Content-Length: 0
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
Host: ProjectName.Endpoint
Date: GMT Date
Authorization: LOG yourAccessKeyId:yourSignature
x-log-date: GMT Date
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project名称。

请求参数

- 请求头

  DeleteProject接口无特有请求头，关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	ali-project-test	Project名称，作为Host的一部分。

返回数据

- 响应头

  DeleteProject接口无特有响应头，关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  HTTP状态码返回200，表示请求成功。该接口调用成功后无任何响应元素。

示例

- 请求示例

```
DELETE / HTTP/1.1
Header :
{
Content-Length: 0
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
Host: ali-project-test.cn-shanghai.log.aliyuncs.com
Date: Sun, 27 May 2018 08:25:04 GMT
Authorization: LOG yourAccessKeyId:/yourSignature
x-log-date: Sun, 27 May 2018 08:25:04 GMT
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header :
{
Server: nginx
Content-Type: application/json
Content-Length: 0
Connection: close
Access-Control-Allow-Origin: *
Date: Sun, 27 May 2018 08:25:04 GMT
x-log-requestid: 5B0A6B60BB6EE39764D458B5
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	The Project does not exist : *projectName*.	项目不存在。
500	InternalServerError	Specified Server Error Message.	内部服务调用错误。

更多错误码，请参见通用错误码。

9.3. UpdateProject

调用UpdateProject接口更新一个Project信息。

请求语法

```
PUT / HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignature
x-log-bodyrawsize: 0
User-Agent:  UserAgent
x-log-apiversion: 0.6.0
Host:  ProjectName.Endpoint
x-log-signaturemethod: hmac-sha1
Date:  GMT Date
Content-Type: application/json
Content-MD5:  Content-MD5
Content-Length:  54
Connection: Keep-Alive
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  UpdateProject接口无特有请求头，关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	ali-project-test	Project的名称，作为Header中Host的一部分。

参数名称	数据类型	是否必填	示例值	描述
description	String	否	Description of my-project-test	Project的描述。默认为空字符串。

返回数据

- 响应头

  UpdateProject接口无特有响应头，关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，表示请求成功。该接口调用成功后无任何响应元素。

示例

- 请求示例

```
PUT / HTTP/1.1
Header :
{
Authorization: LOG yourAccessKeyId:yourSignature
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
Host: ali-project-test.cn-shanghai.log.aliyuncs.com
x-log-signaturemethod: hmac-sha1
Date: Sun, 27 May 2018 07:43:26 GMT
Content-Type: application/json
Content-MD5: A7967D81EFF5E3CD447FB6D8DF294E20
Content-Length: 40
Connection: Keep-Alive
}
Body :
{
  "description": "Description of my-project-test"
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header :
{
Server: nginx
Content-Length: 0
Connection: close
Access-Control-Allow-Origin: *
Date: Sun, 27 May 2018 07:43:27 GMT
x-log-requestid: 5B0A619F205DC3F30EDA9322
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	The Project does not exist : *projectName*.	Project不存在。
400	ParameterInvalid	The body is not valid json string.	无效的参数。

HTTP状态码	错误码	错误信息	描述
500	InternalServerError	Specified Server Error Message.	内部服务调用错误。

更多错误码，请参见通用错误码。

9.4. GetProject

调用GetProject接口查询目标Project的详细信息。

请求语法

```
GET / HTTP/1.1
Content-Length: 0
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
Host: ProjectName.Endpoint
Date: GMT Date
Authorization: LOG yourAccessKeyId:yourSignature
x-log-date: GMT Date
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  GetProject接口无特有请求头，关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	ali-project-test	Project名称。

返回参数

- 响应头

  GetProject接口无特有响应头，关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  HTTP状态码返回200，表示GetProject请求成功。其响应Body中包含Project的详细信息，各参数说明如下所示：

参数名称	数据类型	示例值	描述
createTime	String	2020-11-18 16:55:57	创建Project的时间。
description	String	test	Project描述信息。
lastModifyTime	String	2020-11-18 17:07:26	最后一次更新Project的时间。
owner	String	174****745	创建人的阿里云账号ID。
projectName	String	ali-project-test	Project名称。
status	String	Normal	Project状态。

参数名称	数据类型	示例值	描述
region	String	cn-hangzhou	Project所属地域。

示例

- 请求示例

```
GET / HTTP/1.1
Header :
{
Content-Length: 0
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
Host: ali-project-test.cn-shanghai.log.aliyuncs.com
Date: Sun, 27 May 2018 08:25:04 GMT
Authorization: LOG yourAccessKeyId:yourSignature
x-log-date: Sun, 27 May 2018 08:25:04 GMT
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header :
{
Server: nginx
Content-Type: application/json
Content-Length: 0
Connection: close
Access-Control-Allow-Origin: *
Date: Sun, 27 May 2018 08:25:04 GMT
x-log-requestid: 5B0A6B60BB6EE39764D458B5
}
Body :
{
  "createTime": "2020-11-18 16:55:57",
  "description": "test",
  "lastModifyTime": "2020-11-18 17:07:26",
  "owner": "174****745",
  "projectName": "ali-project-test",
  "region": "cn-hangzhou",
  "status": "Normal"
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	The Project does not exist : *projectName*	Project不存在。
500	InternalServerError	Specified Server Error Message	内部服务调用错误。

更多错误码，请参见通用错误码。

9.5. ListProject

调用ListProject接口列出符合条件的Project信息。

请求语法

```
GET /?offset=offset&size=size HTTP/1.1
Content-Length: 0
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
Host: Endpoint
Date: GMT Date
Authorization: LOG yourAccessKeyId:yourSignature
x-log-date: GMT Date
```

请求参数

- 请求头

  ListProject接口无特有请求头，关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
offset	Integer	否	0	查询开始行，默认值为0。
size	Integer	否	10	分页查询时，设置的每页行数。默认值为100，最大值为500。

返回数据

- 响应头

  ListProject接口无特有响应头，关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，表示请求成功。其响应Body中包含Project列表信息，如下所示：

参数名称	数据类型	示例值	描述
count	Integer	2	当前页返回的Project个数。
total	Integer	11	符合查询条件的Project总数。
projects	Array	不涉及	符合查询条件Project列表。

  projects中的每个参数说明如下所示：

参数名称	数据类型	示例值	描述
createTime	String	1524222931	创建Project的时间。Unix时间戳格式，表示从1970-1-1 00:00:00 UTC计算起的秒数。
description	String	test	Project描述。
lastModifyTime	String	1524539357	最后一次更新Project时间。Unix时间戳格式，表示从1970-1-1 00:00:00 UTC计算起的秒数。

参数名称	数据类型	示例值	描述
owner	String	12****34	创建人的阿里云账户ID。
projectName	String	project1	Project名称。
status	String	Normal	Project状态。 ◦ Normal：正常 ◦ Disable：禁用
region	String	cn-shanghai	Project所属地域。

示例

- 请求示例

```
GET /?offset=0&size=2 HTTP/1.1
Header :
{
Content-Length: 0
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
Host: cn-shanghai.log.aliyuncs.com
Date: Sun, 27 May 2018 09:03:33 GMT
Authorization: LOG yourAccessKeyId:yourSignature
x-log-date: Sun, 27 May 2018 09:03:33 GMT
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header :
{
  Server: nginx
  Content-Type: application/json
  Content-Length: 345
  Connection: close
  Access-Control-Allow-Origin: *
  Date: Sun, 27 May 2018 09:03:33 GMT
  x-log-requestid: 5B0A7465AAEA20CA70DE3064
}
Body :
{
 "count": 2,
 "total": 11,
 "projects": [
  {
   "projectName": "project1",
   "status": "Normal",
   "owner": "12****34",
   "description": "test",
   "region": "cn-shanghai",
   "createTime": "1524222931",
   "lastModifyTime": "1524539357"
  },
  {
   "projectName": "project123456",
   "status": "Normal",
   "owner": "12****34",
   "description": "test",
   "region": "cn-shanghai",
   "createTime": "1471963876",
   "lastModifyTime": "1524539357"
  }
 ]
}
```

错误码

HTTP状态码	错误码	错误信息	描述
400	ParameterInvalid	offset：*offset* pair is invalid	offset参数取值不合法。
500	InternalServerError	Specified Server Error Message.	内部服务调用错误。

更多错误码，请参见通用错误码。

9.6. GetProjectLogs

调用GetProjectLogs接口查询目标Project下的日志，该接口是Project级别的SQL查询接口。

接口说明

- 该接口的query是一个标准的SQL查询语句。
- 查询的Project在请求的域名中指定。
- 查询的Logstore在查询语句的from条件中指定。可以将Logstore看做是SQL中的表。

- 在查询的SQL条件中必须指定要查询的时间范围，时间范围由__date__（Timestamp类型）或__time__（Integer类型，单位是秒）来指定。

请求语法

```
GET /logs query=SELECT avg(latency) as avg_latency FROM  where __date__ >'2017-09-01 00:00:00' and __date__ < '2017-09-02 00:00:00'
Authorization: LOG yourAccessKeyId:yourSignature
Date: Wed, 3 Sept. 2014 08:33:46 GMT
Host: Projectname.endpoint
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  GetProjectLogs接口无特有请求头。关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	big-game	Project名称。
query	String	是	* | SELECT * FROM *logStoreName* where __line__ = 'error' and __date__ >'2014-09-01 00:00:00' and __date__ < '2014-09-01 22:00:00	SQL语句，表示查询时间区间为2014-09-01 00:00:00到2014-09-01 22:00:00，关键字为error的日志数据。

返回数据

- 响应头

  关于Log Service API的公共响应头，请参见公共响应头。该接口特有的响应头如下表所示。

参数名称	数据类型	示例值	描述
x-log-progress	String	Complete	查询结果的状态，包括： ○ Complete：查询已经完成，返回结果为完整结果。 ○ Incomplete：查询已经完成，返回结果为不完整结果，需要重复请求以获得完整结果。
x-log-count	Integer	10000	当前查询结果的日志总数。
x-log-processed-rows	Integer	10000	本次查询处理的行数。
x-log-elapsed-millisecond	Integer	5	本次查询消耗的毫秒时间。

- 响应元素

GetProjectLogs的响应body是一个数组，数组中每个元素是一条日志结果。

参数名称	数据类型	示例值	描述
__time__	Integer	1409529660	日志的时间戳，精度为秒，从1970-1-1 00:00:00 UTC开始计算的秒数。
__source__	String	192.168.1.100	日志的来源，写入日志时指定。
content	Array	"Key3": "error", "Key4": "Value4"	日志原始内容。

示例

以杭州地域内名为big-game的Project为例，查询该Project内名为app_log的Logstore中，主题为groupA的日志数据。查询区间为2014-09-01 00:00:00到2014-09-01 22:00:00，查询关键字为error的日志数据。

- 请求示例

```
GET /logs/?query=SELECT * FROM logStoreName where __line__ = 'error' and __date__ >'2014-09-01 00:00:00' and __date__ < '2014-09-01 22:00:00'&line=20&offset=0 HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignature
Date: Wed, 3 Sept. 2014 08:33:46 GMT
Host: big-game.cn-hangzhou.log.aliyuncs.com
x-log-bodyrawsize: 0
x-log-apiversion: 0.4.0
x-log-signaturemethod: hmac-sha1
```

- 响应示例

```
HTTP/1.1 200 OK
Content-MD5: 36F9F7F0339BEAF571581AF1B0AAAFB5
Content-Type: application/json
Content-Length: 269
Date: Wed, 3 Sept. 2014 08:33:47 GMT
x-log-requestid: efag01234-12341-15432f
x-log-progress : Complete
x-log-count : 10000
x-log-processed-rows: 10000
x-log-elapsed-millisecond:5
{
  "progress": "Complete",
  "count": 2,
  "logs": [
    {
      "__time__": 1409529660,
      "__source__": "192.168.1.100",
      "Key1": "error",
      "Key2": "Value2"
    },
    {
      "__time__": 1409529680,
      "__source__": "192.168.1.100",
      "Key3": "error",
      "Key4": "Value4"
    }
  ]
}
```

响应示例中，x-log-progress的值为Complete，表明整个日志查询已经完成，返回结果为完整结果。本次请求共查询到2条符合条件的日志数据。如果响应结果中的x-log-progress的值为Incomplete，则需要重复请求以获得完整结果。

错误码

HTTP状态码	错误码	错误消息	描述
400	ParameterInvalid	Parameter is invalid.	请求的参数错误。

更多错误码，请参见通用错误码。

10.日志库相关接口

10.1. CreateLogstore

调用CreateLogstore接口创建Logstore。

请求语法

```
POST /logstores HTTP/1.1
x-log-bodyrawsize: 0
Content-Type: application/json
Content-Length: 140
Content-MD5: F3EFCA28442BEEC487451FAD30D78650
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
Host: ProjectName.Endpoint
Date: GMT Date
Authorization: LOG yourAccessKeyId:yourSignature,
x-log-date: Fri, 27 Nov 2020 08:25:10 GMT
{
  "logstoreName" : logStoreName,
  "ttl": ttl,
  "shardCount": shardCount,
  "enable_tracking": enable_tracking,
  "autoSplit": autoSplit,
  "maxSplitShard": maxSplitShard,
  "appendMeta": appendMeta
}
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  CreateLogstore接口无特有请求头，关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	ali-test-project	Project名称。
logstoreName	String	是	test-logstore	Logstore名称。其命名规则如下： ○ 同一个Project下，Logstore名称不可重复。 ○ 只能包括小写字母、数字、短划线（-）和下划线（_）。 ○ 必须以小写字母或者数字开头和结尾。 ○ 长度为3-63字符。
ttl	Integer	是	30	数据的保存时间，单位为天。取值范围为1~3650。如果配置为3650，表示永久保存。
shardCount	Integer	是	2	Shard个数。取值范围为1~10。

参数名称	数据类型	是否必填	示例值	描述
enable_tracking	Boolean	否	false	是否开启WebTracking功能。默认值为false。 ○ true：开启WebTracking。 ○ false：不开启WebTracking。
autoSplit	Boolean	否	true	是否自动分裂Shard功能。默认值为false，表示不开启。 ○ true：自动分裂Shard。 ○ false：不自动分裂Shard。
maxSplitShard	Integer	否	6	自动分裂Shard时的最大分裂数。取值范围为1~64。当autoSplit参数为true时必须设置。
appendMeta	Boolean	否	false	是否开启记录外网IP地址功能。默认值为false。 ○ true：开启记录外网IP地址。 ○ false：不开启记录外网IP地址。

返回数据

- 响应头

  CreateLogstore接口无特有响应头，关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，表示创建Logstore成功。

示例

- 请求示例

```
POST /logstores HTTP/1.1
Header :
{
  'x-log-bodyrawsize': '0',
  'Content-Type': 'application/json',
  'Content-Length': '143',
  'Content-MD5': '2ABE4729F2FE8031328CFA4B9D8A34FB',
  'x-log-apiversion': '0.6.0',
  'x-log-signaturemethod': 'hmac-sha1',
  'Host': 'ali-test-project.cn-hangzhou-devcommon-intranet.sls.aliyuncs.com,
  'Date' : 'Wed, 11 Nov 2015 07:35:00 GMT',
  'Authorization': 'LOG yourAccessKeyId:yourSignature,
  'x-log-date': 'Wed, 11 Nov 2015 07:35:00 GMT'
}
Body :
{
  "logstoreName": "test-logstore",
  "ttl": 30,
  "shardCount": 2,
  "enable_tracking": false,
  "autoSplit": true,
  "maxSplitShard": 6,
  "appendMeta": false
}
```

- 返回示例

```
HTTP/1.1 200 OK
Header:
{
  'Server': 'Tengine',
  'Content-Length': '0',
  'Connection': 'close',
  'Access-Control-Allow-Origin': '*',
  'Date': 'Wed, 11 Nov 2015 07:35:00 GMT',
  'x-log-requestid': '5FC0B8115F2EBE6550063F90'
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	Project *ProjectName* does not exist.	Project不存在。
400	LogstoreAlreadyExist	logstore *logstoreName* already exists.	Logstore已存在。
500	InternalServerError	Specified Server Error Message.	内部服务调用错误。
400	LogStoreInfoInvalid	store name *logstoreName* is invalid	Logstore信息无效。
400	ProjectQuotaExceed	Project Quota Exceed.	超过项目限额。

更多错误码，请参见通用错误码。

10.2. DeleteLogstore

调用DeleteLogstore接口删除指定Logstore，包括所有Shard数据和索引。

请求语法

```
DELETE /logstores/logstoreName HTTP/1.1
Content-Length: 0
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
Host: ProjectName.Endpoint
Date: GMT Date
Authorization : LOG yourAccessKeyId:yourSignature
x-log-date: GMT Date
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  DeleteLogstore接口无特有请求头，关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	my-project	Project名称。
logstoreName	String	是	test_logstore	Logstore名称。

返回数据

- 响应头

  DeleteLogstore接口无特有响应头，关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，表示请求成功。

示例

- 请求示例

```
DELETE /logstores/test_logstore HTTP/1.1
Header:
{
  'Content-Length': '0',
  'x-log-bodyrawsize': '0',
  'x-log-apiversion': '0.6.0',
  'x-log-signaturemethod': 'hmac-sha1',
  'Host': 'my-project.cn-hangzhou-devcommon-intranet.sls.aliyuncs.com',
  'Date': 'Wed, 11 Nov 2015 08:09:38 GMT',
  'Authorization': 'LOG yourAccessKeyId:yourSignature',
  'x-log-date': 'Wed, 11 Nov 2015 08:09:38 GMT'
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header:
{
  'Server': 'Tengine',
  'Content-Length': '0',
  'Connection': 'close',
  'Access-Control-Allow-Origin': '*',
  'Date': 'Wed, 11 Nov 2015 07:35:00 GMT',
  'x-log-requestid': '5FC0B8115F2EBE6550063F90'
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	Project *ProjectName* does not exist.	Project不存在。
404	LogStoreNotExist	logstore *logstoreName* does not exist	Logstore不存在。
405	InvalidMethod	invalid request method: /logstores/*logstoreName*	*logstoreName*参数不合法。

HTTP状态码	错误码	错误信息	描述
500	InternalServerError	Specified Server Error Message	内部服务调用错误。

更多错误码，请参见通用错误码。

10.3. UpdateLogstore

调用UpdateLogstore接口更新Logstore的属性信息。

接口说明

目前该接口只支持更新数据保存时间（TTL）属性。

请求语法

```
PUT /logstores/logstoreName HTTP/1.1
Content-Length: 0
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
Host: ProjectName.Endpoint
Date: GMT Date
Authorization : LOG yourAccessKeyId:yourSignature
x-log-date: Wed, 11 Nov 2015 08:28:19 GMT
{
  "logstoreName": logstoreName,
  "ttl": ttl,
  "enable_tracking": false,
  "shardCount": shardCount,
  "autoSplit": true,
  "maxSplitShard": 64,
  "appendMeta": false
}
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  UpdateLogstore接口无特有请求头。关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	ali-test-project	Project名称。
logstoreName	String	是	test-logstore	Logstore名称。
ttl	Integer	是	1	数据的保存时间，单位为天。取值范围为1~3650。如果配置为3650，表示永久保存。
enable_tracking	Boolean	否	false	是否开启WebTracking。 ○ true：开启WebTracking。 ○ false：不开启WebTracking。

参数名称	数据类型	是否必填	示例值	描述
shardCount	Integer	是	2	Shard分区个数。 **说明** ○ 该接口不支持更新分区个数。 ○ 只能通过SplitShard或MergeShards接口修改分区个数。
autoSplit	Boolean	否	true	是否自动分裂Shard。 ○ true：自动分裂Shard。 ○ false：不自动分裂Shard。
maxSplitShard	Integer	否	64	自动分裂时最大的Shard个数，最小值是1，最大值是64。 **说明** 当autoSplit为true时必须指定。
appendMeta	Boolean	否	false	是否开启记录外网IP地址。 ○ true：开启记录外网IP地址。 ○ false：不开启记录外网IP地址。

返回数据

- 响应头

  UpdateLogstore接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功。

示例

- 请求示例

```
PUT /logstores/test-logstore HTTP/1.1
Header:
{
'x-log-bodyrawsize': '0',
'Content-Type': 'application/json',
'Content-Length': '143',
'Content-MD5': '118D09033B9A4DCBA93A8A496EA095CF',
'x-log-apiversion': '0.6.0',
'x-log-signaturemethod': 'hmac-sha1',
'Host': 'ali-test-project.cn-hangzhou-devcommon-intranet.sls.aliyuncs.com',
'Date': 'Wed, 11 Nov 2015 08:28:19 GMT',
'Authorization': 'LOG yourAccessKeyId:yourSignature',
'x-log-date': 'Wed, 11 Nov 2015 08:28:19 GMT'
}
Body :
{
  "logstoreName": "test-logstore",
  "ttl": 1,
  "enable_tracking": false,
  "shardCount": 2,
  "autoSplit": true,
  "maxSplitShard": 64
  "appendMeta": false
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header:
{
'Server': 'Tengine',
'Content-Length': '0',
'Connection': 'close',
'Access-Control-Allow-Origin': '*',
'Date': 'Wed, 11 Nov 2015 08:28:20 GMT',
'x-log-requestid': '5FC4490F5D13436DA713CBEE'
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	Project *ProjectName* does not exist.	Project不存在。
404	LogStoreNotExist	logstore *logstoreName* does not exist.	LogStore不存在。
400	LogStoreAlreadyExist	logstore *logstoreName* already exists.	LogStore已存在。
500	InternalServerError	Specified Server Error Message.	内部服务调用错误。
400	ParameterInvalid	invalid shard count,you can only modify shardCount by split& merge shard.	无效Shard个数，您只能通过SplitShard或MergeShards接口修改分区个数（shardCount）属性。

更多错误码，请参见通用错误码。

10.4. GetLogstore

调用GetLogstore接口查看Logstore的详细信息。

请求语法

```
GET /logstores/logstoreName HTTP/1.1
Content-Length: 0
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
Host: Projectname.Endpoint
Date: GMT Date
Authorization: LOG yourAccessKeyId:yourSignature
x-log-date: Wed, 11 Nov 2015 07:53:29 GMT
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  GetLogstore接口无特有请求头。关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	ali-test-project	Project名称。
logstoreName	String	是	test-logstore	Logstore名称。

返回数据

- 响应头

  GetLogstore接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，表示请求成功，其响应Body包括具体Logstore属性信息。具体如下：

参数名称	数据类型	示例值	描述
logstoreName	String	test-logstore	Logstore名称。
ttl	Integer	1	数据的保存时间，单位为天。
shardCount	Integer	2	Shard个数。
enable_tracking	Boolean	false	是否开启WebTracking。WebTracking即网络跟踪功能，支持采集HTML、H5、iOS和Android平台的日志。 ○ true：开启WebTracking。 ○ false：不开启WebTracking。
autoSplit	Boolean	true	是否自动分裂Shard。 ○ true：自动分裂Shard。 ○ false：不自动分裂Shard。

参数名称	数据类型	示例值	描述
maxSplitShard	Integer	64	自动分裂时最大的Shard个数，最小值是1，最大值是64。
appendMeta	Boolean	false	是否开启记录外网IP地址。 ◦ true：开启记录外网IP地址。 ◦ false：不开启记录外围IP地址。
createTime	Integer	1447833064	日志库创建时间。Unix时间戳格式，表示从1970-1-1 00:00:00 UTC计算起的秒数。
lastModifyTime	Integer	1447833064	日志库最后一次更新时间。Unix时间戳格式，表示从1970-1-1 00:00:00 UTC计算起的秒数。

示例

- 请求示例

```
GET /logstores/test-logstore HTTP/1.1
Header :
{
'Content-Length': '0',
'x-log-bodyrawsize': '0',
'x-log-apiversion': '0.6.0',
'x-log-signaturemethod': 'hmac-sha1',
'Host': 'ali-test-project.cn-hangzhou-devcommon-intranet.sls.aliyuncs.com',
'Date': 'Wed, 11 Nov 2015 07:53:29 GMT',
'Authorization': 'LOG yourAccessKeyId:yourSignature',
'x-log-date': 'Wed, 11 Nov 2015 07:53:29 GMT'
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header :
{
'Server': 'Tengine',
'Content-Type': 'application/json',
'Content-Length': '274',
'Connection': 'close',
'Access-Control-Allow-Origin': '*',
'Date': 'Mon, 30 Nov 2020 02:27:58 GMT',
'x-log-requestid': '5FC458AE54F7D19F960DB515'
}
Body :
{
  "logstoreName": test-logstore,
  "ttl": 1,
  "shardCount": 2,
  "enable_tracking": False,
  "autoSplit": True,
  "maxSplitShard": 64,
  "createTime": 1447833064,
  "lastModifyTime": 1447833064,
  "appendMeta': False,
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	Project *ProjectName* does not exist.	Project不存在。
404	LogstoreNotExist	logstore *logstoreName* does not exist.	Logstore不存在。
500	InternalServerError	Specified Server Error Message.	内部服务调用错误。

更多错误码，请参见通用错误码。

10.5. ListLogstore

调用ListLogstore接口查询指定Project下的Logstore。支持查询指定Project下所有Logstore列表，也支持查询指定Project下具体Logstore。

请求语法

```
GET /logstores HTTP/1.1
Content-Length: 0
x-log-bodyrawsize': 0
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
Host: ProjectName.Endpoint
Date: GMT Date
Authorization: LOG yourAccessKeyId:yourSignature
x-log-date: Mon, 30 Nov 2020 06:22:17 GMT
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  ListLogstore接口无特有请求头。关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
ProjectName	String	是	test-project	Project名称。
logstoreName	String	否	test-logstore	Logstore名称。支持模糊匹配，例如输入test，则返回名称包含test的Logsotre列表。
offset	Integer	否	1	查询开始行。默认值为0。
size	Integer	否	10	分页查询时，设置的每页行数。最大值为500。

返回数据

- 响应头

ListLogstore接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，表示请求成功。响应Body中包括Logstore信息，如下所示：

参数名称	数据类型	示例值	描述
count	Integer	1	当前页返回的Logstore数目。
total	Integer	1	符合查询条件的Logstore总数。
logstores	Array	["test-logstore"]	返回的Logstore名称列表。

示例

- 请求示例

```
GET /logstores HTTP/1.1
Header:
{
'Content-Length': '0',
'x-log-bodyrawsize': '0',
'x-log-apiversion': '0.6.0',
'x-log-signaturemethod': 'hmac-sha1',
'Host': 'ali-test-project.cn-hangzhou-devcommon-intranet.sls.aliyuncs.com',
'Date': 'Mon, 30 Nov 2020 06:22:17 GMT',
'Authorization': 'LOG yourAccessKeyId:yourSignature',
'x-log-date': 'Mon, 30 Nov 2020 06:22:17 GMT'
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header:
{
'Server': 'Tengine',
'Content-Type': 'application/json',
'Content-Length': '85',
'Connection': 'close',
'Access-Control-Allow-Origin': '*',
'Date': 'Mon, 30 Nov 2020 06:23:03 GMT',
'x-log-requestid': '5FC48FC72068AF63D11472FC'
}
Body:
{
'count':1,
'total':1,
'logstores':["test-logstore"]
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	Project *ProjectName* does not exist.	Project不存在。

HTTP状态码	错误码	错误信息	描述
500	InternalServerError	Specified Server Error Message.	内部服务调用错误。
400	ParameterInvalid	Invalid parameter size, (0.6.0].	无效参数。
400	InvalidLogStoreQuery	logstore Query is invalid.	无效查询。

更多错误码，请参见通用错误码。

10.6. ListShards

调用ListShards接口列出指定Logstore中所有可用的Shard。

请求语法

```
GET /logstores/logstorename/shards HTTP/1.1
Content-Length: 0
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
x-log-signaturemethod': hmac-sha1
Host: ProjectName.Endpoint
Date: GMT Date
Authorization: LOG yourAccessKeyId:yourSignature
x-log-date: Mon, 30 Nov 2020 06:22:17 GMT
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  ListShards接口无特有请求头。关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	ali-test-project	Project名称。
logstoreName	String	是	sls-test-logstore	Logstore名称。

返回数据

- 响应头

  ListShards接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，表示请求成功。请求成功后，其响应Body中包含Shard信息，如下所示：

参数名称	数据类型	示例值	描述
shardID	Integer	0	Shard ID。
status	String	readwrite	Shard状态。

参数名称	数据类型	示例值	描述
inclusiveBeginKey	String	00000000000000000000000000000000	Shard起始的Key值，在Shard MD5范围中包含该值。
exclusiveEndKey	String	80000000000000000000000000000000	Shard结束的Key值，在Shard MD5范围中不包含该值。
createTime	String	1453949705	Shard的创建时间。Unix时间戳格式，表示从1970-1-1 00:00:00 UTC计算起的秒数。

示例

- 请求示例

```
GET /logstores/sls-test-logstore/shards
Header :
{
'Content-Length': '0'
'x-log-bodyrawsize': '0'
'x-log-apiversion': '0.6.0'
'x-log-signaturemethod': 'hmac-sha1'
'Host': ali-test-project.cn-hangzhou-devcommon-intranet.sls.aliyuncs.com
'Date': Mon, 30 Nov 2020 06:22:17 GMT
'Authorization': 'LOG yourAccessKeyId:yourSignature'
'x-log-date': 'Mon, 30 Nov 2020 06:22:17 GMT'
}
```

- 返回示例

```
Header:
{
'Server': 'Tengine',
'Content-Type': 'application/json',
'Content-Length': '335',
'Connection': 'close',
'Access-Control-Allow-Origin': '*',
'Date': 'Mon, 30 Nov 2020 08:32:24 GMT',
'x-log-requestid': '5FC4AE1821CA7D41C5F14728'
}
Body :
[
  {
    "shardID": 1,
    "status": "readwrite",
    "inclusiveBeginKey": "00000000000000000000000000000000",
    "exclusiveEndKey": "80000000000000000000000000000000",
    "createTime": 1453949705
  },
  {
    "shardID": 2,
    "status": "readwrite",
    "inclusiveBeginKey": "80000000000000000000000000000000",
    "exclusiveEndKey": "ffffffffffffffffffffffffffffffff",
    "createTime": 1453949705
  },
  {
    "shardID": 0,
    "status": "readonly",
    "inclusiveBeginKey": "00000000000000000000000000000000",
    "exclusiveEndKey": "ffffffffffffffffffffffffffffffff",
    "createTime": 1453949705
  }
]
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	The Project does not exist : *projectName*	Project不存在。
404	LogStoreNotExist	logstore *logstoreName* does not exist.	Logstore不存在。
500	InternalServerError	Specified Server Error Message.	内部服务调用错误。
400	LogStoreWithoutShard	logstore has no shard.	Logstore中没有Shard。

更多错误码，请参见通用错误码。

10.7. SplitShard

调用SplitShard接口分裂一个指定的readwrite状态的Shard。

请求语法

```
POST /logstores/logstoreName/shards/shardID HTTP/1.1
Content-Length: 0
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
Host: ProjectName.Endpoint
Date': GMT Date
Authorization': LOG yourAccessKeyId:yourSignature
x-log-date: Tue, 01 Dec 2020 01:36:00 GMT
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  SplitShard接口无特有请求头。关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	ali-test-project	Project名称。
logstoreName	String	是	logstorename	Logstore名称。
shardID	Integer	是	33	Shard ID。
splitKey	String	是	ef00000000000000000000000000000000	分裂位置。

返回数据

- 响应头

  SplitShard接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，表示请求成功。请求成功后，其响应Body中包含3个Shard元素组成的数组，第一个Shard为分裂之前的Shard，后两个为分裂的结果。各参数如下所示：

参数名称	数据类型	示例值	描述
shardID	Integer	33	Shard ID。
status	String	readonly	分区状态包括： ○ readonly：只读数据 ○ readwrite：读写数据
inclusiveBeginKey	String	ee00000000000000000000000000000000	分区起始的Key值。
exclusiveEndKey	String	ffffffffffffffffffffffffffffffff	分区结束的Key值。
createTime	String	1453949705	分区的创建时间。Unix时间戳格式，表示从1970-1-1 00:00:00 UTC计算起的秒数。

示例

- 请求示例

```
POST /logstores/logstorename/shards/33 HTTP/1.1
Header :
{
'Content-Length': '0',
'x-log-bodyrawsize': '0',
'x-log-apiversion': '0.6.0',
'x-log-signaturemethod': 'hmac-sha1',
'Host': 'ali-test-project.cn-hangzhou.sls.aliyuncs.com',
'Date': 'Tue, 01 Dec 2020 01:36:00 GMT',
'Authorization': 'LOG yourAccessKeyId:yourSignature',
'x-log-date': 'Tue, 01 Dec 2020 01:36:00 GMT'
}
```

- 正常返回示例

```
Header:
{
  "content-length": "57",
  "server": "nginx/1.6.1",
  "connection": "close",
  "date": "Thu, 12 Nov 2015 03:40:31 GMT",
  "content-type": "application/json",
  "x-log-requestid": "56440A2F99248C050600C74C"
}
Body :
[
  {
    "shardID": 33,
    "status": "readonly",
    "inclusiveBeginKey": "ee000000000000000000000000000000",
    "exclusiveEndKey": "ffffffffffffffffffffffffffffffff",
    "createTime": 1453949705
  },
  {
    "shardID": 163,
    "status": "readwrite",
    "inclusiveBeginKey": "ee000000000000000000000000000000",
    "exclusiveEndKey": "ef000000000000000000000000000000",
    "createTime": 1453949705
  },
  {
    "shardID": 164,
    "status": "readwrite",
    "inclusiveBeginKey": "ef000000000000000000000000000000",
    "exclusiveEndKey": "ffffffffffffffffffffffffffffffff",
    "createTime": 1453949705
  }
]
```

错误码

HTTP状态码	错误码	错误信息	描述
404	LogStoreNotExist	logstore *logstoreName* does not exist.	Logstore不存在。

HTTP状态码	错误码	错误信息	描述
400	ParameterInvalid	invalid shard id.	无效Shard ID。
400	ParameterInvalid	invalid mid hash.	无效分裂位置参数。
500	InternalServerError	Specified Server Error Message.	内部服务调用错误。
400	LogStoreWithoutShard	logstore has no shard.	Logstore没有Shard。

更多错误码，请参见通用错误码。

10.8. MergeShards

调用MergeShards接口合并两个相邻的readwrite状态的Shards。在参数中指定一个shardID，服务端自动找相邻的下一个Shard进行合并。

请求语法

```
POST /logstores/logstoreName/shards/shardID HTTP/1.1
Content-Length: 0
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
Host: ProjectName.Endpoint
Date: GMT Date
Authorization: LOG yourAccessKeyId:yourSignature
x-log-date: Mon, 30 Nov 2020 09:37:18 GMT
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  MergeShards接口无特有请求头。关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	ali-test-project	Project名称。
logstoreName	String	是	logstorename	Logstore名称。
shardID	Integer	是	30	Shard ID。

返回数据

- 响应头

  MergeShards接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，表示请求成功。请求成功后，其响应Body中包含3个Shard元素组成的数组，第一个Shard为合并之后的Shard，后两个为合并之前的Shard。各参数如下所示：

参数名称	数据类型	示例值	描述
shardID	Integer	167	Shard ID。
status	String	readwrite	分区状态包括： ○ readonly：只读数据 ○ readwrite：读写数据
inclusiveBeginKey	String	ee000000000000000000000000000000	分区起始的Key值。
exclusiveEndKey	String	ffffffffffffffffffffffffffffffff	分区结束的Key值。
createTime	String	1453953105	分区的创建时间。Unix时间戳格式，表示从1970-1-1 00:00:00 UTC计算起的秒数。

示例

- 请求示例

```
POST /logstores/logstorename/shards/30
Header:
{
  "Content-Length": 0,
  "x-log-signaturemethod": "hmac-sha1",
  "x-log-bodyrawsize": 0,
  "User-Agent": "log-python-sdk-v-0.6.0",
  "Host": "ali-test-project.cn-hangzhou.sls.aliyuncs.com",
  "Date": "Thu, 12 Nov 2015 03:40:31 GMT",
  "x-log-apiversion": "0.6.0",
  "Authorization": "LOG yourAccessKeyId:yourSignature"
}
```

- 正常返回示例

```
Header:
{
  "content-length": "57",
  "server": "nginx/1.6.1",
  "connection": "close",
  "date": "Thu, 12 Nov 2015 03:40:31 GMT",
  "content-type": "application/json",
  "x-log-requestid": "56440A2F99248C050600C74C"
}
Body :
[
  {
    'shardID': 167,
    'status': 'readwrite',
    'inclusiveBeginKey': 'e00000000000000000000000000000000',
    'createTime': 1453953105,
    'exclusiveEndKey': 'ffffffffffffffffffffffffffffffff'
  },
  {
    'shardID': 30,
    'status': 'readonly',
    'inclusiveBeginKey': 'e00000000000000000000000000000000',
    'createTime': 0,
    'exclusiveEndKey':
    'e70000000000000000000000000000000'
  },
  {
    'shardID': 166,
    'status': 'readonly',
    'inclusiveBeginKey': 'e70000000000000000000000000000000',
    'createTime': 1453953073,
    'exclusiveEndKey': 'ffffffffffffffffffffffffffffffff'
  }
]
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	The Project does not exist : *projectName*	Project不存在。
404	LogStoreNotExist	logstore *logstoreName* does not exist.	Logstore不存在。
400	ParameterInvalid	invalid shard id.	无效Shard ID。
400	ParameterInvalid	can not merge the last shard.	无效merge。
500	InternalServerError	Specified Server Error Message.	内部服务调用错误。
400	LogStoreWithoutShard	logstore has no shard.	Logstore没有Shard。

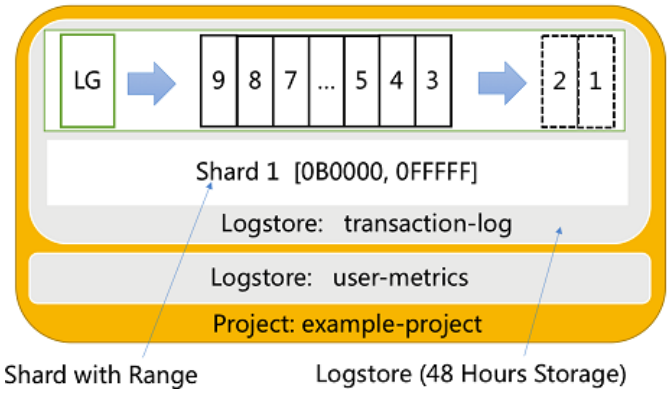
更多错误码，请参见通用错误码。

10.9. GetCursor

调用GetCursor接口可以根据时间获取对应的游标（Cursor）。

接口说明

通过游标（Cursor）可以获取特定日志对应的位置。关于Cursor与Project、Logstore、Shard的关系如下图所示。



- Project下有多个Logstore。
- 每个Logstore会有多个Shard。
- 通过Cursor可以获得特定日志对应的位置。

请求语法

```
GET /logstores/logstoreName/shards/shardID HTTP/1.1
Content-Type: application/json
Content-Length: 0
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
Host: ProjectName.Endpoint
Date: GMT Date
Authorization: LOG yourAccessKeyId:yourSignature
x-log-date: Tue, 01 Dec 2020 05:51:27 GMT
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  GetCursor接口无特有请求头。关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	ali-test-project	Project名称。
logstoreName	String	是	sls-test-logstore	Logstore名称。
shardID	String	是	0	Shard ID。
type	String	是	cursor	请求类型。固定取值为cursor。
from	String	是	begin	时间点（Unix时间戳）或者字符串`begin`、`end`。

通过from可以在Shard中定位生命周期内的日志，假设Logstore的生命周期为[begin_time,end_time)，from=from_time，那么：

- 当 `from_time ≤ begin_time or from_time = "begin"` 时：返回时间点为begin_time对应的Cursor位置。
- 当 `from_time ≥ end_time or from_time = "end"` 时：返回当前时间点下一条将被写入的Cursor位置（当前该Cursor位置上无数据）。
- 当 `from_time > begin_time and from_time < end_time` 时：返回第一个服务端接收时间大于等于from_time的数据包对应的Cursor。

> **说明** Logstore生命周期由属性中TTL字段指定。例如，当前时间为2018-11-11 09:00:00，TTL=5。则每个Shard中可以消费的数据时间段为 `[2018-11-05 09:00:00,2018-11-11 09:00:00)`，这里的时间指的是服务端时间。

返回数据

- 响应头

  GetCursor接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功，其响应Body会包括对应游标信息，具体如下：

参数名称	数据类型	示例值	描述
cursor	String	MTQ0NzI5OTYwNjg5NjYzMjM1Ng==	Cursor值。

示例

- 请求示例

```
GET /logstores/sls-test-logstore/shards/0?type=cursor&from=begin
Header:
{
  "Content-Length": 0,
  "x-log-signaturemethod": "hmac-sha1",
  "x-log-bodyrawsize": 0,
  "User-Agent": "log-python-sdk-v-0.6.0",
  "Host": "ali-test-project.cn-hangzhou-devcommon-intranet.sls.aliyuncs.com",
  "Date": "Thu, 12 Nov 2015 03:56:57 GMT",
  "x-log-apiversion": "0.6.0",
  "Content-Type": "application/json",
  "Authorization": "LOG yourAccessKeyId:yourSignature"
}
```

- 正常返回示例

```
Header:
{
  "content-length": "41",
  "server": "nginx/1.6.1",
  "connection": "close",
  "date": "Thu, 12 Nov 2015 03:56:57 GMT",
  "content-type": "application/json",
  "x-log-requestid": "56440E0999248C070600C6AA"
}
Body:
{
  "cursor": "MTQ0NzI5OTYwNjg5NjYzMjM1Ng=="
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	LogStoreNotExist	Logstore *logstoreName* does not exist.	Logstore不存在。
400	ParameterInvalid	Parameter From is not valid.	无效参数。
400	ShardNotExist	Shard *ShardID* does not exist.	Shard不存在。
500	InternalServerError	Specified Server Error Message.	内部服务调用错误。
400	LogStoreWithoutShard	The logstore has no shard.	Logstore没有Shard。

更多错误码，请参见通用错误码。

10.10. GetCursorTime

调用GetCursorTime接口可以根据Cursor获取服务端时间。

请求语法

```
GET /logstores/logstoreName/shards/shardID?cursor=cursor&type=cursor_time HTTP/1.1
Content-Type: application/json
Content-Length: 0
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
Host: ProjectName.Endpoint
Date: GMT Date
Authorization: LOG yourAccessKeyId:yourSignature
x-log-date: Tue, 01 Dec 2020 05:51:27 GMT
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  GetCursorTime接口无特有请求头。关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	参数类型	是否必填	示例值	参数说明
projectName	String	是	ali-test-project	Project名称。
logstoreName	String	是	internal-operation_log	Logstore名称。
shardID	Integer	是	0	Shard ID。
cursor	String	是	MTU0NzQ3MDY4MjM3NjUxMzQ0Ng%3D%3D	希望获取时间戳的Cursor。

参数名称	参数类型	是否必填	示例值	参数说明
type	String	是	cursor_time	默认为cursor_time。

返回数据

- 响应头

  GetCursorTime接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功，其响应Body会获取服务端时间信息，具体如下：

参数名称	数据类型	示例值	描述
cursor_time	String	1554260243	Cursor的服务端时间。Unix时间戳格式，表示从1970-1-1 00:00:00 UTC计算起的秒数。

示例

- 请求示例

```
GET /logstores/internal-operation_log/shards/0?cursor=MTU0NzQ3MDY4MjM3NjUxMzQ0Ng%3D%3D&type=cursor_time HTTP/1.1
Header :
{
'Content-Type': 'application/json',
'Content-Length': '0',
'x-log-bodyrawsize': '0',
'x-log-apiversion': '0.6.0',
'x-log-signaturemethod': 'hmac-sha1',
'Host': 'ali-test-project.cn-hangzhou.log.aliyuncs.com',
'Date': 'Tue, 01 Dec 2020 06:41:14 GMT',
'Authorization': 'LOG yourAccessKeyId:yourSignature',
'x-log-date': 'Tue, 01 Dec 2020 06:41:14 GMT'
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header :
{
'Server': 'Tengine',
'Content-Type': 'application/json',
'Content-Length': '26',
'Connection': 'close',
'Access-Control-Allow-Origin': '*',
'Date': 'Tue, 01 Dec 2020 06:42:01 GMT',
'x-log-requestid': '5FC5E5B9C920EAE5D5B9D1D4'
}
Body :
{
 "cursor_time": 1554260243
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	LogStoreNotExist	Logstore *logstoreName* does not exist.	Logstore不存在。
400	ShardNotExist	Shard *shardID* does not exist.	Shard不存在。
500	InternalServerError	Specified Server Error Message.	内部服务调用错误。
400	LogStoreWithoutShard	the logstore has no shard.	Logstore没有Shard。

更多错误码，请参见通用错误码。

10.11. PullLogs

调用PullLogs接口获取指定游标（Cursor）位置的日志数据。

接口说明

- 获取日志时必须指定Shard。
- 目前仅支持读取Protocol Buffer格式数据。

请求语法

```
GET /logstores/logstoreName/shards/shardID HTTP/1.1
Accept: application/x-protobuf
Accept-Encoding: lz4
Authorization: LOG yourAccessKeyId:yourSignature
Date: GMT Date
Host: Projectname.Endpoint
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  PullLogs接口的特有请求头如下所示：

  - Accept：application/x-protobuf
  - Accept-Encoding：lz4

  其中，Accept-Encoding取值包括lz4、deflate或双引号（“”）之一。

  关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	ali-test-project	Project名称。
logstoreName	String	是	sls-test-logstore	Logstore名称。
shardID	Integer	是	0	Shard ID。

参数名称	数据类型	是否必填	示例值	描述
type	String	是	logs	请求类型，固定取值为logs。
cursor	String	是	MTQ0NzMyOTQwMTEwMjEzMDkwNA	游标，表示从什么位置开始读取数据，相当于起点。
count	Integer	否	1000	返回的Loggroup数目，最小值为1，最大值为1000。

返回数据

- 响应头

  PullLogs接口的特有响应元素如下所示：

  - x-log-cursor：当前读取数据下一条Cursor。
  - x-log-count：当前返回数量。

  关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  protobuf格式序列化后的数据（可能经过压缩）。

示例

- 请求示例

```
GET /logstores/sls-test-logstore/shards/0?cursor=MTQ0NzMyOTQwMTEwMjEzMDkwNA==&count=1000&type=log
Header:
{
  "Authorization"="LOG yourAccessKeyId:yourSignature",
  "x-log-bodyrawsize"=0,
  "User-Agent" : "sls-java-sdk-v-0.6.0",
  "x-log-apiversion" : "0.6.0",
  "Host" : "ali-test-project.cn-hangzhou-failover-intranet.sls.aliyuncs.com",
  "x-log-signaturemethod" : "hmac-sha1",
  "Accept-Encoding" : "lz4",
  "Content-Length": 0,
  "Date" : "Thu, 12 Nov 2015 12:03:17 GMT",
  "Content-Type" : "application/x-protobuf",
  "accept" : "application/x-protobuf"
}
```

- 正常返回示例

```
Header:
{
  "x-log-count" : "1000",
  "x-log-requestid" : "56447FB20351626D7C000874",
  "Server" : "nginx/1.6.1",
  "x-log-bodyrawsize" : "34121",
  "Connection" : "close",
  "Content-Length" : "4231",
  "x-log-cursor" : "MTQ0NzMyOTQwMTEwMjEzMDkwNA==",
  "Date" : "Thu, 12 Nov 2015 12:01:54 GMT",
  "x-log-compresstype" : "lz4",
  "Content-Type" : "application/x-protobuf"
}
Body:
<protobuf格式loggrouplist内容>压缩后结果
```

- 翻页

  如果仅进行翻页，获取下一组Token，不返回数据，可以通过HTTP HEAD方式进行请求。

错误码

HTTP状态码	错误码	错误信息	描述
404	LogStoreNotExist	Logstore *logstoreName* does not exist.	Logstore不存在。
400	ParameterInvalid	Invalid cursor *cursor*.	游标参数无效。
400	ParameterInvalid	ParameterCount should be in [0-1000].	count参数取值范围应该为[0-1000]。
400	ShardNotExist	Shard *ShardID* does not exist.	Shard不存在。
500	InternalServerError	Specified Server Error Message.	内部服务调用错误。

更多错误码，请参见通用错误码。

10.12. PutLogs

调用PutLogs接口向指定的Logstore中写入日志数据。

接口说明

- 服务端会对每次PutLogs写入的日志数据做格式检查，只要日志数据中有任何一条日志不符合规范，则整个请求失败且无任何日志数据成功写入。
- 目前仅支持写入PB格式的日志数据，日志数据以LogGroup的形式展示。
- 日志数据写入时有两种模式：
  - 负载均衡模式（LoadBalance）：自动根据Logstore下所有可写的Shard进行负载均衡写入。该方法写入可用性较高（SLA：99.95%），适合写入消费数据与Shard无关的场景，例如不保序。
  - Key路由Shard模式（KeyHash）：在URL参数中增加Key字段，用来判断数据写入哪个Shard中。该参数为可选参数，不设置时自动切换为负载均衡写入模式。例如，可以将某个生产者（例如instance）根据名称Hash固定到Shard上，这样就能保证写入与消费在该Shard上的数据是严格有序的（在合并、分裂过程中能够严格保证对于Key在一个时间点只会出现在一个Shard上，请参见分区（Shard））。
- PutLogs接口每次可以写入的日志组数据量上限为3 MB或者4096条，日志组中每条日志下的Value部分不超过1 MB大小。只要日志数据量超过这两条上限中的任意一条则整个请求失败，且无任何日志数据成功写入。

PB数据

PB格式日志压缩数据字段描述如下。详情请参见数据模型和数据编码方式。

- Log

参数名称	数据类型	是否必填	描述
Time	Integer	是	日志时间。Unix时间戳格式，表示从1970-1-1 00:00:00 UTC计算起的秒数。
Contents	List	是	日志字段列表，至少包含一个元素，每个元素类型请参见Content表格。

- Content

参数名称	数据类型	是否必填	描述
Key	String	是	自定义Key名称。
Value	String	是	自定义Key对应的值。

- LogTag

参数名称	数据类型	是否必填	描述
Key	String	是	自定义Key名称。
Value	String	是	自定义Key对应的值。

- LogGroup

参数名称	数据类型	是否必填	描述
Logs	List	是	日志列表，每个元素请参见Log字段表格。
Topic	String	否	日志主题，用户自定义字段，用于区分不同特征的日志数据。
Source	String	否	日志的来源。例如产生该日志的机器的IP地址。
LogTags	List	是	日志的标签列表，每个元素请参见LogTag。

请求语法

- 负载均衡模式

```
POST /logstores/logstoreName/shards/lb HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignature
Content-Type: application/x-protobuf
Content-Length: Content Length
Content-MD5: Content MD5
Date: GMT Date
Host: ProjectName.Endpoint
x-log-apiversion: 0.6.0
x-log-bodyrawsize: BodyRawSize
x-log-compresstype: lz4
x-log-signaturemethod: hmac-sha1
<PB 格式日志压缩数据>
```

- Key路由Shard模式

```
POST /logstores/logstoreName/shards/route?key=14d2f850ad6ea48e46e4547edbbb27e0
Authorization: LOG yourAccessKeyId:yourSignature
Content-Type: application/x-protobuf
Content-Length: Content Length
Content-MD5: Content MD5
Date: GMT Date
Host: ProjectName.Endpoint
x-log-apiversion: 0.6.0
x-log-bodyrawsize: BodyRawSize
x-log-compresstype: lz4
x-log-signaturemethod: hmac-sha1
<PB 格式日志压缩数据>
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  关于Log Service API的公共请求头，请参见公共请求头。

- 请求参数

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	ali-test-project	Project名称。
logstoreName	String	是	sample-logtail-config	Logstore名称。

返回数据

- 响应头

  PutLogs接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功。该接口调用成功后无任何响应元素。

示例

- 请求示例

```
POST /logstores/sls-test-logstore/shards/lb
{
  "Content-Length": 118,
  "Content-Type":"application/x-protobuf",
  "x-log-bodyrawsize":1356,
  "Host": "ali-test-project.cn-hangzhou-devcommon-intranet.sls.aliyuncs.com",
  "Content-MD5":"6554BD042149C844761C2C094A8FECCE",
  "Date":"Thu, 12 Nov 2015 06:54:26 GMT",
  "x-log-apiversion": "0.6.0",
  "x-log-compresstype":"lz4"
  "x-log-signaturemethod": "hmac-sha1",
  "Authorization":"LOG yourAccessKeyId:yourSignature"
}
<PB格式日志使用Lz4压缩后的二进制数据>
```

- 正常返回示例

```
Header
{
  "Access-control-allow-origin" : "*",
  "Date" : "Wed, 11 Nov 2015 08:28:20 GMT",
  "Server" : "Tengine",
  "Content-Length" : 0,
  "x-log-requestid" : "5642FC2399248C8F7B0145FD",
  "Connection" : "close"
}
```

错误码

HTTP状态码	错误码	错误消息	描述
400	PostBodyInvalid	Protobuffer content cannot be parsed.	Protobuffer内容不合法，无法解析。
400	InvalidTimestamp	Invalid timestamps are in logs.	日志内容中有无效的日志时间戳。
400	InvalidEncoding	Non-UTF8 characters are in logs.	日志内容中有非UTF8字符。
400	InvalidKey	Invalid keys are in logs.	日志内容中有无效的Key。
400	PostBodyTooLarge	Logs must be less than or equal to 3 MB and 4096 entries.	日志内容包含的日志必须小于3 MB和4096条。
400	PostBodyUncompressError	Failed to decompress logs.	日志内容解压失败。
400	PostBodyInvalid	The post data time is out of range	日志中时间范围不在[-7*24小时 + 15分钟]有效范围内。
404	LogStoreNotExist	logstore *logstoreName* does not exist.	Logstore不存在。

更多错误码，请参见通用错误码。

10.13. GetShipperStatus

调用GetShipperStatus接口查询日志投递任务状态。

接口说明

该接口只支持查询最近48小时之内的投递任务状态。

请求语法

```
GET /logstores/logstoreName/shipper/shipperName/tasks HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignature
Date: GMT Date
Host: ProjectName.Endpoint
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  GetShipperStatus接口无特有请求头。关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	ali-test-project	Project名称。
logstoreName	String	是	test-logstore	Logstore名称。
shipperName	String	是	test-shipper	日志投递名称。
startTime	Integer	是	1448748198	日志投递任务创建开始时间。Unix时间戳格式，表示从1970-1-1 00:00:00 UTC计算起的秒数。
endTime	Integer	是	1448948198	日志投递任务创建结束时间。Unix时间戳格式，表示从1970-1-1 00:00:00 UTC计算起的秒数。
statusType	String	否	success	默认为空，表示返回所有状态的任务，支持success、fail和running状态。
offset	Integer	否	0	查询开始行，默认值为0。
size	Integer	否	100	分页查询时，设置的每页行数。默认值为100，最大值为500。

返回数据

- 响应头

  GetShipperStatus接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

返回HTTP状态码200，则表示请求成功，其响应Body会包括指定的日志投递任务列表，具体如下：

参数名称	数据类型	示例值	描述
count	Integer	10	当前页返回的任务个数。
total	Integer	20	任务总数。
statistics	Json	无	任务汇总状态，具体请参见下表。
tasks	Array	无	投递任务详情，具体请参见下表。

statistics中每个元素，具体如下：

参数名称	数据类型	示例值	描述
running	Integer	0	状态为running的任务个数。
success	Integer	20	状态为success的任务个数。
fail	Integer	0	状态为fail的任务个数。

tasks中每个元素，具体如下：

参数名称	数据类型	示例值	描述
id	String	abcdefghijk	投递任务ID。
taskStatus	String	success	投递任务状态，为running、success和fail中的一种。
taskMessage	String	无	投递任务失败时的具体错误信息。
taskCreateTime	Integer	1448925013	投递任务创建时间。Unix时间戳格式，表示从1970-1-1 00:00:00 UTC计算起的秒数。
taskLastDataReceiveTime	Integer	1448915013	投递任务中的最近一条日志到达服务端时间（非日志时间，是服务端接收时间）。Unix时间戳格式，表示从1970-1-1 00:00:00 UTC计算起的秒数。
taskFinishTime	Integer	1448926013	投递任务结束时间。Unix时间戳格式，表示从1970-1-1 00:00:00 UTC计算起的秒数。

示例

- 请求示例

```
GET /logstores/test-logstore/shipper/test-shipper/tasks?from=1448748198&to=1448948198&status=success&offset=0
&size=100 HTTP/1.1
Header:
{
  "x-log-apiversion" : 0.6.0,
  "Authorization" : "LOG yourAccessKeyId:yourSignature",
  "Host" : "ali-test-project.cn-hangzhou-devcommon-intranet.sls.aliyuncs.com",
  "Date" : "Wed, 11 Nov 2015 08:28:19 GMT",
  "Content-Length" : 55,
  "x-log-signaturemethod" : "hmac-sha1",
  "Content-MD5" : "757C60FC41CC7D3F60B88E0D916D051E",
  "User-Agent" : "sls-java-sdk-v-0.6.0",
  "Content-Type" : "application/json"
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header:
{
  "Date" : "Wed, 11 Nov 2015 08:28:20 GMT",
  "Content-Length" : 0,
  "x-log-requestid" : "5642FC2399248C8F7B0145FD",
  "Connection" : "close",
  "Server" : "nginx/1.6.1"
}
Body:
{
  "count" : 10,
  "total" : 20,
  "statistics" : {
    "running" : 0,
    "success" : 20,
    "fail" : 0
  }
  "tasks" : [
    {
      "id" : "abcdefghijk",
      "taskStatus" : "success",
      "taskMessage" : "",
      "taskCreateTime" : 1448925013,
      "taskLastDataReceiveTime" : 1448915013,
      "taskFinishTime" : 1448926013
    }
  ]
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	Project *ProjectName* does not exist.	Project不存在。
404	LogStoreNotExist	Logstore *logstoreName* does not exist.	Logstore不存在。
400	ShipperNotExist	Shipper *shipperName* does not exist.	Shipper不存在。

HTTP状态码	错误码	错误信息	描述
500	InternalServerError	Internal server error.	内部服务调用错误。
400	ParameterInvalid	Start time must be earlier than end time.	开始时间必须早于结束时间。
400	ParameterInvalid	Only support query last 48 hours task status.	只支持查询最近48小时的任务状态。
400	ParameterInvalid	Status only contains success/running/fail.	状态只包含success、running和fail。

更多错误码，请参见通用错误码。

10.14. RetryShipperTask

调用RetryShipperTask接口重新执行失败日志投递任务。

接口说明

每次最多只能重新执行10个失败的投递任务。

请求语法

```
PUT /logstores/logstoreName/shipper/shipperName/tasks HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignature
Date: GMT Date
Host: ProjectName.Endpoint
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
["task-id-1", "task-id-2", "task-id-2"]
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  RetryShipperTask接口无特有请求头。关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	ali-test-project	Project名称。
logstoreName	String	是	test-logstore	Logstore名称。
shipperName	String	是	test-shipper	日志投递名称。

返回数据

- 响应头

  RetryShipperTask接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功。该接口调用成功后无任何响应元素。

示例

- 请求示例

```
PUT /logstores/test-logstore/shipper/test-shipper/tasks HTTP/1.1
Header:
{
  "x-log-apiversion" : 0.6.0,
  "Authorization" : "LOG yourAccessKeyId:yourSignature",
  "Host" : "ali-test-project.cn-hangzhou-devcommon-intranet.sls.aliyuncs.com",
  "Date" : "Wed, 11 Nov 2015 08:28:19 GMT",
  "Content-Length" : 55,
  "x-log-signaturemethod" : "hmac-sha1",
  "Content-MD5" : "757C60FC41CC7D3F60B88E0D916D051E",
  "User-Agent" : "sls-java-sdk-v-0.6.0",
  "Content-Type" : "application/json"
}
Body :
["task-id-1", "task-id-2", "task-id-2"]
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header:
{
  "Date" : "Wed, 11 Nov 2015 08:28:20 GMT",
  "Content-Length" : 0,
  "x-log-requestid" : "5642FC2399248C8F7B0145FD",
  "Connection" : "close",
  "Server" : "nginx/1.6.1"
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	Project *projectName* does not exist.	Project不存在。
404	LogStoreNotExist	Logstore *logstoreName* does not exist.	Logstore不存在。
400	ShipperNotExist	Shipper does not exist.	投递任务不存在。
500	InternalServerError	Specified Server Error Message.	内部服务调用错误。
400	ParameterInvalid	Each time allows 10 task retries only.	每次最多只能重新执行10个任务。

更多错误码，请参见通用错误码。

10.15. GetLogs

调用GetLogs接口查询指定Project下某个Logstore中的日志数据。

接口说明

- 该接口中由请求参数from和to定义的时间区间遵循左闭右开原则，即该时间区间包括区间开始时间点，但不包括区间结束时间点。如果from和to的值相同，则为无效区间，函数直接返回错误。

- 当查询涉及的日志数量变化非常大时，日志服务API无法预测需要调用多少次该接口来获取完整结果。所以需要您查看每次请求返回结果中的x-log-progress状态值，根据状态值来确定是否需要重复调用该接口来获取最终完整结果。每次重复调用该接口都会重新消耗相同数量的查询CU。
- 当日志写入到Logstore中，日志服务的查询接口（GetHistograms和GetLogs）能够查到该日志的延时因写入日志类型不同而异。日志服务按日志时间戳把日志分为如下两类：
  - 实时数据：日志中时间点为服务器当前时间点（-180秒，900秒]。例如，日志时间为UTC 2014-09-25 12:03:00，服务器收到时为UTC 2014-09-25 12:05:00，则该日志被作为实时数据处理，一般出现在正常场景下。
  - 历史数据：日志中时间点为服务器当前时间点[-7x86400秒，-180秒）。例如，日志时间为UTC 2014-09-25 12:00:00，服务器收到时为UTC 2014-09-25 12:05:00，则该日志被作为历史数据处理，一般出现在补数据场景下。

  其中，实时数据写入至可查询的最大延时为3秒，99.9%情况下1秒内即可查询完毕。

请求语法

```
GET /logstores/logstoreName?type=log&topic=topic&from=from&to=to&query=query&line=line&offset=offset&reverse=reverse HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignature
Date: GMT Date
Host: ProjectName.Endpoint
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  GetLogs接口无特有请求头。关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	big-game	Project名称。
logstoreName	String	是	app_log	Logstore名称。
type	String	是	log	查询Logstore数据的类型。在该接口中固定取值为log。
from	Integer	是	1409529600	查询开始时间点。Unix时间戳格式，表示从1970-1-1 00:00:00 UTC计算起的秒数。
to	Integer	是	1409608800	查询结束时间点。Unix时间戳格式，表示从1970-1-1 00:00:00 UTC计算起的秒数。
topic	String	否	groupA	日志主题。

参数名称	数据类型	是否必填	示例值	描述
query	String	否	error	查询和分析语句。更多信息，请参见查询简介和分析简介。 在query参数的分析语句中加上 `set session parallel_sql=true;` ，表示使用SQL独享实例。例如 `* | set session parallel_sql=true; select count(*) as pv` 。 **说明** 当query参数中有分析语句（SQL语句）时，line参数和offset参数需要设置为0，通过LIMIT语法翻页。更多信息，请参见分析结果分页。
line	Integer	否	20	请求返回的最大日志条数。最小值为0，最大值为100，默认值为100。
offset	Integer	否	0	查询开始行。默认值为0。
reverse	Boolean	否	false	是否按日志时间戳逆序返回日志，精确到分钟级别。 ○ true：按照逆序返回日志。 ○ false（默认值）：按照顺序返回日志。
powerSql	Boolean	否	false	是否使用SQL独享实例。更多信息，请参见开启SQL独享实例。 ○ true：使用SQL独享实例。 ○ false（默认值）：使用SQL普通实例。 除通过powerSql参数配置SQL独享实例外，您还可以使用query参数。

返回数据

- 响应头

  GetLogs接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

  响应头中有专门成员表示请求返回结果是否完整。具体响应元素格式如下：

参数名称	数据类型	示例值	描述

参数名称	数据类型	示例值	描述
x-log-progress	String	Complete	查询结果的状态，包括： ○ Complete：查询已经完成，返回结果为完整结果。 ○ Incomplete：查询已经完成，但由于返回结果数据量大且结果时间区间跨度大，导致返回结果为不完整结果。需要重复请求以获得完整结果。
x-log-count	Integer	10000	当前查询扫描的日志总数。

- 响应元素

  返回HTTP状态码200，则表示请求成功，其响应Body会包括查询命中的日志数据。当需要查询的日志数据量非常大（T级别）时，该接口的响应结果可能并不完整，GetLogs的响应body是一个数组，数组中每个元素是一条日志结果。数组中的每个元素结构如下：

参数名称	数据类型	示例值	描述
__time__	Integer	1409529660	日志的时间戳。Unix时间戳格式，表示从1970-1-1 00:00:00 UTC计算起的秒数。
__source__	String	192.168.1.100	日志的来源，写入日志时指定。例如产生该日志的机器的IP地址。
contents	List	["Key1": "error","Key2": "Value2"]	日志原始内容。

示例

以杭州地域内名为big-game的Project为例，查询该Project内名为app_log的Logstore中，主题为groupA的日志数据。查询区间为2014-09-01 00:00:00到2014-09-01 22:00:00，查询关键字为error，且从时间区间头开始查询，最多返回20条日志数据。

- 请求示例：

```
GET /logstores/app_log?type=log&topic=groupA&from=1409529600&to=1409608800&query=error&line=20&offset=0
HTTP/1.1
Header:
{
Authorization: LOG yourAccessKeyId:yourSignature
Date: Wed, 3 Sept. 2014 08:33:46 GMT
Host: big-game.cn-hangzhou.log.aliyuncs.com
x-log-bodyrawsize: 0
x-log-apiversion: 0.4.0
x-log-signaturemethod: hmac-sha1
}
```

- 正常返回示例：

```
HTTP/1.1 200 OK
Header :
{
Content-MD5: 36F9F7F0339BEAF571581AF1B0AAAFB5
Content-Type: application/json
Content-Length: 269
Date: Wed, 3 Sept. 2014 08:33:47 GMT
x-log-requestid: efag01234-12341-15432f
x-log-progress : Complete
x-log-count : 10000
x-log-processed-rows: 10000
x-log-elapsed-millisecond:5
}
Body :
{
  "progress": "Complete",
  "count": 2,
  "logs": [
    {
      "__time__": 1409529660,
      "__source__": "192.168.1.100",
      "contents": ["Key1": "error","Key2": "Value2"]
    },
    {
      "__time__": 1409529680,
      "__source__": "192.168.1.100",
      "contents": ["Key3": "error","Key4": "Value4"]
    }
  ]
}
```

在这个响应示例中，x-log-progress成员的状态为Complete，表明整个日志查询已经完成，返回结果为完整结果。在这次请求中共查询到2条符合条件的日志，且日志数据在logs成员中。如果响应结果中的x-log-progress成员的状态为Incomplete，则表示查询结果不完整，需要重复相同请求以获得完整结果。

错误码

HTTP状态码	错误码	错误消息	描述
404	ProjectNotExist	Project *ProjectName* does not exist.	Project不存在。
404	LogStoreNotExist	logstore *logstoreName* does not exist.	Logstore不存在。
400	InvalidTimeRange	Request time range is invalid.	请求的时间区间无效。
400	InvalidQueryString	Query string is invalid.	请求的查询字符串无效。
400	InvalidOffset	Offset is invalid.	请求的offset参数无效。
400	InvalidLine	Line is invalid.	请求的line参数无效。
400	InvalidReverse	Reverse value is invalid.	Reverse参数的值无效。
400	IndexConfigNotExist	Logstore without index config.	Logstore未开启索引。

HTTP状态码	错误码	错误消息	描述
400	ParameterInvalid	ErrorType:OLSQueryParseError.ErrorMessage:offset is not available for pagination in sql query, please use limit x,y syntax for pagination.	当query参数中有分析语句（SQL语句）时，line参数和offset参数需要设置为0，通过LIMIT语法翻页。

更多错误码，请参见通用错误码。

10.16. GetContextLogs

调用GetContextLogs接口查询指定日志前（上文）后（下文）的若干条日志。

接口说明

上下文查询的时间范围为起始日志的前后一天。

请求语法

```
GET /logstores/logstoreName?type=context_log&pack_id=pack_id&pack_meta=pack_meta&back_lines=back_lines&forward_lines=forward_lines HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignature
Date: GMT Date
Host: ProjectName.Endpoint
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  GetContextLogs接口无特有请求头。关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	big-game	Project名称。
logstoreName	String	是	app_log	Logstore名称。
type	String	是	context_log	Logstore中数据的类型。该接口中该参数固定为context_log。
pack_id	String	是	id	起始日志所属的LogGroup的唯一身份标识。
pack_meta	String	是	meta	起始日志在对应LogGroup内的唯一上下文结构标识。

参数名称	数据类型	是否必填	示例值	描述
back_lines	Integer	是	100	指定起始日志往前（上文）的日志条数，取值范围为(0,100]。
forward_lines	Integer	是	100	指定起始日志往后（下文）的日志条数，取值范围为(0,100]。

返回数据

- 响应头

  GetContextLogs接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功，其响应Body包括查询到的日志（可能为空），具体如下：

参数名称	数据类型	示例值	描述
total_lines	Integer	201	返回的总日志条数，包含请求参数中所指定的起始日志。
back_lines	Integer	100	向前查询到的日志条数。
forward_lines	Integer	100	向后查询到的日志条数。
progress	String	Complete	查询的结果是否完整。 ○ Complete：查询已经完成，返回结果为完整结果。 ○ Incomplete：查询已经完成，返回结果为不完整结果，需要重复请求以获得完整结果。
logs	Array	无	获取到的日志，按上下文顺序排列。当根据指定起始日志查询不到上下文日志时，此参数为空。

logs中的每一项都是该日志的内容（键值对），除用户日志内容外，还包含三个字段，具体如下：

参数名称	数据类型	示例值	描述
__index_number__	String	-100	该日志在本次查询结果中相对上下文的位置，负数表示上文，0表示起始日志，正数表示下文。例如：-100表示起始日志往前的第100条日志。
__tag__:__pack_id__	String	895CEA449A52FE-8c8	该日志所属的LogGroup的唯一身份标识，可作为请求参数中的pack_id进行查询。

参数名称	数据类型	示例值	描述
__pack_meta__	String	0|MTU1OTI4NTExMjg3NTQ2NDU1OA==|4|1	该日志在所属LogGroup内的唯一上下文结构标识，可作为请求参数中的pack_meta进行查询。

示例

- 请求示例

```
GET /logstores/app_log?type=context_log&&pack_id=id&pack_meta=meta&back_lines=100&forward_lines=100 HTTP/1.1
Header:
{
Authorization: LOG yourAccessKeyId:yourSignature
Date: Wed, 3 Sept. 2014 08:33:46 GMT
Host: big-game.cn-hangzhou.log.aliyuncs.com
x-log-bodyrawsize: 0
x-log-apiversion: 0.4.0
x-log-signaturemethod: hmac-sha1
}
```

- 正常返回示例

```
{
  "total_lines": 201,
  "back_lines": 100,
  "forward_lines": 100,
  "progress": "Complete",
  "logs": [
    {
      "__index_number__": "-100",
      "__tag__:__pack_id__": "895CEA449A52FE-8c8",
      "__pack_meta__": "0|MTU1OTI4NTExMjg3NTQ2NDU1OA==|4|1",
      ...
    }
  ]
}
```

- 此处以Java SDK为例，请使用Java SDK 0.6.38及以上的版本。

  如果您通过Logtail采集方式采集日志，该方式将自动为日志附加上下文信息。

```
package sdksample;
import com.aliyun.openservices.log.Client;
import com.aliyun.openservices.log.common.LogContent;
import com.aliyun.openservices.log.common.LogItem;
import com.aliyun.openservices.log.common.QueriedLog;
import com.aliyun.openservices.log.common.TagContent;
import com.aliyun.openservices.log.exception.LogException;
import com.aliyun.openservices.log.request.PutLogsRequest;
import com.aliyun.openservices.log.response.GetContextLogsResponse;
import com.aliyun.openservices.log.response.GetLogsResponse;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
public class GetContextLogsSample {
  private static int getCurrentTimestamp() {
    return (int) (new Date().getTime() / 1000);
  }
```

```
  private static class PackInfo {
    public String packID;
    public String packMeta;
    public PackInfo(String id, String meta) {
      this.packID = id;
      this.packMeta = meta;
    }
  }
  private static PackInfo extractPackInfo(QueriedLog log) {
    PackInfo ret = new PackInfo("", "");
    ArrayList<LogContent> contents = log.GetLogItem().GetLogContents();
    for (int i = 0; i < contents.size(); ++i) {
      LogContent content = contents.get(i);
      if (content.GetKey().equals("__tag__:__pack_id__")) {
        ret.packID = content.GetValue();
      } else if (content.GetKey().equals("__pack_meta__")) {
        ret.packMeta = content.GetValue();
      }
    }
    return ret;
  }
  public static void main(String args[]) throws InterruptedException, LogException {
    String endpoint = "EndPoint";
    String accessKeyId = "yourAccessKeyId"; // 使用您的阿里云访问密钥AccessKey ID。
    String accessKeySecret = "yourSignature"; // 使用您的阿里云访问密钥AccessKey Secret。
    String project = "ProjectName"; // 要查询的Project。
    String logstore = "logstoreName"; // 要查询的Logstore。
    // 构建一个客户端实例。
    Client client = new Client(endpoint, accessKeyId, accessKeySecret);
    System.out.println("请确保指定的logstore已开启索引。");
    Thread.sleep(3000);
    // 使用GetLogs并在查询语句中加上|with_pack_meta来获取起始日志的pack_id和pack_meta。
    // 查询时间范围：最近15分钟。
    // 起始日志：返回结果的第一条。
    String query = "*|with_pack_meta";
    GetLogsResponse response = client.GetLogs(project, logstore,
        (int) getCurrentTimestamp() - 900, (int) getCurrentTimestamp(),"", query);
    ArrayList<QueriedLog> logs = response.GetLogs();
    if (logs.isEmpty()) {
      System.out.println("未查询到任何日志。");
      System.exit(1);
    }
    // 提取第一条日志的pack信息。
    PackInfo info = extractPackInfo(logs.get(0));
    if (info.packMeta.isEmpty() || info.packID.isEmpty()) {
      System.out.println("pack ID: " + info.packID + ", pack meta: " + info.packMeta);
      System.out.println("起始日志的pack信息不完整，请确保该日志是通过logtail写入。");
      System.exit(1);
    }
    // 使用得到的pack信息进行上下文查询（双向查询）。
    GetContextLogsResponse contextRes = client.getContextLogs(project, logstore,
        info.packID, info.packMeta, 10, 10);
    System.out.println("双向查询");
    System.out.println("pack ID: " + info.packID + ", pack meta: " + info.packMeta);
    System.out.println("is complete: " + contextRes.isCompleted());
    System.out.println("total lines: " + contextRes.getTotalLines());
    System.out.println("back lines: " + contextRes.getBackLines());
    System.out.println("forward lines: " + contextRes.getForwardLines());
    Thread.sleep(1000);
    // 使用查询结果中的第一条日志，向前查询上文（单向），至多三次。
```

```
    List<QueriedLog> contextLogs = contextRes.getLogs();
    for (int i = 0; i < 3 && !contextLogs.isEmpty(); i++) {
      QueriedLog log = contextLogs.get(0);
      info = extractPackInfo(log);
      GetContextLogsResponse res = client.getContextLogs(project, logstore,
          info.packID, info.packMeta, 10, 0);
      System.out.println("向前查询上文");
      System.out.println("pack ID: " + info.packID + ", pack meta: " + info.packMeta);
      System.out.println("is complete: " + res.isCompleted());
      System.out.println("total lines: " + res.getTotalLines());
      System.out.println("back lines: " + res.getBackLines());
      System.out.println("forward lines: " + res.getForwardLines());
      contextLogs = res.getLogs();
      Thread.sleep(1000);
    }
    // 使用查询结果中的最后一条日志，向后查询下文（单向），至多三次。
    contextLogs = contextRes.getLogs();
    for (int i = 0; i < 3 && !contextLogs.isEmpty(); i++) {
      QueriedLog log = contextLogs.get(contextLogs.size() - 1);
      info = extractPackInfo(log);
      GetContextLogsResponse res = client.getContextLogs(project, logstore,
          info.packID, info.packMeta, 0, 10);
      System.out.println("向后查询下文");
      System.out.println("pack ID: " + info.packID + ", pack meta: " + info.packMeta);
      System.out.println("is complete: " + res.isCompleted());
      System.out.println("total lines: " + res.getTotalLines());
      System.out.println("back lines: " + res.getBackLines());
      System.out.println("forward lines: " + res.getForwardLines());
      contextLogs = res.getLogs();
      Thread.sleep(1000);
    }
  }
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	Project *ProjectName* does not exist.	Project不存在。
404	LogStoreNotExist	logstore *logstoreName* does not exist.	Logstore不存在。
400	InvalidParameter	Invalid pack meta/id.	请求参数中的pack_meta或pack_id为非法值。
400	InvalidParameter	back_lines or forward_lines must be postive.	请求参数中的back_lines或forward_lines中有非法值，其中至少有一个参数应为正数。

更多错误码，请参见通用错误码。

10.17. GetHistograms

调用GetHistograms接口查询指定Logstore中满足查询语法条件的日志分布情况。

接口说明

- 该接口中query参数仅支持 `查询语句` ，不支持 `分析语句` 。关于查询语句的详细语法，请参见查询语法。
- 该接口中涉及的时间区间（无论是请求参数from和to定义的时间区间，还是返回结果中各个子时间区间）都遵循“左闭右开”原则，即该时间区间包括区间开始时间点，但不包括区间结束时间点。如果from和to的值相同，则为无效区间，函数直接返回错误。
- 该接口的响应中子区间划分方式是一直稳定的。如果您在请求查询的时间区间不变，则响应中子区间划分结果也不会改变。
- 当查询涉及的日志数量变化非常大时，日志服务API无法预测需要调用多少次该接口来获取完整结果。所以需要您查看每次请求返回结果中的progress成员状态值，根据成员状态值来确定是否需要重复调用该接口来获取最终完整结果。每次重复调用该接口都会重新消耗相同数量的查询CU。
- 从日志写入日志库到查询接口（GetHistograms和GetLogs）查到该日志，延时时长因写入日志类型不同而异。日志服务按日志时间戳把日志分为如下两类：
  - 实时数据：日志中时间点为服务器当前时间点（-180秒，900秒]。例如，日志时间为UTC 2014-09-25 12:03:00，服务器收到时为UTC 2014-09-25 12:05:00，则该日志被视作实时数据处理。实时数据从写入到在日志查询界面查询到该数据的延迟为3秒。
  - 历史数据：日志中时间点为服务器当前时间点[-7 x 86400秒，-180秒）。例如，日志时间为UTC 2014-09-25 12:00:00，服务器收到时为UTC 2014-09-25 12:05:00，则该日志被作为历史数据处理，一般出现在补数据场景下。

请求语法

```
GET /logstores/logstoreName?type=histogram&topic=topic&from=from&to=to&query=query HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignature
Date: GMT Date
Host: ProjectName.Endpoint
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  GetHistograms接口无特有请求头。关于Log Service API的公共请求头，请参见 公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	big-game	Project名称。
logstoreName	String	是	app_log	Logstore名称。
type	String	是	histogram	Logstore中数据的类型。该接口中固定取值为histogram。
from	Integer	是	1409529600	查询开始时间点。Unix时间戳格式，表示从1970-1-1 00:00:00 UTC计算起的秒数。
to	Integer	是	1409608800	查询结束时间点。Unix时间戳格式，表示从1970-1-1 00:00:00 UTC计算起的秒数。
topic	String	否	groupA	日志主题。

参数名称	数据类型	是否必填	示例值	描述
query	String	否	error	查询语句。仅支持`查询语句`，不支持`分析语句`。关于查询语句的详细语法，请参见查询语法。

返回数据

- 响应头

  GetHistograms接口无特有响应头。关于Log Service API的公共响应头，请参见 公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功，其响应Body会包括查询命中的日志数量在时间轴上的分布情况。响应结果会把您查询的时间范围均匀分割成若干个子时间区间，并返回每个子时间区间内命中的日志条数。具体如下：

参数名称	数据类型	示例值	描述
progress	String	Complete	查询结果的状态。 ○ Complete：查询已经完成，返回结果为完整结果。 ○ Incomplete：查询已经完成，返回结果为不完整结果，需要重复请求以获得完整结果。
count	Integer	2	当前查询结果中所有日志条数。
histograms	Array	无	当前查询结果在划分的子时间区间上的分布状况，具体结构见下面的描述。

histograms数组中的每个元素结构如下：

参数名称	数据类型	示例值	描述
from	Integer	1409529600	子时间区间的开始时间点。Unix时间戳格式，表示从1970-1-1 00:00:00 UTC计算起的秒数。
to	Integer	1409569200	子时间区间的结束时间点。Unix时间戳格式，表示从1970-1-1 00:00:00 UTC计算起的秒数。
count	Integer	2	该子时间区间内查询到的日志条数。
progress	Sting	Complete	当前查询结果在该子时间区间内的结果是否完整。 ○ Complete：查询已经完成，返回结果为完整结果。 ○ Incomplete：查询已经完成，返回结果为不完整结果，需要重复请求以获得完整结果。

示例

以杭州地域内名为big-game的Project为例，查询该Project内名为app_log的Logstore中，主题为groupA的日志分布情况。查询区间为2014-09-01 00:00:00到2014-09-01 22:00:00，查询关键字是error。

- 请求示例

```
GET /logstores/app_log?type=histogram&topic=groupA&from=1409529600&to=1409608800&query=error HTTP/1.1
Header :
{
Authorization: LOG yourAccessKeyId:yourSignature
Date: Wed, 3 Sept. 2014 08:33:46 GMT
Host: big-game.cn-hangzhou.log.aliyuncs.com
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
}
```

- 响应示例

```
HTTP/1.1 200 OK
Header :
{
Content-Type: application/json
Content-MD5: E6AD9C21204868C2DE84EE3808AAA8C8
Content-Type: application/json
Date: Wed, 3 Sept. 2014 08:33:47 GMT
Content-Length: 232
x-log-requestid: efag01234-12341-15432f
}
Body :
{
    {
      "from": 1409529600,
      "to": 1409569200,
      "count": 2,
      "progress": "Complete"
    },
    {
      "from": 1409569200,
      "to": 1409608800,
      "count": 2,
      "progress": "Complete"
    }
}
```

在下面这个响应示例中，服务端把整个Histogram划分到两个均等的时间区间，分别为[2014-09-01 00:00:00，2014-09-01 11:00:00）和[2014-09-01 11:00:00，2014-09-01 22:00:00）。由于数据量过大，第一次查询会返回不完整数据。响应结果告知您命中日志条数为3，但整体结果不完整，仅在时间段[2014-09-01 00:00:00，2014-09-01 11:00:00）结果完整且命中2条，而在另外一个时间段结果不完整但已经命中1条。在这个情况下，如果您希望得到完整结果，则需要重复调用上面的请求示例若干次直到整个响应中的progress成员值变成Complete。响应示例如下：

```
HTTP/1.1 200 OK
Header :
{
Content-Type: application/json
Content-MD5: E6AD9C21204868C2DE84EE3808AAA8C8
Content-Type: application/json
Date: Wed, 3 Sept. 2014 08:33:48 GMT
Content-Length: 232
x-log-requestid: afag01322-1e241-25432e
}
Body :
{
    {
      "from": 1409529600,
      "to": 1409569200,
      "count": 2,
      "progress": "Complete"
    },
    {
      "from": 1409569200,
      "to": 1409608800,
      "count": 1,
      "progress": "Incomplete"
    }
}
```

错误码

HTTP状态码	错误码	错误消息	描述
404	ProjectNotExist	Project *ProjectName* does not exist.	Project不存在。
404	LogStoreNotExist	logstore *logstoreName* does not exist.	Logstore不存在。
400	InvalidTimeRange	request time range is invalid.	请求的时间区间无效。
400	InvalidQueryString	query string is invalid.	请求的查询字符串无效。

更多错误码，请参见通用错误码。

10.18. UpdateIndex

调用UpdateIndex接口更新指定Logstore的索引。

请求语法

```
PUT /logstores/logstoreName/index HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignature
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
Host: ProjectName.Endpoint
x-log-signaturemethod: hmac-sha1
Date: GMT Date
{
 "line": line,
 "keys": keys
}
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  UpdateIndex接口无特有请求头，关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	my-project	Project名称。
logstoreName	String	是	logstore-4	Logstore名称。
keys	Object	否	无	字段索引配置，key为字段名称，value为字段索引配置。
line	Object	否	无	全文索引配置。

参数keys和line必须至少指定一个。

- 全文索引配置包含如下信息：

参数名称	数据类型	是否必填	示例值	描述
token	Array	是	["\n", "\t","\r"]	分词符列表。可以设置一个分词参数，指定这个字段按照哪一种方式分词。更多分词符可，请参见示例。
caseSensitive	Boolean	否	true	是否大小写敏感。 ◦ true：大小写敏感。 ◦ false：大小写不敏感。
chn	Boolean	否	false	是否包含中文。 ◦ true：包含中文。 ◦ false：不包含中文。

参数名称	数据类型	是否必填	示例值	描述
include_keys	Array	否	无	包含的字段列表，不能与exclude_keys同时指定。
exclude_keys	Array	否	无	排除的字段列表，不能与include_keys同时指定。

- 字段索引配置包含如下信息：

参数名称	数据类型	是否必填	示例值	描述
type	String	是	text	字段类型。
alias	String	否	agent_alias	字段别名。
chn	Boolean	否	false	是否包含中文。仅当type参数取值为text时，必须设置。 ◦ true：包含中文。 ◦ false：不包含中文。
token	Array	否	["\n", "\t","\r"]	分词符列表。仅当type参数取值为text时，必须设置。
caseSensitive	Boolean	否	true	是否大小写敏感。仅当type参数取值为text时，必须设置。 ◦ true：大小写敏感。 ◦ false：大小写不敏感。
doc_value	Boolean	否	true	是否开启统计。 ◦ true：开启统计。 ◦ false：不开启统计。

返回数据

- 响应头

  UpdateIndex接口无特有响应头，关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功。该接口调用成功后无任何响应元素。

示例

- 请求示例

```
PUT /logstores/logstore-4/index HTTP/1.1
```

```
Header :
{
Authorization: LOG yourAccessKeyId:yourSignature
x-log-bodyrawsize: 0
User-Agent: sls-java-sdk-v-0.6.1
x-log-apiversion: 0.6.0
Host: my-project.cn-shanghai.log.aliyuncs.com
x-log-signaturemethod: hmac-sha1
Date: Mon, 07 May 2018 09:47:20 GMT
Content-Type: application/json
Content-MD5: 1860B805A6AA5288B97B32CF3B519112
Content-Length: 316
Connection: Keep-Alive
}
Body :
{
 "line": {
  "token": [
   ",",
   " ",
   "'",
   "\",
   ";",
   "=",
   "(",
   ")",
   "[",
   "]",
   "{",
   "}",
   "?",
   "@",
   "&",
   "<",
   ">",
   "/",
   ":",
   "\n",
   "\t",
   "\r"
  ]
 },
 "keys": {
  "agent": {
   "doc_value": true,
   "caseSensitive": true,
   "alias": "agent_alias",
   "type": "text",
   "token": [
    ",",
    " ",
    "'",
    "\"",
    ";",
    "=",
    "(",
    ")",
    "[",
    "]",
    "{",
    "}"
```

```
      "}",
      "?",
      "@",
      "&",
      "<",
      ">",
      "/",
      ":",
      "\n",
      "\t",
      "\r"
    ]
   }
  }
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Server: nginx/1.12.1
Content-Length: 0
Connection: close
Access-Control-Allow-Origin: *
Date: Mon, 07 May 2018 09:47:21 GMT
x-log-requestid: 5AF020A98CBAA2EF52AB66C9
```

错误码

HTTP状态码	错误码	错误信息	描述
400	ParameterInvalid	log store index is not created.	没有创建日志存储索引。
400	ParameterInvalid	key config must has type.	关键配置必须有类型。
400	IndexInfoInvalid	required field token is lacking or of error format.	缺少必要的字段标记或格式错误。
400	IndexInfoInvalid	required fields line/keys are lacking or of error format.	缺少所需的字段、行键或格式错误。
404	ProjectNotExist	The Project does not exist : *projectName*.	Project不存在。
404	LogStoreNotExist	logstore *logstoreName* dose not exist.	Logstore不存在。
500	InternalServerError	Specified Server Error Message.	内部服务调用错误。

更多错误码，请参见通用错误码。

10.19. CreateIndex

调用CreateIndex接口为指定Logstore创建索引。

请求语法

```
POST /logstores/logstoreName/index HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignature
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
Host: ProjectName.Endpoint
x-log-signaturemethod: hmac-sha1
Date: GMT Date
{
 "line": line,
 "keys": keys
}
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  CreateIndex接口无特有请求头，关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	my-project	Project名称。
logstoreName	String	是	my-logstore	Logstore的名称。
keys	Object	否	无	字段索引配置，key为字段名称，value为字段索引配置。该参数和line必须至少指定一个，更多示例，请参见示例。
line	Object	否	无	全文索引配置。该参数和keys必须至少指定一个，更多示例，请参见示例。

- 全文索引配置包含如下信息：

参数名称	数据类型	是否必填	示例值	描述
caseSensitive	Boolean	否	true	是否大小写敏感。 ■ true：大小写敏感。 ■ false：大小写不敏感。
chn	Boolean	否	true	是否包含中文。 ■ true：包含中文。 ■ false：不包含中文。
token	Array	是	["\n","\t","\r"]	分词符列表。可以设置一个分词参数，指定这个字段按照哪一种方式分词。更多分词符，请参见示例。
include_keys	Array	否	无	包含的字段列表，不能与exclude_keys同时指定。
exclude_keys	Array	否	无	排除的字段列表，不能与include_keys同时指定。

- 字段索引配置包含如下信息：

参数名称	数据类型	是否必填	示例值	描述
type	String	是	index	字段类型。
alias	String	否	agent_alias	字段别名。
chn	Boolean	否	true	是否包含中文。仅当type参数取值为text时，必须设置。 ■ true：包含中文。 ■ false：不包含中文。
token	Array	否	["\n","\t","\r"]	分词符列表。仅当type参数取值为text时，必须设置。
caseSensitive	Boolean	否	true	是否大小写敏感。仅当type参数取值为text时，必须设置。 ■ true：大小写敏感。 ■ false：大小写不敏感。
doc_value	Boolean	否	true	是否开启统计。 ■ true：开启统计。 ■ false：不开启统计。

返回数据

- 响应头

  CreateIndex接口无特有响应头，关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功。该接口调用成功后无任何响应元素。

示例

- 请求示例

```
POST /logstores/my-logstore/index HTTP/1.1
Header:
{
Authorization: LOG yourAccessKeyId:yourSignature
x-log-bodyrawsize: 0
User-Agent: sls-java-sdk-v-0.6.1
x-log-apiversion: 0.6.0
Host: my-project.cn-shanghai.log.aliyuncs.com
x-log-signaturemethod: hmac-sha1
Date: Mon, 07 May 2018 09:47:20 GMT
Content-Type: application/json
Content-MD5: 1860B805A6AA5288B97B32CE3B519112
```

```
Content-MD5: 1800B805A0AA5288B57B52CF5B319112
Content-Length: 316
Connection: Keep-Alive
}
Body :
{
 "line": {
  "token": [
   ",",
   " ",
   "'",
   "\"",
   ";",
   "=",
   "(",
   ")",
   "[",
   "]",
   "{",
   "}",
   "?",
   "@",
   "&",
   "<",
   ">",
   "/",
   ":",
   "\n",
   "\t",
   "\r"
  ]
 },
 "keys": {
  "agent": {
   "doc_value": true,
   "caseSensitive": true,
   "alias": "agent_alias",
   "type": "text",
   "token": [
    ",",
    " ",
    "'",
    "\"",
    ";",
    "=",
    "(",
    ")",
    "[",
    "]",
    "{",
    "}",
    "?",
    "@",
    "&",
    "<",
    ">",
    "/",
    ":",
    "\n",
    "\t",
    "\r"
```

```
    ]
  }
 }
}
```

- 正常返回示例

```
HTTP/1.1 200
Server: nginx/1.12.1
Content-Length: 0
Connection: close
Access-Control-Allow-Origin: *
Date: Mon, 07 May 2018 09:47:21 GMT
x-log-requestid: 5AF020A98CBAA2EF52AB66C9
```

错误码

HTTP状态码	错误码	错误信息	描述
400	IndexInfoInvalid	Required field token is lacking or of error format.	缺少必要的字段标记或格式配置错误。
400	IndexAlreadyExist	Logstore index is already created.	日志索引已经创建。
404	ProjectNotExist	The Project does not exist : *projectName*.	Project不存在。
404	LogStoreNotExist	Logstore *logstoreName* dose not exist.	Logstore不存在。
500	InternalServerError	Specified Server Error Message.	内部服务调用错误。

更多错误码，请参见通用错误码。

10.20. DeleteIndex

调用DeleteIndex接口删除指定Logstore的索引。

请求语法

```
DELETE /logstores/logstoreName/index HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignaturex-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
Host: ProjectName.Endpoint
x-log-signaturemethod: hmac-sha1
Date: GMT Date
Content-Type: application/x-protobuf
Connection: Keep-Alive
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  DeleteIndex接口无特有请求头，关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	my-project	Project名称。
logstoreName	String	是	my-logstore	Logstore名称。

返回数据

- 响应头

  DeleteIndex接口无特有响应头，关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功。该接口调用成功后无任何响应元素。

示例

- 请求示例

```
DELETE /logstores/my-logstore/index HTTP/1.1
Header :
{
Authorization: LOG yourAccessKeyId:yourSignature
x-log-bodyrawsize: 0
User-Agent: sls-java-sdk-v-0.6.1
x-log-apiversion: 0.6.0
Host: my-project.cn-shanghai.log.aliyuncs.com
x-log-signaturemethod: hmac-sha1
Date: Sun, 06 May 2018 13:22:57 GMT
Content-Type: application/x-protobuf
Connection: Keep-Alive
```

- 正常返回示例

```
HTTP/1.1 200
Server: nginx/1.12.1
Content-Length: 0
Connection: close
Access-Control-Allow-Origin: *
Date: Sun, 06 May 2018 13:22:57 GMT
x-log-requestid: 5AEF01B1A458E41C2F8326A8
```

错误码

HTTP状态码	错误码	错误信息	描述
400	ParameterInvalid	Log store index not created.	没有创建日志索引。
404	ProjectNotExist	The Project does not exist : *projectName*.	Project不存在。
404	LogStoreNotExist	logstore *logstoreName* does not exist.	Logstore不存在。
500	InternalServerError	Specified Server Error Message.	内部服务调用错误。

更多错误码，请参见通用错误码。

10.21. GetIndex

调用GetIndex接口查询指定Logstore的索引。

请求语法

```
GET /logstores/logstoreName/index HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignature
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
Host: ProjectName.Endpoint
x-log-signaturemethod: hmac-sha1
Date: GMT Date
Content-Type: application/x-protobuf
Connection: Keep-Alive
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  GetIndex接口无特有请求头，关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	my-project	Project名称。
logstoreName	String	是	logstore-4	Logstore名称。

返回数据

- 响应头

  GetIndex接口无特有响应头，关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  GetIndex请求成功，其响应Body会包括指定Project和Logstore的索引，具体如下：

参数名称	数据类型	示例值	描述
index_mode	String	v2	索引类型。
keys	Object	无	字段索引配置。key为字段名称，value为索引配置。
line	Object	无	全文索引配置。
storage	String	pg	存储类型，目前固定取值为pg。
ttl	Integer	30	索引文件生命周期，支持7天、30天、90天。
lastModifyTime	Integer	1524155379	索引最后更新时间。Unix时间戳格式，表示从1970-1-1 00:00:00 UTC计算起的秒数。

- 全文索引配置包含如下信息：

参数名称	数据类型	示例值	描述
caseSensitive	Boolean	false	是否大小写敏感。 ■ true：大小写敏感。 ■ false：大小写不敏感。
chn	Boolean	false	是否包含中文。 ■ true：包含中文。 ■ false：不包含中文。
token	Array	["\n","\t","\r"]	分词符列表。
include_keys	Array	无	包含的字段列表。
exclude_keys	Array	无	排除的字段列表。

- 字段索引配置包含如下信息：

属性名称	类型	示例值	描述
type	String	text	字段类型。
alias	String	无	字段别名。
chn	Boolean	false	是否包含中文。仅当type参数取值为text时，必须设置。 ■ true：包含中文。 ■ false：不包含中文。
token	Array	["\n","\t","\r"]	分词符列表。仅当type参数取值为text时，必须设置。
caseSensitive	Boolean	false	是否大小写敏感。仅当type参数取值为text时，必须设置。 ■ true：大小写敏感。 ■ false：大小写不敏感。
doc_value	Boolean	true	是否开启字段统计。 ■ true：开启统计。 ■ false：不开启统计。

示例

- 请求示例

```
GET /logstores/logstore-4/index HTTP/1.1
Header :
{
Authorization: LOG yourAccessKeyId:yourSignature
x-log-bodyrawsize: 0
User-Agent: sls-java-sdk-v-0.6.1
x-log-apiversion: 0.6.0
Host: my-project.cn-shanghai.log.aliyuncs.com
x-log-signaturemethod: hmac-sha1
Date: Sun, 06 May 2018 13:08:42 GMT
Content-Type: application/x-protobuf
Connection: Keep-Alive
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header :
{
Server: nginx/1.12.1
Content-Type: application/json
Content-Length: 712
Connection: close
Access-Control-Allow-Origin: *
Date: Sun, 06 May 2018 13:08:42 GMT
x-log-requestid: 5AEEFE5A8B8AEB5E6C82B395
}
Body :
{
 "index_mode": "v2",
 "keys": {
  "agent": {
   "alias": "",
   "caseSensitive": false,
   "chn": false,
   "doc_value": true,
   "token": [
    ",",
    " ",
    "'",
    "\"",
    ";",
    "=",
    "(",
    ")",
    "[",
    "]",
    "{",
    "}",
    "?",
    "@",
    "&",
    "<",
    ">",
    "/",
    ":",
    "\n",
    "\t",
    "\r"
   ],
```

```
    "type": "text"
   },
   "bytes": {
    "alias": "",
    "doc_value": true,
    "type": "long"
   },
   "remote_ip": {
    "alias": "",
    "caseSensitive": false,
    "chn": false,
    "doc_value": true,
    "token": [
     ",",
     " ",
     "'",
     "\"",
     ";",
     "=",
     "(",
     ")",
     "[",
     "]",
     "{",
     "}",
     "?",
     "@",
     "&",
     "<",
     ">",
     "/",
     ":",
     "\n",
     "\t",
     "\r"
    ],
    "type": "text"
   },
   "response": {
    "alias": "",
    "doc_value": true,
    "type": "long"
   }
  },
  "line": {
   "caseSensitive": false,
   "chn": false,
   "token": [
    ",",
    " ",
    "'",
    "\"",
    ";",
    "=",
    "(",
    ")",
    "[",
    "]",
    "{",
    "}",
    "?"
```

```
    "?",
    "@",
    "&",
    "<",
    ">",
    "/",
    ":",
    "\n",
    "\t",
    "\r"
   ]
  },
  "storage": "pg",
  "ttl": 30,
  "lastModifyTime": 1524155379
}
```

错误码

HTTP状态码	错误码	错误信息	描述
400	IndexConfigNotExist	index config doesn't exist.	查询的索引不存在。
404	ProjectNotExist	The Project does not exist : *projectName*.	Project不存在。
404	LogStoreNotExist	logstore *logstoreName* dose not exist.	Logstore不存在。
500	InternalServerError	Specified Server Error Message.	服务器错误信息。

更多错误码，请参见通用错误码。

10.22. PutWebtracking

调用PutWebTracking接口将多条日志合并进行采集。

接口说明

- 适用于在网页或者客户端采集日志的场景。
- 使用Web Tracking采集日志时，单个请求只能写入一条日志。更多信息，请参见使用Web Tracking采集日志。
- 针对日志量较大的场景，可以调用PutWebTracking接口将多条日志合并为一次请求。
- 使用PutWebTracking接口写入日志时，需要先为Logstore打开Web Tracking开关。更多信息，请参见使用Web Tracking采集日志。
- 该接口不支持同时写入多个Topic的日志数据。

请求语法

```
POST ProjectName.Endpoint/logstores/logstoreName/track HTTP/1.1
x-log-apiversion: 0.6.0
x-log-bodyrawsize: 1234
x-log-compresstype: lz4
{
 "__topic__": "topic",
 "__source__": "source",
 "__logs__": [
  {
   "key1": "value1",
   "key2": "value2"
  },
  {
   "key1": "value1",
   "key2": "value2"
  }
 ],
 "__tags__": {
  "tag1": "value1",
  "tag2": "value2"
 }
}
```

请求参数

- 请求头

  仅支持如下三个请求头，在调用PutWebTracking接口时前两个为必选，格式和含义请参见公共请求头。

  - `x-log-apiversion: 0.6.0`
  - `x-log-bodyrawsize: 1234`
  - `x-log-compresstype: lz4`

  如果发送的数据没有经过任何压缩，不需要指定x-log-compresstype。如果需要对数据压缩发送，当前仅支持lz4和Deflate算法，其分别对应的请求头为：`x-log-compresstype: lz4` 或 `x-log-compresstype: deflate`。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	my-project	Project名称。
logstoreName	String	是	logstore-test	Logstore名称。
endpoint	String	是	cn-hangzhou.log.aliyuncs.com	日志服务的Endpoint，请参见服务入口。
__topic__	String	否	topic	日志主题。
__source__	String	否	source	日志来源。

参数名称	数据类型	是否必填	示例值	描述
__logs__	List	是	[{"key1": value1", "key2": "value2"}, {"key1": "value1","key2": "value2"}]	日志内容列表。每个元素为一个JSON对象，表示一条日志。 **说明** WebTracking采集的日志时间为日志到达服务端的时间，每条日志中无需设置__time__字段，如果存在该字段，将被服务端使用日志到达的时间覆盖。
__tags__	Object	否	{"tag1": "value1", "tag2": "value2" }	日志标签。

返回数据

- 响应头

  PutWebtracking接口无特有响应头。关于Log Service API的公共响应头，请参考公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功。该接口调用成功后无任何响应元素。

示例

- 请求示例

```
POST my-project.cn-hangzhou.log.aliyuncs.com/logstores/logstore-test/track HTTP/1.1
Header :
{
x-log-apiversion: 0.6.0
x-log-bodyrawsize: 1234
x-log-compresstype: lz4
}
Body :
{
 "__topic__": "topic",
 "__source__": "source",
 "__logs__": [
  {
   "foo": "bar",
   "foo1": "bar1"
  },
  {
   "bar": "foo",
   "bar1": "foo1"
  }
 ],
 "__tags__": {
  "tag1": "value1",
  "tag2": "value2"
 }
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header:
{
  'Date': 'Wed, 20 May 2019 07:35:00 GMT',
  'Content-Length': '0',
  'x-log-requestid': '5642EFA499248C827B012B39',
  'Connection': 'close',
  'Server': 'nginx/1.6.1'
}
```

错误码

HTTP状态码	错误码	错误消息	描述
400	PostBodyInvalid	Protobuffer content cannot be parsed.	Protobuffer内容无效，无法解析。
400	InvalidTimestamp	Invalid timestamps are in logs.	日志内容中有无效的日志时间戳。
400	InvalidEncoding	Non-UTF8 characters are in logs.	日志内容中有非UTF-8字符。
400	InvalidKey	Invalid keys are in logs.	日志内容中有无效的key。
400	PostBodyTooLarge	Logs must be less than or equal to 3 MB and 4096 entries.	日志内容过大，包含的日志必须小于3MB和4096条。
400	PostBodyUncompressError	Failed to decompress logs.	日志内容解压失败。
499	PostBodyInvalid	The post data time is out of range.	日志中时间范围不在[-7*24Hour,+15Min]有效范围内。
404	LogStoreNotExist	logstore *logstoreName* does not exist.	Logstore不存在。
404	ProjectNotExist	The Project does not exist : *projectName*.	Project不存在。

更多错误码，请参见通用错误码。

11.Logtail机器组相关接口

11.1. CreateMachineGroup

调用CreateMachineGroup接口创建一个机器组。

请求语法

```
POST /machinegroups HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignature
Content-Type: application/json
Content-Length: Content Length
Content-MD5: Content MD5
Date: GMT Date
Host: ProjectName.Endpoint
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
{
  "groupName" : "testgroup",
  "groupType" : "",
  "groupAttribute" : {
    "externalName" : "testgroup",
    "groupTopic": "testgrouptopic"
  },
  "machineIdentifyType" : "ip",
  "machineList" : [
    "192.168.2.1",
    "192.168.2.2"
  ]
}
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  CreateMachineGroup接口无特有请求头。关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	ali-test-project	Project名称。
groupName	String	是	test-machine-group	机器组名称。其命名规则如下： ○ 同一个Project下，机器组名称不可重复。 ○ 只能包括小写字母、数字、短划线（-）和下划线（_）。 ○ 必须以小写字母或者数字开头和结尾。 ○ 长度为3-128字符。

参数名称	数据类型	是否必填	示例值	描述
machineIdentifyType	String	是	ip	机器标识类型。 ○ ip：IP地址机器组。 ○ userdefined：用户自定义标识机器组。
groupAttribute	Json	是	不涉及	机器组的属性。详细请参考下表groupAttribute参数说明。
machineList	Array	是	["192.168.2.1", "192.168.2.2"]	机器组的标识信息。 ○ 如果machineIdentifyType配置为ip，则此处填写服务器的IP地址。 ○ 如果machineIdentifyType配置为userdefined，则此处填写自定义的标识。
groupType	String	否	Armory	机器组类型，可选值为空或者Armory。

groupAttribute参数说明如下表所示：

参数名称	数据类型	是否必填	示例值	描述
groupTopic	String	否	testtopic	机器组的日志主题。
externalName	String	否	testgroup	机器组所依赖的外部管理系统（Armory）标识。

返回数据

- 响应头

  CreateMachineGroup接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功。该接口调用成功后无任何响应元素。

示例

- 请求示例

```
POST /machinegroups HTTP/1.1
Header:
{
  "x-log-apiversion": "0.6.0",
  "Authorization": "LOG yourAccessKeyId:yourSignature",
  "Host": "ali-test-project.cn-hangzhou-devcommon-intranet.sls.aliyuncs.com",
  "Date": "Tue, 10 Nov 2015 17:57:33 GMT",
  "Content-Length": "187",
  "x-log-signaturemethod": "hmac-sha1",
  "Content-MD5": "82033D507DEAAD72067BB58DFDCB590D",
  "User-Agent": "sls-java-sdk-v-0.6.0",
  "Content-Type": "application/json",
  "x-log-bodyrawsize": "0"
}
Body:
{
  "groupName": "test-machine-group",
  "groupType": "",
  "machineIdentifyType": "ip",
  "groupAttribute":  {
    "groupTopic": "testtopic",
    "externalName": "testgroup"
  },
  "machineList":  [
    "192.168.2.1",
    "192.168.2.2"
  ]
}
```

- 返回示例

```
HTTP/1.1 200 OK
Header:
{
  "Date": "Tue, 10 Nov 2015 17:57:33 GMT",
  "Content-Length": "0",
  "x-log-requestid": "5642300D99248CB76D005D36",
  "Connection": "close",
  "Server": "nginx/1.6.1"
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	Project *projectName* does not exist.	Project不存在。
400	MachineGroupAlreadyExist	group *groupName* already exists.	机器组已存在。
400	InvalidParameter	invalid group resource json	无效机器组参数。
500	InternalServerError	Internal server error	内部服务调用错误。

更多错误码，请参见通用错误码。

11.2. DeleteMachineGroup

调用DeleteMachineGroup接口删除机器组。如果机器组已应用Logtail采集配置，则删除机器组后，会解绑对应的Logtail配置。

请求语法

```
DELETE /machinegroups/groupName HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignature
Date: GMT Date
Host: ProjectName.Endpoint
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  DeleteMachineGroup接口无特有请求头。关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	ali-test-project	Project名称。
groupName	String	是	test-machine-group-4	机器组名称。

返回数据

- 响应头

  DeleteMachineGroup接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功。该接口调用成功后无任何响应元素。

示例

- 请求示例

```
DELETE /machinegroups/test-machine-group-4 HTTP/1.1
Header :
{
  "x-log-apiversion": "0.6.0",
  "Authorization": "LOG yourAccessKeyId:yourSignature",
  "Host": "ali-test-project.cn-hangzhou-devcommon-intranet.sls.aliyuncs.com",
  "Date": "Tue, 10 Nov 2015 19:13:28 GMT",
  "Content-Length": "0",
  "x-log-signaturemethod": "hmac-sha1",
  "User-Agent": "sls-java-sdk-v-0.6.0",
  "Content-Type": "application/x-protobuf",
  "x-log-bodyrawsize": "0"
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header:
{
  "Date": "Tue, 10 Nov 2015 19:13:28 GMT",
  "Content-Length": "0",
  "x-log-requestid": "564241D899248C827B000CFE",
  "Connection": "close",
  "Server": "nginx/1.6.1"
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	The Project does not exist : *projectName*	Project不存在。
404	MachineGroupNotExist	MachineGroup *groupName* does not exist.	机器组不存在。
500	InternalServerError	Internal server error.	内部服务调用错误。

更多错误码，请参见通用错误码。

11.3. UpdateMachineGroup

调用UpdateMachineGroup接口修改机器组配置信息。

请求语法

```
PUT /machinegroups/groupName HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignature
Content-Type:application/json
Date: GMT Date
Host: ProjectName.Endpoint
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
{
  "groupName": "test-machine-group",
  "groupType" : "",
  "groupAttribute" : {
    "externalName" : "testgroup",
    "groupTopic": "testgrouptopic"
  },
  "machineIdentifyType" : "ip",
  "machineList" : [
    "192.168.3.1",
    "192.168.3.2"
  ]
}
```

请求参数

- 请求头

  UpdateMachineGroup接口无特有请求头。关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	ali-test-project	Project名称。
groupName	String	是	test-machine-group	机器组名称。
groupType	String	否	无	机器组类型，默认为空。
machineIdentifyType	String	是	userdefined	机器组标识类型。 ◦ ip：IP地址机器组。 ◦ userdefined：用户自定义标识机器组。
groupAttribute	Json	是	{"externalName" : "testgroup", "groupTopic": "testgrouptopic"}	机器组的属性，默认为空。详细请参考下表groupAttribute参数说明。
machineList	Array	是	[uu_id_1，uu_id_2]	机器组的标识信息。 ◦ 如果machineIdentifyType配置为ip，则此处填写服务器的IP地址。 ◦ 如果machineIdentifyType配置为userdefined，则此处填写自定义的标识。

groupAttribute参数说明如下表所示：

参数名称	数据类型	是否必填	示例值	描述
groupTopic	String	否	testtopic2	机器组的日志主题。默认为空。
externalName	String	否	testgroup2	机器组所依赖的外部管理系统（Armory）标识。默认为空。

返回数据

- 响应头

  UpdateMachineGroup接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功。该接口调用成功后无任何响应元素。

示例

- 请求示例

```
PUT /machinegroups/test-machine-group HTTP/1.1
Header:
{
  "x-log-apiversion": "0.6.0",
  "Authorization": "LOG yourAccessKeyId:yourSignature",
  "Host": "ali-test-project.cn-hangzhou-devcommon-intranet.sls.aliyuncs.com",
  "Date": "Tue, 10 Nov 2015 18:41:43 GMT",
  "Content-Length": "194",
  "x-log-signaturemethod": "hmac-sha1",
  "Content-MD5": "2CEBAEBE53C078891527CB70A855BAF4",
  "User-Agent": "sls-java-sdk-v-0.6.0",
  "Content-Type": "application/json",
  "x-log-bodyrawsize": "0"
}
Body:
{
  "groupName": "test-machine-group",
  "groupType": "",
  "machineIdentifyType": "userdefined",
  "groupAttribute":  {
    "groupTopic": "testtopic2",
    "externalName": "testgroup2"
  },
  "machineList":  [
    "uu_id_1",
    "uu_id_2"
  ]
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header:
{
  "Date": "Tue, 10 Nov 2015 18:41:43 GMT",
  "Content-Length": "0",
  "x-log-requestid": "56423A6799248CA57B00035C",
  "Connection": "close",
  "Server": "nginx/1.6.1"
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	Project *projectName* does not exist.	Project不存在。
404	GroupNotExist	group *groupName* does not exist.	机器组不存在。
400	InvalidParameter	invalid group resource json	无效机器组参数。
500	InternalServerError	Internal server error	内部服务调用错误。

更多错误码，请参见通用错误码。

11.4. ListMachineGroup

调用ListMachineGroup接口列出目标Project下的机器组。

请求语法

```
GET /machinegroups?offset=1&size=100 HTTP/1.1
Authorization: AuthorizationString
Date: GMT Date
Host: ProjectName.Endpoint
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  ListMachineGroup接口无特有请求头。关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
offset	Integer	否	1	查询开始行。默认值为0。
size	Integer	否	10	分页查询时，设置的每页行数。最大值为500。
groupName	String	否	test-machine-group	机器组名称。用于过滤机器组，支持部分匹配。

返回数据

- 响应头

  ListMachineGroup接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，表示请求成功。请求成功后，其响应Body中包含目标Project下的机器组列表，如下所示：

参数名称	数据类型	示例值	描述
count	Integer	2	当前页返回的机器组数。
total	Integer	2	符合查询条件的机器组总数。
machinegroups	Json Array	["test-machine-group-1", "test-machine-group-2"]	符合查询条件的机器组列表。

示例

- 请求示例

```
GET /machinegroups?groupName=test-machine-group&offset=0&size=3 HTTP/1.1
Header :
{
  "x-log-apiversion": "0.6.0",
  "Authorization": "LOG yourAccessKeyId:yourSignature",
  "Host": "ali-test-project.cn-hangzhou-devcommon-intranet.sls.aliyuncs.com",
  "Date": "Tue, 10 Nov 2015 18:34:44 GMT",
  "Content-Length": "0",
  "x-log-signaturemethod": "hmac-sha1",
  "User-Agent": "sls-java-sdk-v-0.6.0",
  "Content-Type": "application/x-protobuf",
  "x-log-bodyrawsize": "0"
}
```

- 返回示例

```
HTTP/1.1 200 OK
Header :
{
  "Date": "Tue, 10 Nov 2015 18:34:44 GMT",
  "Content-Length": "83",
  "x-log-requestid": "564238C499248C8F7B0001DE",
  "Connection": "close",
  "Content-Type": "application/json",
  "Server": "nginx/1.6.1"
}
Body :
{
  "machinegroups":  [
    "test-machine-group-1",
    "test-machine-group-2"
  ],
  "count": 2,
  "total": 2
}
```

错误码

HTTP状态码	错误码	错误信息	描述
500	InternalServerError	internal server error	内部服务调用错误。

更多错误码，请参见通用错误码。

11.5. GetMachineGroup

调用GetMachineGroup接口查看目标机器组的具体信息。

请求语法

```
GET /machinegroups/groupName HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignature
Date: GMT Date
Host: ProjectName.Endpoint
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  GetMachineGroup接口无特有请求头。关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	ali-test-project	Project名称。
groupName	String	是	test-machine-group	机器组名称。

返回数据

- 响应头

  GetMachineGroup接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功。请求成功后，其响应Body中包含目标机器组的具体信息，如下所示：

参数名称	数据类型	示例值	描述
groupName	String	test-machine-group	机器组名称。
groupType	String	Armory	机器组类型，值为空或者Armory。
machineIdentifyType	String	ip	机器组标识的类型。 ◦ ip：IP地址机器组。 ◦ userdefined：用户自定义标识机器组。
groupAttribute	Json Object	{"externalName": "testgroup", "groupTopic": "testtopic"}	机器组的属性。详细请参考下表groupAttribute参数说明。
machineList	Json Array	["127.0.0.1", "127.0.0.2"]	机器组的标识信息。 ◦ 如果machineIdentifyType配置为ip，则此处填写服务器的IP地址。 ◦ 如果machineIdentifyType配置为userdefined，则此处填写自定义的标识。
createTime	integer	1447178253	机器组的创建时间。Unix时间戳格式，表示从1970-1-1 00:00:00 UTC计算起的秒数。
lastModifyTime	integer	1447178253	最近一次更新机器组的时间。Unix时间戳格式，表示从1970-1-1 00:00:00 UTC计算起的秒数。

groupAttribute参数说明如下表所示：

参数名称	数据类型	示例值	描述
groupTopic	String	testtopic	机器组的日志主题。
externalName	String	testgroup	机器组所依赖的外部管理系统（Armory）标识。

示例

- 请求示例

```
GET /machinegroups/test-machine-group HTTP/1.1
Header:
{
  "x-log-apiversion": "0.6.0",
  "Authorization": "LOG yourAccessKeyId:yourSignature",
  "Host": "ali-test-project.cn-hangzhou-devcommon-intranet.sls.aliyuncs.com",
  "Date": "Tue, 10 Nov 2015 18:15:24 GMT",
  "Content-Length": "0",
  "x-log-signaturemethod": "hmac-sha1",
  "User-Agent": "sls-java-sdk-v-0.6.0",
  "Content-Type": "application/x-protobuf",
  "x-log-bodyrawsize": "0"
}
```

- 返回示例

```
HTTP/1.1 200 OK
Header:
{
  "Date": "Tue, 10 Nov 2015 18:15:23 GMT",
  "Content-Length": "239",
  "x-log-requestid": "5642343B99248CB36D0060B8",
  "Connection": "close",
  "Content-Type": "application/json",
  "Server": "nginx/1.6.1"
}
Body:
{
  "groupName": "test-machine-group",
  "groupType": "",
  "groupAttribute":  {
    "externalName": "testgroup",
    "groupTopic": "testtopic"
  },
  "machineIdentifyType": "ip",
  "machineList":  [
    "127.0.0.1",
    "127.0.0.2"
  ],
  "createTime": 1447178253,
  "lastModifyTime": 1447178253
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	Project *projectName* does not exist.	Project不存在。
404	GroupNotExist	group *groupName* does not exist.	机器组不存在。
500	InternalServerError	Internal server error	内部服务调用错误。

更多错误码，请参见通用错误码。

11.6. ApplyConfigToMachineGroup

调用ApplyConfigToMachineGroup接口将Logtail配置应用到机器组。

请求语法

```
PUT /machinegroups/groupName HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignature
Date: GMT Date
Host: ProjectName.Endpoint
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  ApplyConfigToMachineGroup接口无特有请求头。关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	ali-test-project	Project名称。
groupName	String	是	sample-group	机器组名称。
configName	String	是	logtail-config-sample	Logtail配置名称。

返回数据

- 响应头

  ApplyConfigToMachineGroup接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功。该接口调用成功后无任何响应元素。

示例

- 请求示例

```
PUT /machinegroups/sample-group/configs/logtail-config-sample
Header :
{
  "Content-Length": 0,
  "x-log-signaturemethod": "hmac-sha1",
  "x-log-bodyrawsize": 0,
  "User-Agent": "log-python-sdk-v-0.6.0",
  "Host": "ali-test-project.cn-hangzhou-devcommon-intranet.sls.aliyuncs.com",
  "Date": "Mon, 09 Nov 2015 09:44:43 GMT",
  "x-log-apiversion": "0.6.0",
  "Authorization": "LOG yourAccessKeyId:yourSignature"
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header：
{
  "date": "Mon, 09 Nov 2015 09:44:43 GMT",
  "connection": "close",
  "x-log-requestid": "56406B0B99248CAA230BA094",
  "content-length": "0",
  "server": "nginx/1.6.1"
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	The Project does not exist : *projectName*	Project不存在。
404	GroupNotExist	group *groupName* does not exist.	机器组不存在。
404	ConfigNotExist	Config *configNam*does not exist.	Logtail配置不存在。
500	InternalServerError	Internal server error.	内部服务调用错误。

更多错误码，请参见通用错误码。

11.7. RemoveConfigFromMachineGroup

调用RemoveConfigFromMachineGroup接口从机器组中移除Logtail配置。

请求语法

```
DELETE /machinegroups/groupName HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignature
Date: GMT Date
Host: ProjectName.Endpoint
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  RemoveConfigFromMachineGroup接口无特有请求头。关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必须	示例值	描述
projectName	String	是	ali-test-project	Project名称。
groupName	String	是	sample-group	机器组名称。
configName	String	是	logtail-config-sample	Logtail配置名称。

返回数据

- 响应头

  RemoveConfigFromMachineGroup接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功。该接口调用成功后无任何响应元素。

示例

- 请求示例

```
DELETE /machinegroups/sample-group/configs/logtail-config-sample
{
  "Content-Length": 0,
  "x-log-signaturemethod": "hmac-sha1",
  "x-log-bodyrawsize": 0,
  "User-Agent": "log-python-sdk-v-0.6.0",
  "Host": "ali-test-project.cn-hangzhou-devcommon-intranet.sls.aliyuncs.com",
  "Date": "Mon, 09 Nov 2015 09:48:48 GMT",
  "x-log-apiversion": "0.6.0",
  "Authorization": "LOG <yourAccessKeyId>:<yourSignature>"
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header :
{
  "date": "Mon, 09 Nov 2015 09:48:48 GMT",
  "connection": "close",
  "x-log-requestid": "56406C0099248CAA230BE135",
  "content-length": "0",
  "server": "nginx/1.6.1"
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	MachineGroupNotExist	MachineGroup *groupName* does not exist.	机器组不存在。
404	ConfigNotExist	Config *configName* does not exist.	Logtail配置不存在。
500	InternalServerError	Internal server error.	内部服务调用错误。

更多错误码，请参见通用错误码。

11.8. GetAppliedConfigs

调用GetAppliedConfigs接口获取目标机器组上已经被应用的Logtail配置列表。

请求语法

```
GET /machinegroups/groupName/configs HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignature
Date: GMT Date
Host: ProjectName.Endpoint
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  GetAppliedConfigs接口无特有请求头。关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	ali-test-project	Project名称。
groupName	String	是	test-machine-group	机器组名称。

返回数据

- 响应头

  GetAppliedConfigs接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功，其响应Body会包括指定机器组下的所有Logtail配置列表，具体格式如下：

参数名称	数据类型	示例值	描述
count	Integer	3	Logtail配置数量。
configs	Array	["two", "three", "test_logstore"]	Logtail配置名称列表。

示例

- 请求示例

```
GET /machinegroups/test-machine-group/configs HTTP/1.1
Header :
{
x-log-apiversion: 0.6.0
Authorization: LOG yourAccessKeyId:yourSignature
Host: ali-test-project.cn-hangzhou-devcommon-intranet.sls.aliyuncs.com
Date: Tue, 10 Nov 2015 19:45:48 GMT
Content-Length: 0
x-log-signaturemethod: hmac-sha1
User-Agent: sls-java-sdk-v-0.6.0
Content-Type: application/x-protobuf
x-log-bodyrawsize: 0
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header :
{
Date: Tue, 10 Nov 2015 19:45:48 GMT
Content-Length: 53
x-log-requestid: 5642496C99248C8C7B00173F
Connection: close
Content-Type: application/json
Server: nginx/1.6.1
}
Body :
{
  "configs":  [
    "two",
    "three",
    "test_logstore"
  ],
  "count": 3
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	The Project does not exist : *projectName*	Project不存在。
404	MachineGroupNotExist	MachineGroup *groupName* does not exist.	机器组不存在。
500	InternalServerError	Internal server error.	内部服务调用错误。

更多错误码，请参见通用错误码。

12.Logtail配置相关接口

12.1. CreateConfig

调用CreateConfig接口创建Logtail配置。

接口说明

每个Project最多可创建100个Logtail配置。

请求语法

```
POST /configs HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignature
Content-Type:application/json
Date: GMT Date
Host: ProjectName.Endpoint
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
{
  "configName": "testcategory1",
  "inputType": "file",
  "inputDetail": {
    "logType": "common_reg_log",
    "logPath": "/var/log/httpd/",
    "filePattern": "access*.log",
    "localStorage": true,
    "timeFormat": "%Y/%m/%d %H:%M:%S",
    "logBeginRegex": ".*",
    "regex": "(\w+)(\s+)",
    "key" :["key1", "key2"],
    "filterKey":["key1"],
    "filterRegex":["regex1"],
    "fileEncoding":"utf8",
    "topicFormat": "none"
  },
  "outputType": "LogService",
  "outputDetail":
  {
    "logstoreName": "perfcounter"
  }
}
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  CreateConfig接口无特有请求头。关于Log Service API的公共请求头，请参见公共请求头。

- 请求参数

  关于请求参数说明和各种模式的Logtail配置样例，请参见Logtail配置。

返回数据

- 响应头

  CreateConfig接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功。该接口调用成功后无任何响应元素。

示例

- 请求示例

```
POST /configs HTTP/1.1
Header :
{
Content-Length: 737
Host: ali-test-project.cn-hangzhou-devcommon-intranet.sls.aliyuncs.com
x-log-bodyrawsize: 737
Content-MD5: FBA01ECF7255BE143379BC70C56BBF68
x-log-signaturemethod: hmac-sha1
Date: Mon, 09 Nov 2015 07:45:30 GMT
x-log-apiversion: 0.6.0
User-Agent: log-python-sdk-v-0.6.0
Content-Type: application/json
Authorization: LOG yourAccessKeyId:yourSignature
}
Body:
{
  "configName": "sample-logtail-config",
  "inputType": "file",
  "inputDetail": {
    "logType": "common_reg_log",
    "logPath": "/var/log/httpd/",
    "filePattern": "access*.log",
    "localStorage": true,
    "timeFormat": "%d/%b/%Y:%H:%M:%S",
    "logBeginRegex": "\\d+\\.\\d+\\.\\d+\\.\\d+ - .*",
    "regex": "([\\d\\.]+) \\S+ \\S+ \\[(\\S+) \\S+\\] \"(\\w+) ([^\"]*)\" ([\\d\\.]+) (\\d+) (\\d+) (\\d+|-) \"([^\"]*)\" \"([^\"]*)\".*",
    "key": ["ip", "time", "method", "url", "request_time", "request_length", "status", "length", "ref_url", "browser"],
    "filterKey": [],
    "filterRegex": [],
    "topicFormat": "none",
    "fileEncoding": "utf8"
  },
  "outputType": "LogService",
  "outputDetail":
  {
    "logstoreName": "sls-test-logstore"
  }
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header
{
date: Mon, 09 Nov 2015 07:45:30 GMT
connection: close
x-log-requestid: 56404F1A99248CA26C002180
content-length: 0
server: nginx/1.6.1
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	The Project does not exist : *projectName*.	Project不存在。
400	ConfigAlreadyExist	config *configname* already exists.	Logtail配置已存在。
400	InvalidParameter	Invalid config resource json.	无效的Logtail配置参数。
500	InternalServerError	Internal server error.	内部服务调用错误。

更多错误码，请参见通用错误码。

12.2. ListConfig

调用ListConfig接口查询指定Project下所有的Logtail配置。

请求语法

```
GET /configs?offset=0&size=100 HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignature
Date: GMT Date
Host: ProjectName.Endpoint
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  ListConfig接口无特有请求头。关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	ali-test-project	Project名称。
offset	Integer	否	0	查询开始行。默认值为0。
size	Integer	否	10	分页查询时，设置的每页行数。最大值为500。
logstoreName	String	否	logstore-4	Logstore名称。
configName	String	否	logtail-config-sample	Logtail配置名称。

返回数据

- 响应头

  ListConfig接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，表示请求成功。ListConfig接口的响应Body中包括Logtail采集配置信息，如下所示：

参数名称	数据类型	示例值	描述
count	Integer	3	当前页返回的Logtail配置数量。
total	Integer	3	符合查询条件的Logtail配置总数。
configs	Array	["logtail-config-sample", "logtail-config-sample-2", "logtail-config-sample-3"]	当前页返回的Logtail配置列表。

示例

- 请求示例

```
GET /configs?offset=0&size=10 HTTP/1.1
Header:
{
  "Content-Length": 0,
  "x-log-signaturemethod": "hmac-sha1",
  "x-log-bodyrawsize": 0,
  "User-Agent": "log-python-sdk-v-0.6.0",
  "Host": "ali-test-project.cn-hangzhou-devcommon-intranet.sls.aliyuncs.com",
  "Date": "Mon, 09 Nov 2015 09:19:13 GMT",
  "x-log-apiversion": "0.6.0",
  "Authorization": "LOG yourAccessKeyId:yourSignature"
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header:
{
  "content-length": "103",
  "server": "nginx/1.6.1",
  "connection": "close",
  "date": "Mon, 09 Nov 2015 09:19:13 GMT",
  "content-type": "application/json",
  "x-log-requestid": "5640651199248CAA2300C2BA"
}
Body:
{
  "count": 3,
  "total": 3,
  "configs":
  [
    "logtail-config-sample",
    "logtail-config-sample-2",
    "logtail-config-sample-3"
  ]
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	Project *ProjectName* does not exist.	Project不存在。

HTTP状态码	错误码	错误信息	描述
404	LogstoreNotExist	logstore *logstoreName* does not exist.	Logstore不存在。
404	ConfigNotExist	config *configName* does not exist.	Logtail配置不存在。
500	InternalServerError	internal server error.	内部服务调用错误。

更多错误码，请参见通用错误码。

12.3. GetAppliedMachineGroups

调用GetAppliedMachineGroups接口获取已绑定指定Logtail配置的机器组列表。

请求语法

```
GET /configs/configName/machinegroups HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignature
Date: GMT Date
Host: Projectname.Endpoint
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  GetAppliedMachineGroups接口无特有请求头。关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	ali-test-project	Project名称。
configName	String	是	logtail-config-sample	Logtail配置名称。

返回数据

- 响应头

  GetAppliedMachineGroups接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，表示请求成功。请求成功后，其响应Body中包含机器组信息，如下所示：

参数名称	数据类型	示例值	描述
count	Integer	1	返回的机器组数量。
machinegroups	Array	["sample-group1","sample-group2"]	返回的机器组名称列表。

示例

- 请求示例

```
GET /configs/logtail-config-sample/machinegroups
Header:
{
  "Content-Length": 0,
  "x-log-signaturemethod": "hmac-sha1",
  "x-log-bodyrawsize": 0,
  "User-Agent": "log-python-sdk-v-0.6.0",
  "Host": "ali-test-project.cn-hangzhou-devcommon-intranet.sls.aliyuncs.com",
  "Date": "Mon, 09 Nov 2015 09:51:38 GMT",
  "x-log-apiversion": "0.6.0",
  "Authorization": "LOG yourAccessKeyId:yourSignature"
}
```

- 返回示例

```
HTTP/1.1 200 OK
Header :
{
  "content-length": "44",
  "server": "nginx/1.6.1",
  "connection": "close",
  "date": "Mon, 09 Nov 2015 09:51:38 GMT",
  "content-type": "application/json",
  "x-log-requestid": "56406CAA99248CAA230BE828"
}
Body:
{
  "count": 1,
  "machinegroups":
  [
    "sample-group1",
    "sample-group2"
  ]
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	The Project does not exist : *projectName*	Project不存在。
404	ConfigNotExist	Config *confiName* does not exist.	Logtail配置不存在。
500	InternalServerError	internal server error.	内部服务调用错误。

更多错误码，请参见通用错误码。

12.4. GetConfig

调用GetConfig接口获取Logtail配置的详细信息。

请求语法

```
GET /configs/configName HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignature
Date: GMT Date
Host: ProjectName.Endpoint
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  GetConfig接口无特有请求头。关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	ali-test-project	Project名称。
configName	String	是	logtail-config-sample	Logtail配置名称。

返回数据

- 响应头

  GetConfig接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  响应内容中包含的参数信息和各种模式的Logtail配置样例，请参见Logtail配置。

示例

- 请求示例

```
GET /configs/logtail-config-sample
Header :
{
  "Content-Length": 0,
  "x-log-signaturemethod": "hmac-sha1",
  "x-log-bodyrawsize": 0,
  "User-Agent": "log-python-sdk-v-0.6.0",
  "Host": "ali-test-project.cn-hangzhou-devcommon-intranet.sls.aliyuncs.com",
  "Date": "Mon, 09 Nov 2015 08:29:15 GMT",
  "x-log-apiversion": "0.6.0",
  "Authorization": "LOG yourAccessKeyId:yourSignature"
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header :
{
  "content-length": "730",
  "server": "nginx/1.6.1",
  "connection": "close",
  "date": "Mon, 09 Nov 2015 08:29:15 GMT",
  "content-type": "application/json",
  "x-log-requestid": "5640595B99248CAA23004A59"
}
Body :
{
  "configName": "logtail-config-sample",
  "outputDetail": {
    "endpoint": "http://cn-hangzhou-devcommon-intranet.sls.aliyuncs.com",
    "logstoreName": "sls-test-logstore"
  },
  "outputType": "LogService",
  "inputType": "file",
  "inputDetail": {
    "regex": "([\\d\\.]+) \\S+ \\S+ \\[(\\S+) \\S+\\] \"(\\w+) ([^\"]*)\" ([\\d\\.]+) (\\d+) (\\d+) (\\d+|-) \"([^\"]*)\" \"([^\"]*)\".*",
    "filterKey": [],
    "logPath": "/var/log/httpd/",
    "logBeginRegex": "\\d+\\.\\d+\\.\\d+\\.\\d+ - .*",
    "logType": "common_reg_log",
    "topicFormat": "none",
    "localStorage": true,
    "key": [
      "ip",
      "time",
      "method",
      "url",
      "request_time",
      "request_length",
      "status",
      "length",
      "ref_url",
      "browser"
    ],
    "filePattern": "access*.log",
    "timeFormat": "%d/%b/%Y:%H:%M:%S",
    "filterRegex": []
  },
  "createTime": 1447040456,
  "lastModifyTime": 1447050456
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	Project *ProjectName* does not exist.	Project不存在。
404	ConfigNotExist	Config *configName* does not exist.	Logtail配置不存在。
500	InternalServerError	Specified Server Error Message.	内部服务调用错误。

更多错误码，请参见通用错误码。

12.5. DeleteConfig

调用DeleteConfig接口删除指定的Logtail配置。

接口说明

如果Logtail配置已经被应用到机器组，删除该配置，将无法采集到机器组数据。

请求语法

```
DELETE /configs/configName HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignature
Date: GMT Date
Host: ProjectName.Endpoint
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  DeleteConfig接口无特有请求头。关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必须	示例值	描述
projectName	String	是	ali-test-project	Project名称。
ConfigName	String	是	logtail-config-sample	Logtail配置名称。

返回数据

- 响应头

  DeleteConfig接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功。该接口调用成功后无任何响应元素。

示例

- 请求示例

```
DELETE /configs/logtail-config-sample
Header :
{
  "Content-Length": 0,
  "x-log-signaturemethod": "hmac-sha1",
  "x-log-bodyrawsize": 0,
  "User-Agent": "log-python-sdk-v-0.6.0",
  "Host": "ali-test-project.cn-hangzhou-devcommon-intranet.sls.aliyuncs.com",
  "Date": "Mon, 09 Nov 2015 09:28:21 GMT",
  "x-log-apiversion": "0.6.0",
  "Authorization": "LOG yourAccessKeyId:yourSignature"
}
```

- 响应示例

```
HTTP/1.1 200 OK
Header:
{
  "date": "Mon, 09 Nov 2015 09:28:21 GMT",
  "connection": "close",
  "x-log-requestid": "5640673599248CAA230836C6",
  "content-length": "0",
  "server": "nginx/1.6.1"
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ConfigNotExist	config *configname* does not exist.	Logtail配置不存在。
400	InvalidParameter	invalid config resource json.	无效Logtail配置参数。
500	InternalServerError	internal server error.	内部服务调用错误。

更多错误码，请参见通用错误码。

12.6. UpdateConfig

调用UpdateConfig接口更新Logtail配置。

接口说明

更新已经被应用到机器组的Logtail配置，则对应机器组的Logtail配置会立即生效。

请求语法

```
PUT /configs/configName HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignature
Content-Type:application/json
Date: GMT Date
Host: Projectname.Endpoint
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
{
  "configName": "testcategory1",
  "inputType": "file",
  "inputDetail": {
    "logType": "common_reg_log",
    "logPath": "/var/log/httpd/",
    "filePattern": "access.log",
    "localStorage": true,
    "timeFormat": "%Y/%m/%d %H:%M:%S",
    "logBeginRegex": ".*",
    "regex": "(\w+)(\s+)",
    "key" :["key1", "key2"],
    "filterKey":["key1"],
    "filterRegex":["regex1"],
    "topicFormat": "none"
  },
  "outputType": "LogService",
  "outputDetail":
  {
    "logstoreName": "perfcounter"
  }
}
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  UpdateConfig接口无特有请求头。关于Log Service API的公共请求头，请参见公共请求头。

- 请求参数

  请求参数和各种模式的Logtail配置样例，请参见Logtail配置。

返回数据

- 响应头

  UpdateConfig接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功。该接口调用成功后无任何响应元素。

示例

- 请求示例

```
PUT /configs/logtail-config-sample
Header :
{
  "Content-Length": 737,
  "Host": "ali-test-project.cn-hangzhou-devcommon-intranet.sls.aliyuncs.com",
  "x-log-bodyrawsize": 737,
  "Content-MD5": "431263EB105D584A5555762A81E869C0",
  "x-log-signaturemethod": "hmac-sha1",
  "Date": "Mon, 09 Nov 2015 09:14:32 GMT",
  "x-log-apiversion": "0.6.0",
  "User-Agent": "log-python-sdk-v-0.6.0",
  "Content-Type": "application/json",
  "Authorization": "LOG yourAccessKeyId:yourSignature"
}
Body :
{
  "outputDetail": {
    "logstoreName": "sls-test-logstore"
  },
  "inputType": "file",
  "inputDetail": {
    "regex": "([\\d\\.]+) \\S+ \\S+ \\[(\\S+) \\S+\\] \"(\\w+) ([^\"]*)\" ([\\d\\.]+) (\\d+) (\\d+) (\\d+|-) \"([^\"]*)\" \"([^\"]*)\".*",
    "filterKey": [],
    "logPath": "/var/log/nginx/",
    "logBeginRegex": "\\d+\\.\\d+\\.\\d+\\.\\d+ - .*",
    "logType": "common_reg_log",
    "topicFormat": "none",
    "localStorage": true,
    "key": [
      "ip",
      "time",
      "method",
      "url",
      "request_time",
      "request_length",
      "status",
      "length",
      "ref_url",
      "browser"
    ],
    "filePattern": "access*.log",
    "timeFormat": "%d/%b/%Y:%H:%M:%S",
    "filterRegex": []
  },
  "outputType": "LogService",
  "configName": "logtail-config-sample"
}
```

- 响应示例

```
HTTP/1.1 200 OK
Header :
{
  "date": "Mon, 09 Nov 2015 09:14:32 GMT",
  "connection": "close",
  "x-log-requestid": "564063F899248CAA2300B778",
  "content-length": "0",
  "server": "nginx/1.6.1"
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ConfigNotExist	config *configName* does not exist.	Logtail配置不存在。
400	InvalidParameter	invalid config resource json.	无效Logtail配置参数。
400	BadRequest	config resource configname does not match with request.	Logtail配置名称不符合要求。
500	InternalServerError	internal server error.	内部服务调用错误。

更多错误码，请参见通用错误码。

13.消费组接口

13.1. CreateConsumerGroup

调用CreateConsumerGroup接口在指定的Logstore上创建一个消费组。

接口说明

每个Logstore最多创建30个消费组。

请求语法

```
POST /logstores/logstoreName/consumergroups HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignature
x-log-bodyrawsize: 0
User-Agent: UserAgent
x-log-apiversion: 0.6.0
Host: ProjectName.Endpoint
x-log-signaturemethod: hmac-sha1
Date: GMT Date
Content-Type: application/json
Content-MD5: F58544E4D022CC28A93D0B7CC208A5AA
Content-Length: 54
{
 "consumerGroup": consumerGroup,
 "timeout": timeout,
 "order": order
}
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  CreateConsumerGroup接口无特有请求头，关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	my-project	Project名称。
logstoreName	String	是	my-logstore	Logstore名称。
consumerGroup	String	是	consumer-group-1	消费组名称，在Project下必须唯一。
timeout	Integer	是	300	超时时间。在超时时间段内没有收到心跳，消费者将被删除。单位：秒。

参数名称	数据类型	是否必填	示例值	描述
order	Boolean	否	true	在单个Shard中是否按顺序消费。 ○ true：表示在单个Shard中按顺序消费。Shard分裂后，先消费原Shard数据，然后并列消费两个新Shard的数据。 ○ false：表示不按顺序消费。

返回数据

- 响应头

  CreateConsumerGroup接口无特有响应头，关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功。该接口调用成功后无任何响应元素。

示例

- 请求示例

```
POST /logstores/my-logstore/consumergroups HTTP/1.1
Header :
{
Authorization: LOG yourAccessKeyId:yourSignature
x-log-bodyrawsize: 0
User-Agent: sls-java-sdk-v-0.6.1
x-log-apiversion: 0.6.0
Host: my-project.cn-shanghai.log.aliyuncs.com
x-log-signaturemethod: hmac-sha1
Date: Fri, 04 May 2018 08:02:22 GMT
Content-Type: application/json
Content-MD5: F58544E4D022CC28A93D0B7CC208A5AA
Content-Length: 65
Connection: Keep-Alive
}
Body :
{
 "consumerGroup": "consumer-group-1",
 "timeout": 300,
 "order": true
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header :
{
Server: nginx/1.12.1
Content-Length: 0
Connection: close
Access-Control-Allow-Origin: *
Date: Fri, 04 May 2018 08:15:11 GMT
x-log-requestid: 5AEC168FA796F4195BF404CB
}
```

错误码

HTTP状态码	错误码	错误信息	描述
400	ConsumerGroupAlreadyExist	consumer group already exist.	消费组已存在。
400	JsonInfoInvalid	consumerGroup or timeout is of error format.	参数consumerGroup或timeout格式错误。
404	ProjectNotExist	The Project does not exist : *ProjectName*.	Project不存在。
404	LogStoreNotExist	logstore *logstoreName* dose not exist.	Logstore不存在。
500	InternalServerError	Specified Server Error Message.	内部服务调用错误。

更多错误码，请参见通用错误码。

13.2. DeleteConsumerGroup

调用DeleteConsumerGroup接口删除一个指定的消费组。

请求语法

```
DELETE /logstores/logstoreName/consumergroups/consumerGroupName HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignature
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
Host: ProjectName.Endpoint
x-log-signaturemethod: hmac-sha1
Date: GMT Date
Content-Type: application/x-protobuf
Content-MD5: F58544E4D022CC28A93D0B7CC208A5AA
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  DeleteConsumerGroup接口无特有请求头，关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	ali-test-project	Project名称。
logstoreName	String	是	logstore-test	Logstore名称。
consumerGroup	String	是	consumer-group-1	消费组名称。

返回数据

- 响应头

DeleteConsumerGroup接口无特有响应头，关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功。该接口调用成功后无任何响应元素。

示例

- 请求示例

```
DELETE /logstores/logstore-test/consumergroups/consumer-group-1 HTTP/1.1
Header:
{
Authorization: LOG yourAccessKeyId:yourSignature
x-log-bodyrawsize: 0
User-Agent: sls-java-sdk-v-0.6.1
x-log-apiversion: 0.6.0
Host: ali-test-project.cn-shanghai.log.aliyuncs.com
x-log-signaturemethod: hmac-sha1
Date: Fri, 04 May 2018 08:02:22 GMT
Content-Type: application/json
Content-MD5: F58544E4D022CC28A93D0B7CC208A5AA
Content-Length: 65
Connection: Keep-Alive
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header :
{
Server: nginx/1.12.1
Content-Length: 0
Connection: close
Access-Control-Allow-Origin: *
Date: Fri, 04 May 2018 08:15:11 GMT
x-log-requestid: 5AEC168FA796F4195BF404CB
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	The Project does not exist : *projectName*.	Project不存在。
404	LogStoreNotExist	The logstore *logstoreName* dose not exist.	Logstore不存在。
500	InternalServerError	Specified Server Error Message.	内部服务调用错误。

更多错误码，请参见通用错误码。

13.3. GetCheckPoint

调用GetCheckPoint接口获取指定消费组消费数据时Shard的checkpoint。

请求语法

```
GET /logstores/logstoreName/consumergroups/consumerGroup?shard=shardId HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignature
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
Host: ProjectName.Endpoint
x-log-signaturemethod: hmac-sha1
Date: GMT Date
Content-Type: application/json
Connection: Keep-Alive
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  GetCheckPoint接口无特有请求头。关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	my-project	Project名称。
logstoreName	String	是	my-logstore	Logstore名称。
consumerGroup	String	是	consumer_group_test	消费组名称。
shardId	Integer	否	0	Shard ID。 ○ 如果指定的Shard不存在，则返回空列表。 ○ 如果不指定Shard，则返回所有Shard的checkpoint。

返回数据

- 响应头

  GetCheckPoint接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功，其响应Body会包含指定消费组消费数据时Shard的checkpoint，具体如下：

参数名称	数据类型	示例值	描述
shard	Integer	0	Shard ID。
checkpoint	String	MTUyNDE1NTM3OTM3MzkwODQ5Ng==	checkpoint值。
updateTime	Integer	1524224984800922	checkpoint最后的更新时间。Unix时间戳格式，表示从1970-1-1 00:00:00 UTC计算起的秒数。
consumer	String	consumer_1	消费者。

示例

- 请求示例

```
GET /logstores/my-logstore/consumergroups/consumer_group_test?shard=0
Header：
{
Authorization: LOG yourAccessKeyId:yourSignature
x-log-bodyrawsize: 0
User-Agent: sls-java-sdk-v-0.6.1
x-log-apiversion: 0.6.0
Host: my-project.cn-shanghai.log.aliyuncs.com
x-log-signaturemethod: hmac-sha1
Date: Fri, 04 May 2018 09:26:53 GMT
Content-Type: application/x-protobuf
Connection: Keep-Alive
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header：
{
Server: nginx/1.12.1
Content-Type: application/json
Content-Length: 111
Connection: close
Access-Control-Allow-Origin: *
Date: Fri, 04 May 2018 09:26:53 GMT
x-log-requestid: 5AEC275D1FFC036AB254B20C
}
Body：
{
[
 {
  "shard": 0,
  "checkpoint": "MTUyNDE1NTM3OTM3MzkwODQ5Ng==",
  "updateTime": 1524224984800922,
  "consumer": "consumer_1"
 }
]
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	The Project does not exist：*projectName*.	Project不存在。
404	LogStoreNotExist	logstore *logstoreName* dose not exist.	Logstore不存在。
404	ConsumerGroupNotExist	consumer group not exist.	消费组不存在。
500	InternalServerError	Specified Server Error Message.	内部服务调用错误。

更多错误码，请参见通用错误码。

13.4. HeartBeat

调用HeartBeat接口为指定消费者发送心跳到服务端。

接口说明

只有消费者和服务端建立连接之后才能发送心跳。

请求语法

```
POST /logstores/logstoreName/consumergroups/consumerGroup?type=heartbeat&consumer=<consumer> HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignature
x-log-bodyrawsize: 0
User-Agent: UserAgent
x-log-apiversion: 0.6.0
Host: ProjectName.Endpoint
x-log-signaturemethod: hmac-sha1
Date: GMT Date
Content-Type: application/json
Content-MD5: F58544E4D022CC28A93D0B7CC208A5AA
Content-Length: 54
Body :
{ "shards" :shard ID list}
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  HeartBeat接口无特有请求头，关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	my-project	Project名称。
logstoreName	String	是	my-logstore	日志库名称，同一Project中日志库名称为唯一值。
consumerGroup	String	是	consumer_group_test	消费组名称，同一Project中消费组名称为唯一值。
consumer	String	是	consumer_1	消费者。
shards	Array	否	[0]	正在消费的Shard ID列表。

返回数据

- 响应头

  HeartBeat接口无特有响应头，关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，则表示HeartBeat请求成功，同时返回消费者消费的所有Shard ID列表。

示例

- 请求示例

```
POST /logstores/my-logstore/consumergroups/consumer_group_test?type=heartbeat&consumer=consumer_1 HTTP/1.1
Header :
{
Authorization: LOG yourAccessKeyId:yourSignature
x-log-bodyrawsize: 0
User-Agent: sls-java-sdk-v-0.6.1
x-log-apiversion: 0.6.0
Host: my-project.cn-shanghai.log.aliyuncs.com
x-log-signaturemethod: hmac-sha1
Date: Sun, 06 May 2018 09:44:10 GMT
Content-Type: application/json
Content-MD5: 8D5162CA104FA7E79FE80FD92BB657FB
Content-Length: 3
Connection: Keep-Alive
}
Body :
{ "shards" : [0]}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header :
{
Server: nginx/1.12.1
Content-Type: application/json
Content-Length: 5
Connection: close
Access-Control-Allow-Origin: *
Date: Sun, 06 May 2018 09:44:11 GMT
x-log-requestid: 5AEECE6B1FFC0366B2553FB5
}
Body :
{ "shards" : [0,1]}
```

错误码

HTTP状态码	错误码	错误信息	描述
400	NotExistConsumerWithBody	non-exist consumer with non-empty body of heartbeat message.	不存在非空心跳消息的消费者。
404	ProjectNotExist	The Project does not exist : *ProjectName*.	Project不存在。
404	LogStoreNotExist	logstore *logstoreName* dose not exist.	Logstore不存在。
404	ConsumerGroupNotExist	consumer group not exist.	消费组不存在。
500	InternalServerError	Specified Server Error Message.	内部服务调用错误。

更多错误码，请参见通用错误码。

13.5. ListConsumerGroup

调用ListConsumerGroup接口查询指定Logstore的所有消费组。

请求语法

```
GET /logstores/logstoreName/consumergroups HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignature
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
Host: ProjectName.Endpoint
x-log-signaturemethod: hmac-sha1
Date: GMT Date
Content-Type: application/x-protobuf
Connection: Keep-Alive
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  ListConsumerGroup接口无特有请求头，关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	my-project	Project名称。
logstoreName	String	是	my-logstore	Logstore名称。

返回数据

- 响应头

  ListConsumerGroup接口无特有响应头，关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功。其响应Body中包含目标Project下的消费组信息，具体如下：

参数名称	数据类型	示例值	描述
count	Integer	1	消费组数量。
consumerGroups	Array	[{''name'': ''test-consumer-group'', ''timeout'': 30, ''order'': False}]	消费组列表。更多信息，请参见下表consumerGroups参数说明。

consumerGroups参数说明如下表所示：

参数名称	数据类型	示例值	描述
name	String	test-consumer-group	消费组名称。
timeout	Integer	30	超时时间。
order	Boolean	true	是否按顺序消费。

示例

- 请求示例

```
GET /logstores/my-logstore/consumergroups HTTP/1.1
Header :
{
Authorization: LOG yourAccessKeyId:yourSignature
x-log-bodyrawsize: 0
User-Agent: sls-java-sdk-v-0.6.1
x-log-apiversion: 0.6.0
Host: my-project.cn-shanghai.log.aliyuncs.com
x-log-signaturemethod: hmac-sha1
Date: Fri, 04 May 2018 08:47:30 GMT
Content-Type: application/x-protobuf
Connection: Keep-Alive
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header :
{
Server: nginx/1.12.1
Content-Type: application/json
Connection: close
Access-Control-Allow-Origin: *
Date: Fri, 04 May 2018 08:47:31 GMT
x-log-requestid: 5AEC1E23048191954B42EAB9
}
Body :
{
  "count": "1",
  "consumerGroups":
[
 {
  "name": "test-consumer-group",
  "timeout": 30,
  "order": False
 }
]
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	The Project does not exist : *projectName*.	Project不存在。
404	LogStoreNotExist	logstore *logstoreName* dose not exist.	Logstore不存在。
500	InternalServerError	Specified Server Error Message.	内部服务调用错误。

更多错误码，请参见通用错误码。

13.6. UpdateCheckPoint

调用UpdateCheckPoint接口更新指定消费组消费数据时Shard的checkpoint。

接口说明

当不指定消费者时，必须指定forceSuccess为true才能更新checkpoint。

请求语法

```
POST /logstores/logstoreName/consumergroups/consumerGroup HTTP/1.1
Authorization: AuthorizationString
x-log-bodyrawsize: 0
User-Agent: UserAgent
x-log-apiversion: 0.6.0
Host: ProjectName.Endpoint
x-log-signaturemethod: hmac-sha1
Date: GMT Date
Content-Type: application/json
Content-Length: 54
Connection: Keep-Alive
{
 "shard": shardID,
 "checkpoint": checkpoint
}
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  UpdateCheckPoint接口无特有请求头，关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	my-project	Project名称。
logstoreName	String	是	logstore-4	Logstore名称。
consumerGroup	String	是	consumer_group_test	消费组名称。
shard	Integer	是	0	Shard ID。
checkpoint	String	是	MTUyNDE1NTM3OTM3MzkwODQ5Ng==	checkpoint值。
consumer	String	否	consumer_1	消费者。
forceSuccess	Boolean	否	False	是否强制更新。 ○ True：强制更新 ○ False：不强制更新

返回数据

- 响应头

  UpdateCheckPoint接口无特有响应头，关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功。该接口调用成功后无任何响应元素。

示例

- 请求示例

```
POST /logstores/logstore-4/consumergroups/consumer_group_test?forceSuccess=false&type=checkpoint&consumer=consumer_1 HTTP/1.1
Header：
{
Authorization: LOG yourAccessKeyId:yourSignature
x-log-bodyrawsize: 0
User-Agent: sls-java-sdk-v-0.6.1
x-log-apiversion: 0.6.0
Host: my-project.cn-shanghai.log.aliyuncs.com
x-log-signaturemethod: hmac-sha1
Date: Sun, 06 May 2018 09:14:53 GMT
Content-Type: application/json
Content-MD5: 59457BAFB3F85A5117BDE243947583B0
Content-Length: 55
Connection: Keep-Alive
}
Body：
{
 "shard": 0,
 "checkpoint": "MTUyNDE1NTM3OTM3MzkwODQ5Ng=="
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header：
{
Server: nginx/1.12.1
Content-Length: 0
Connection: close
Access-Control-Allow-Origin: *
Date: Sun, 06 May 2018 09:14:54 GMT
x-log-requestid: 5AEEC78E1FFC0366B24F1C63
}
```

错误码

HTTP状态码	错误码	错误信息	描述
400	InvalidShardCheckPoint	shard checkpoint not encoded by base64.	checkpoint不是Base64编码，格式错误。
404	ProjectNotExist	The Project does not exist : *ProjectName*.	Project不存在。
404	LogStoreNotExist	logstore *logstoreName* dose not exist.	Logstore不存在。
404	ConsumerGroupNotExist	consumer group not exist.	消费组不存在。
404	ConsumerNotExist	consumer not exist in the consumer group	消费组中不存在该消费者。
404	ShardNotExist	shard not exist.	Shard不存在。
500	InternalServerError	Specified Server Error Message.	内部服务调用错误。

更多错误码，请参见通用错误码。

13.7. UpdateConsumerGroup

调用UpdateConsumerGroup接口修改指定消费组属性。

请求语法

```
PUT /logstores/logstoreName/consumergroups/consumerGroup HTTP/1.1
Authorization: LOG yourAccessKeyId:yourSignature
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
Host: ProjectName.Endpoint
x-log-signaturemethod: hmac-sha1
Date: GMT Date
Content-Type: application/json
Content-MD5: F58544E4D022CC28A93D0B7CC208A5AA
Content-Length: 20
{
 "timeout": timeout,
 "order": order
}
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  UpdateConsumerGroup接口无特有请求头，关于Log Service API的公共请求头，请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	my-project	Project名称。
logstoreName	String	是	logstore-test	Logstore名称。
consumerGroup	String	是	consumer-group-1	消费组名称。
timeout	Integer	否	100	超时时间。在超时时间段内没有收到心跳，消费组将被删除。单位：秒。 **说明** 该参数和order参数需要至少指定一个。

参数名称	数据类型	是否必填	示例值	描述
order	Boolean	否	False	在单个Shard中是否按顺序消费。 ○ True：表示在单个Shard中按顺序消费。Shard分裂后，先消费原Shard数据，然后并列消费两个新Shard的数据。 ○ False：表示不按顺序消费。 **说明** 该参数和timeout参数需要至少指定一个。

返回数据

- 响应头

  UpdateConsumerGroup接口无特有响应头，关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功。该接口调用成功后无任何响应元素。

示例

- 请求示例

```
PUT /logstores/logstore-test/consumergroups/consumer-group-1 HTTP/1.1
Header:
Authorization: LOG yourAccessKeyId:yourSignature
x-log-bodyrawsize: 0
User-Agent: sls-java-sdk-v-0.6.1
x-log-apiversion: 0.6.0
Host: my-project.cn-shanghai.log.aliyuncs.com
x-log-signaturemethod: hmac-sha1
Date: Fri, 04 May 2018 08:02:22 GMT
Content-Type: application/json
Content-MD5: F58544E4D022CC28A93D0B7CC208A5AA
Content-Length: 65
Connection: Keep-Alive
Body:
{
 "order": False,
 "timeout": 100
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header :
{
Server: nginx/1.12.1
Content-Length: 0
Connection: close
Access-Control-Allow-Origin: *
Date: Fri, 04 May 2018 08:15:11 GMT
x-log-requestid: 5AEC168FA796F4195BF404CB
}
```

错误码

HTTP状态码	错误码	错误信息	描述
400	JsonInfoInvalid	timeout is of error value type.	参数timeout类型错误。
404	ProjectNotExist	The Project does not exist : *projectName*.	Project不存在。
404	ConsumerGroupNotExist	consumer group not exist.	消费组不存在。
404	LogStoreNotExist	logstore *logstoreName* dose not exist.	Logstore不存在。
500	InternalServerError	Specified Server Error Message.	内部服务调用错误。

更多错误码，请参见通用错误码。

14.外部存储

14.1. 概述

在进行日志数据查询分析时，日志中的部分信息需要和外部存储（ExternalStore）的信息进行关联查询，从而进一步丰富日志服务中的日志数据。

外部存储简介

- 外部存储和Logstore的区别

项目	外部存储（ExternalStore）	Logstore
数据存储	日志保存在ExternalStore中，与日志相关联的、静态的meta信息。	日志保存在Logstore中，日志数据源源不断且持续更新。

- 使用场景
  - Logstore和ExternalStore进行关联查询分析。
  - 将Logstore的查询分析结果写入到外部存储中。
- 存储类型

  目前仅支持RDS VPC类型的外部数据源。

  > **说明** RDS仅支持北京、青岛、杭州三个Region，若需要其他Region的RDS作为外部数据源，请提工单申请。

使用步骤

1. 创建外部数据源，指定数据源的meta，请参见CreateExternalStore，UpdateExternalStore，ListExternalStore，GetExternalStore，DeleteExternalStore或Log Service Python SDK外部数据源。
2. 在查询分析中，引用外部数据源的名称，进行查询和写入。

14.2. CreateExternalStore

调用CreateExternalStore接口创建外部存储数据。

接口说明

目前支持OSS数据源和VPC下的RDS MySQL数据库作为外部存储数据。本文以RDS MySQL为例，OSS数据源作为外部存储数据时其参数配置请参见关联OSS数据源。

请求语法

```
POST /externalstores HTTP/1.1
x-log-bodyrawsize: 0
Content-Type: application/json
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
Host: ProjectName.Endpoint
Date: GMT Date
Authorization: LOG yourAccessKeyId:yourSignature
{
 "externalStoreName": "externalStoreName",
 "storeType": "rds-vpc",
 "parameter": {
  "vpc-id": "vpc-id",
  "instance-id": "instance-id",
  "host": "host",
  "port": "port",
  "username": "username",
  "password": "password",
  "db": "db",
  "table": "table",
  "region": "region"
 }
}
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  CreateExternalStore接口无特有请求头，关于Log Service API的公共请求头请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	ali-test-project	Project名称。
externalStoreName	String	是	rds_store	外部存储名称，在同一Project中名称不能重复，且和Logstore名称不能重复。
storeType	String	是	rds-vpc	存储类型。固定取值为rds-vpc，表示VPC下的RDS MySQL数据库。
vpc-id	String	否	vpc-bp1aevy8sofi8mh1q****	RDS MySQL实例所属的VPC ID。
instance-id	String	否	i-bp1b6c719dfa08exf****	RDS MySQL实例ID。
host	String	否	192.168.XX.XX	RDS MySQL实例的内网地址或外网地址。
port	String	是	3306	RDS MySQL实例的内网或者外网端口。
username	String	是	root	RDS MySQL实例中的账号名称。

参数名称	数据类型	是否必填	示例值	描述
password	String	是	sfdsfldsfksfls****	RDS MySQL实例中账号对应的密码。
db	String	是	meta	RDS MySQL实例的数据库名称。
table	String	是	join_meta	RDS MySQL实例的数据库表名称。
region	String	是	cn-qingdao	RDS MySQL实例所在地域，目前仅支持cn-qingdao、cn-beijing、cn-hangzhou。

返回数据

- 响应头

  CreateExternalStore接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功。该接口调用成功后无任何响应元素。

示例

- 请求示例

```
POST /externalstores HTTP/1.1
Header：
{
x-log-bodyrawsize: 0
Content-Type: application/json
Content-Length: 307
Content-MD5: 7C1D14659C0BBBA7C7BFF9E5A1A46705
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
Host: ali-test-project.cn-chengdu.log.aliyuncs.com
Date: Thu, 19 Apr 2018 02:15:41 GMT
Authorization: LOG yourAccessKeyId:yourSignature
}
Body：
{
 "externalStoreName": "rds_store",
 "storeType": "rds-vpc",
 "parameter": {
 "vpc-id": "vpc-bp1aevy8sofi8mh1q****",
 "instance-id": "i-bp1b6c719dfa08exf****",
 "host": "192.168.XX.XX",
 "port": "3306",
 "username": "root",
 "password": "sfdsfldsfksfls****",
 "db": "meta",
 "table": "join_meta",
 "region": "cn-qingdao"
 }
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header
{
date: Mon, 09 Nov 2015 07:45:30 GMT
connection: close
x-log-requestid: 56404F1A99248CA26C002180
content-length: 0
server: nginx/1.6.1
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	The Project does not exist : *projectName*	Project不存在。
400	ParameterInvalid	The body is not valid json string.	外部存储配置中存在无效的参数。
500	InternalServerError	Specified Server Error Message.	内部服务调用错误。

更多错误码，请参见通用错误码。

14.3. UpdateExternalStore

调用UpdateExternalStore接口修改外部存储数据的配置信息。

接口说明

目前支持OSS数据源和VPC下的RDS MySQL数据库作为外部存储数据。本文以RDS MySQL为例，OSS数据源作为外部存储数据时其参数配置请参见关联OSS数据源。

请求语法

```
PUT /externalstores/externalStoreName HTTP/1.1
x-log-bodyrawsize: 0
Content-Type: application/json
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
Host: ProjectName.Endpoint
Date: GMT Date
Authorization: LOG yourAccessKeyId:yourSignature
{"externalStoreName": "externalStoreName",
 "storeType": "rds-vpc",
 "parameter": {
        "vpc-id": "vpc-id",
        "instance-id": "instance-id",
        "host": "host",
        "port": "port",
        "username": "username",
        "password": "password",
        "db": "db",
        "table": "table",
        "region": "region"
        }
}
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  UpdateExternalStore接口无特有请求头，关于Log Service API的公共请求头请参见公共请求头。

- 请求参数

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	ali-test-project	Project名称。
externalStoreName	String	是	rds_store	外部存储名称。
storeType	String	是	rds-vpc	存储类型。固定取值为rds-vpc，表示VPC下的RDS MySQL数据库。
vpc-id	String	否	vpc-bp1aevy8sofi8mh1q****	RDS MySQL实例所属的VPC ID。
intance-id	String	否	i-bp1b6c719dfa08exf****	RDS MySQL实例ID。
host	String	否	192.168.XX.XX	RDS MySQL实例的内网地址或外网地址。
port	String	是	3306	RDS MySQL实例的内网或者外网端口。
username	String	是	root	RDS MySQL实例中的账号名称。
password	String	是	sfdsfldsfksfl****	RDS MySQL实例中账号对应的密码。
db	String	是	meta	RDS MySQL实例的数据库名称。
table	String	是	join_meta	RDS MySQL实例的数据库表名称。
region	String	是	cn-qingdao	RDS MySQL实例所在地域，目前仅支持cn-qingdao、cn-beijing、cn-hangzhou。

返回数据

- 响应头

  UpdateExternalStore接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功。该接口调用成功后无任何响应元素。

示例

- 请求示例

```
PUT /externalstores/rds_store  HTTP/1.1
Header :
{
x-log-bodyrawsize: 0
Content-Type: application/json
Content-Length: 307
Content-MD5: 7C1D14659C0BBBA7C7BFF9E5A1A46705
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
Host: ali-test-project.cn-chengdu.log.aliyuncs.com
Date: Thu, 19 Apr 2018 02:15:41 GMT
Authorization: LOG yourAccessKeyId:yourSignature
}
Body :
{"externalStoreName": "rds_store",
 "storeType": "rds-vpc",
 "parameter": {
        "vpc-id": "vpc-p1aevy8sofi8mh1q****",
        "instance-id": "i-bp1b6c719dfa08exf****",
        "host": "192.168.XX.XX",
        "port": "3306",
        "username": "root",
        "password": "sfdsfldsfksfl****",
        "db": "meta",
        "table": "join_meta",
        "region": "cn-qingdao"
        }
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header
{
date: Mon, 09 Nov 2015 07:45:30 GMT
connection: close
x-log-requestid: 56404F1A99248CA26C002180
content-length: 0
server: nginx/1.6.1
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	The Project does not exist : *projectName*	Project不存在。
400	ParameterInvalid	The body is not valid json string.	外部存储配置中存在无效的参数。
500	InternalServerError	Specified Server Error Message.	内部服务调用错误。

更多错误码，请参见通用错误码。

14.4. ListExternalStore

调用ListExternalStore接口查询外部存储数据列表。支持查询指定Project下所有外部存储列表，也支持查询指定Project下具体外部存储信息。

请求语法

```
GET /externalstores?externalStoreName=externalStoreName&offset=offset&sizes=sizes
Content-Length: 0
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
Host: ProjectName.Endpoint
Date: GMT Date
Authorization: LOG yourAccessKeyId:yourSignature
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  ListExternalStore接口无特有请求头，关于Log Service API的公共请求头请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	ali-test-project	Project名称。
externalStoreName	String	否	store	外部存储名称。支持查询出包含该字符串的外部存储。
offset	Integer	否	0	查询开始行。默认值为0。
sizes	Integer	否	10	分页查询时，设置的每页行数。最大值为500。

返回数据

- 响应头

  ListExternalStore接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，表示请求成功。响应Body中包括外部存储名称，具体格式如下：

参数名称	数据类型	示例值	描述
count	Integer	3	当前页返回的外部存储数量。
total	Integer	3	符合查询条件的外部存储总数。
externalstores	Array	["ecs_store", "rds_store", "ecs_store1"]	返回的外部存储名称列表。

示例

- 请求示例

```
GET http://ali-test-project.cn-chengdu.log.aliyuncs.com:80/externalstores?externalStoreName=store&offset=0&sizes
=10
Header :
{
Content-Length: 0
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
x-log-signaturemethod':hmac-sha1
Host: ali-test-project.cn-chengdu.log.aliyuncs.com
Date: Thu, 19 Apr 2018 03:03:16 GMT
Authorization: LOG yourAccessKeyId:yourSignature
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header :
{
content-length: 103
server: nginx/1.6.1
connection: close
date: Mon, 09 Nov 2015 09:19:13 GMT
content-type: application/json
x-log-requestid: 5640651199248CAA2300C2BA
}
Body:
{
  "count": 3,
  "total": 3,
  "externalstores":
  [
    "ecs_store",
    "rds_store",
    "ecs_store1"
  ]
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	The Project does not exist : *projectName*	Project不存在。
500	InternalServerError	Specified Server Error Message.	内部服务调用错误。

更多错误码，请参见通用错误码。

14.5. GetExternalStore

调用GetExternalStore接口获取指定外部存储数据的详细信息。

接口说明

目前支持OSS数据源和VPC下的RDS MySQL数据库作为外部存储数据。本文以RDS MySQL为例，OSS数据源作为外部存储数据时其参数配置请参见关联OSS数据源。

请求语法

```
GET /externalstores/externalStoreName
Content-Length: 0
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
Host: ProjectName.Endpoint
Date: GMT Date
Authorization: LOG yourAccessKeyId:yourSignature
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  GetExternalStore接口无特有请求头，关于Log Service API的公共请求头请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	ali-test-project	Project名称。
externalStoreName	String	是	rds_store	外部存储名称。

返回数据

- 响应头

  GetExternalStore接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功。响应Body中包含该Project下指定外部存储详细信息，具体如下：

参数名称	数据类型	示例值	描述
storeType	String	rds-vpc	存储类型。固定取值为rds-vpc，表示VPC下的RDS MySQL数据库。
vpc-id	String	vpc-bp1aevy8sofi8mh1q****	RDS MySQL实例所属的VPC ID。
instance-id	String	i-bp1b6c719dfa08exf****	RDS MySQL的实例ID。
host	String	192.***.***.***	RDS MySQL实例的内网地址或外网地址。
port	String	3306	RDS MySQL实例的内网或者外网端口。
username	String	root	RDS MySQL实例中创建的账号名称。
db	String	meta	RDS MySQL实例的数据库名称。
table	String	join_meta	RDS MySQL实例的数据库表名称。
region	String	cn-qingdao	RDS MySQL实例所在地域，目前仅支持cn-qingdao、cn-beijing、cn-hangzhou。

示例

- 请求示例

```
GET /externalstores/rds_store
Header :
{
Content-Length: 0
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
Host: ali-test-project.cn-chengdu.log.aliyuncs.com
Date: Thu, 19 Apr 2018 03:26:49 GMT
Authorization: LOG yourAccessKeyId:yourSignature
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header :
{
content-length: 730
server: nginx/1.6.1
connection: close
date: Mon, 09 Nov 2015 08:29:15 GMT
content-type: application/json
x-log-requestid: 5640595B99248CAA23004A59
}
Body :
{
 'storeType': 'rds-vpc',
 'parameter': {
        'region': 'cn-qingdao',
        'vpc-id': 'vpc-p1aevy8sofi8mh1q****',
        'instance-id': 'i-bp1b6c719dfa08exf****',
        'host': '192.168.XX.XX',
        'port': '3306',
        'username': 'root',
        'db': 'meta',
        'table': 'join_meta'
        }
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	The Project does not exist : *projectName*	Project不存在。
404	LogStoreNotExist	Logstore *externalStoreName* does not exist	外部存储不存在。
500	InternalServerError	Specified Server Error Message.	内部服务调用错误。

更多错误码，请参见通用错误码。

14.6. DeleteExternalStore

调用DeleteExternalStore接口删除指定外部存储数据。

请求语法

```
DELETE /externalstores/externalStoreName
Content-Length: 0
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
Host: ProjectName.Endpoint
Date: GMT Date
Authorization: LOG yourAccessKeyId:yourSignature
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  DeleteExternalStore接口无特有请求头，关于Log Service API的公共请求头请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	ali-test-project	Project名称。
externalStoreName	String	是	rds_store	外部存储名称。

返回数据

- 响应头

  DeleteExternalStore接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功。该接口调用成功后无任何响应元素。

示例

- 请求示例

```
DELETE /externalstores/rds_store
Header :
{
Content-Length: 0
x-log-bodyrawsize: 0
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
Host: ali-test-project.cn-chengdu.log.aliyuncs.com
Date: Thu, 19 Apr 2018 03:32:49 GMT
Authorization: LOG yourAccessKeyId:yourSignature
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header
{
  date: Mon, 09 Nov 2015 07:45:30 GMT
  connection: close
  x-log-requestid: 56404F1A99248CA26C002180
  content-length: 0
  server: nginx/1.6.1
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	The Project does not exist : *projectName*	Project不存在。
400	ParameterInvalid	External store does not exist	外部存储不存在。
500	InternalServerError	Specified Server Error Message.	内部服务调用错误。

更多错误码，请参见通用错误码。

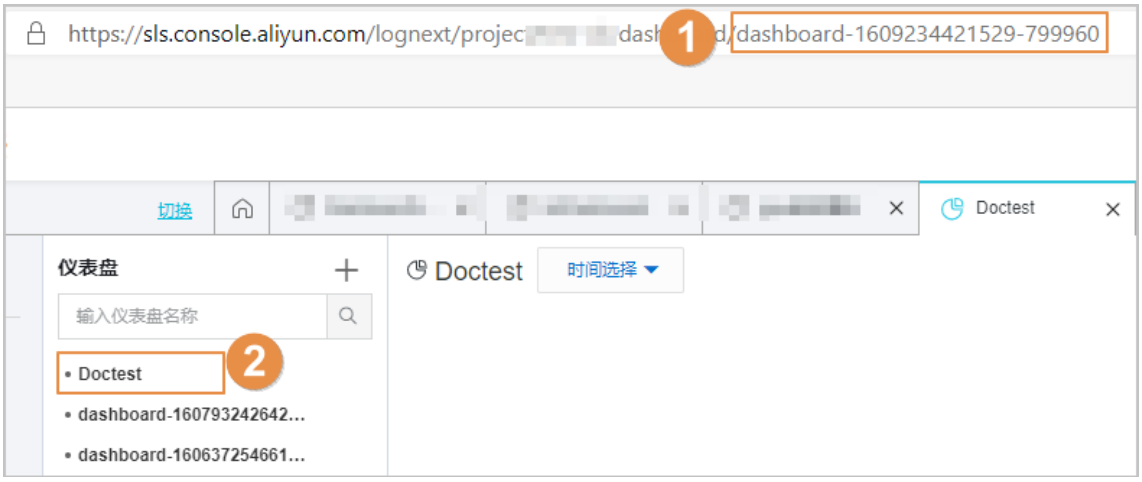
15.仪表盘接口

15.1. ListDashboard

调用ListDashboard接口查询指定Project下的仪表盘（Dashboard）简要信息，包括仪表盘ID和仪表盘名称。支持查询指定Project下所有仪表盘列表，也支持查询指定Project下具体仪表盘信息。

接口说明

仪表盘ID和名称在控制台如下所示。



序号	描述
①	仪表盘ID
②	仪表盘名称

请求语法

```
GET /dashboards HTTP/1.1
Content-Length: 0
x-log-bodyrawsize': 0
x-log-apiversion: 0.6.0
x-log-signaturemethod: hmac-sha1
Host: ProjectName.Endpoint
Date: GMT Date
Authorization: LOG yourAccessKeyId:yourSignature
x-log-date: Mon, 30 Nov 2020 06:22:17 GMT
```

其中，Host由Project名称和日志服务Endpoint构成，您需要在Host中指定Project。

请求参数

- 请求头

  ListDashboard接口无特有请求头，关于Log Service API的公共请求头请参见公共请求头。

- 参数列表

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	ali-test-project	Project名称。

参数名称	数据类型	是否必填	示例值	描述
dashboardName	String	否	dashboard-1609294922657-434834	仪表盘ID。同一个Project下，仪表盘ID唯一，不可重复。支持模糊查询，例如输入da，会查询出所有以da开头的仪表盘。
displayName	String	否	data-ingest	仪表盘显示名称。
offset	Integer	否	0	查询开始行。默认值为0。
sizes	Integer	否	10	分页查询时，设置的每页行数。最大值为500。

返回数据

- 响应头

  ListDashboard接口无特有响应头。关于Log Service API的公共响应头，请参见公共响应头。

- 响应元素

  返回HTTP状态码200，则表示请求成功。响应Body中包括仪表盘简要信息，具体格式如下：

参数名称	数据类型	示例值	描述
count	Integer	1	当前页返回的仪表盘数量。
total	Integer	1	符合查询条件的仪表盘总数。
dashboards	Array	[{"dashboardName":"dashboard-1609294922657-434834","displayName":"data-ingest"}]	返回的仪表盘简要信息列表。

示例

- 请求示例

```
GET /dashboards HTTP/1.1
Header:
{
  "Content-Length": "0",
  "x-log-bodyrawsize": "0",
  "x-log-apiversion": "0.6.0",
  "x-log-signaturemethod": "hmac-sha1",
  "Host": "ali-test-project.cn-hangzhou-devcommon-intranet.sls.aliyuncs.com",
  "Date": "Mon, 30 Nov 2020 06:22:17 GMT",
  "Authorization": "LOG yourAccessKeyId:yourSignature",
  "x-log-date": "Mon, 30 Nov 2020 06:22:17 GMT"
}
```

- 正常返回示例

```
HTTP/1.1 200 OK
Header:
{
  "content-length": "85"
  "server": "nginx/1.6.1"
  "connection": "close"
  "date": "Mon, 09 Nov 2015 09:19:13 GMT"
  "content-type": "application/json"
  "x-log-requestid": "5640651199248CAA2300C2BA"
}
Body:
{
  "count": 1,
  "total": 1,
  "dashboards": :[{"dashboardName":"dashboard-1609294922657-434834","displayName":"data-ingest"}]
}
```

错误码

HTTP状态码	错误码	错误信息	描述
404	ProjectNotExist	The Project does not exist : *projectName*	Project不存在。
400	ParameterInvalid	offset: *offset* is invalid.	offset参数取值无效。
400	ParameterInvalid	sizes: *sizes* is invalid.	sizes参数取值无效。

更多错误码，请参见通用错误码。

16.鉴权规则

16.1. 概览

本文介绍借助RAM实现RAM用户对主账号日志服务资源的访问。

背景信息

您创建的Project、Logstore、采集配置、机器组，都是您自己拥有的资源。默认情况下，您对自己的资源拥有完整的操作权限，可以使用本文档中列举的所有API对资源进行操作。

但在RAM用户场景下，刚创建的RAM用户无权限操作主账号资源，需要通过RAM授权方式，授予RAM用户操作主账号资源的权限。

> 说明 在了解如何使用RAM来授权和访问日志服务资源之前，请确保您已详细阅读了授权用户角色和RAM简介。

如果您不需要跨账户进行日志服务资源的授权和访问，您可以跳过此章节。跳过这些部分并不影响您对文档中其余部分的理解和使用。

授权策略

RAM和日志服务相关的授权策略：

- AliyunLogFullAccess

  给RAM用户授予该权限后，RAM用户将对主账号拥有的日志服务资源具备访问权限。授权策略描述如下：

```
{
 "Version": "1",
 "Statement": [
  {
   "Action": "log:*",
   "Resource": "*",
   "Effect": "Allow"
  }
 ]
}
```

- AliyunLogReadOnlyAccess

  给RAM用户授予该权限后，RAM用户将对主账号拥有的日志服务资源具备只读权限。授权策略描述如下：

```
{
 "Version": "1",
 "Statement": [
  {
   "Action": [
    "log:Get*",
    "log:List*"
   ],
   "Resource": "*",
   "Effect": "Allow"
  }
 ]
}
```

- 向指定日志库（Logstore）上传数据

  给RAM用户授予该权限后，RAM用户可以通过API、SDK直接向指定日志库上传数据，授权策略描述如下：

```
{
 "Version": "1",
 "Statement": [
  {
   "Action": [
    "log:Post*",
    "log:BatchPost*"
   ],
   "Resource": ["acs:log:*:*:project/<指定的project名称>/logstore/<指定的logstore名称>"],
   "Effect": "Allow"
  }
 ]
}
```

- 控制台查询指定日志库（Logstore）数据

  给RAM用户授予该权限后，RAM用户在控制台上拥有指定日志库资源的只读访问权限（查询日志、拖取日志、查看日志库列表）。授权策略描述如下：

```
{
 "Version": "1",
 "Statement": [
  {
   "Action": ["log:List*"],
   "Resource": ["acs:log:*:*:project/<指定的project名称>/*"],
   "Effect": "Allow"
  },
  {
   "Action": ["log:Get*"],
   "Resource": ["acs:log:*:*:project/<指定的project名称>/logstore/<指定的logstore名称>"],
   "Effect": "Allow"
  }
 ]
}
```

更多信息

- RAM中可授权的Log Service资源类型
- RAM中可对Log Service资源进行授权的Action
- Log Service API发生RAM用户访问主账号资源时的鉴权规则

16.2. 资源列表

日志服务支持通过主账号为子账号授予相关资源权限，方便子账号对部分资源进行操作。本文档主要介绍为子账号授权的资源类型。

目前可以授予RAM用户的资源类型及其描述方式如下：

资源类型	授权策略中的资源描述方式
Project、Logstore	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}
Project、Logstore、Shipper	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}/shipper/${shipperName}
	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/*

资源类型	授权策略中的资源描述方式
Project、Config	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logtailconfig/${logtailConfigName}
	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logtailconfig/*
Project、MachineGroup	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/machinegroup/${machineGroupName}
	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/machinegroup/*
Project、ConsumerGroup	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}/consumergroup/${consumerGroupName}
	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}/consumergroup/*
Project、SavedSearch	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/savedsearch/${savedSearchName}
	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/savedsearch/*
Project、Dashboard	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/dashboard/${dashboardName}
	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/dashboard/*
Project、Alarm	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/alert/${alarmName}
	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/alert/*
泛指模式	acs:log:${regionName}:${projectOwnerAliUid}:*
	acs:log:*:${projectOwnerAliUid}:*

> **说明** Log Service中的资源具有层级关系。Project是根资源，Logstore、config、machinegroup是Project的子资源且相互之间平级，shipper和consumergroup是Logstore的子资源。

授权策略中各个参数的说明如下：

参数	说明
${regionName}	某个region的名称。
${projectOwnerAliUid}	阿里云账号ID。
${projectName}	Project的名称。
${logstoreName}	Logstore的名称。

参数	说明
${logtailconfig}	Logtail配置的名称。
${machineGroupName}	机器组的名称。
${shipperName}	日志投递规则的名称。
${consumerGroupName}	协同消费组的名称。
${savedSearchName}	快速查询的名称。
${dashboardName}	仪表盘的名称。
${alarmName}	报警规则的名称。

16.3. 动作列表

本文介绍RAM中对日志服务资源进行授权的动作列表及说明。

RAM中可对Log Service资源进行授权的Action

在RAM中，可以对一个Log Service资源进行以下Action的授权。

资源类型	动作（Action）	说明
Project	log:CreateProject	创建Project。
	log:GetProject	查看Project属性。
Logstore	log:GetLogStoreLogs	查询指定Project下某个Logstore中的日志数据。
	log:GetLogStoreHistogram	查询指定的Project下某个Logstore中日志的分布情况。
	log:GetLogStore	查看Logstore属性。
	log:ListLogStores	列出指定Project下的所有Logstore的名称。
	log:CreateLogStore	在Project下创建Logstore。
	log:DeleteLogStore	删除Logstore，包括所有Shard数据以及索引等。
	log:UpdateLogStore	更新Logstore的属性。
	log:GetCursorOrData（GetCursor，PullLogs，GetCursorTime）	根据时间获得游标（cursor）；根据游标获取指定Shard的日志数据；根据游标获取对应服务端时间。
	log:ListShards	列出Logstore下当前所有可用Shard。
	log:PostLogStoreLogs	向指定的Logstore写入日志数据。
	log:GetShipperStatus	查询日志投递任务状态。
	log:RetryShipperTask	重新执行失败的日志投递任务。
	log:CreateIndex	为指定Logstore创建索引。

资源类型	动作（Action）	说明
	log:DeleteIndex	删除指定Logstore的索引。
	log:GetIndex	查询指定Logstore的索引。
	log:UpdateIndex	更新指定Logstore的索引。
Config	log:CreateConfig	在Project下创建日志Logtail配置。
	log:UpdateConfig	更新Logtail配置内容。
	log:DeleteConfig	删除指定Logtail配置。
	log:GetConfig	获得一个Logtail配置的详细信息。
	log:ListConfig	列出Project下所有Logtail配置信息，可以通过参数进行翻页。
MachineGroup	log:CreateMachineGroup	创建机器组，用以指定需要收集日志的服务器。
	log:UpdateMachineGroup	更新机器组信息。
	log:DeleteMachineGroup	删除机器组。
	log:GetMachineGroup	查看具体的机器组信息。
	log:ListMachineGroup	查看机器组名称列表。
	log:ListMachines	获得机器组下属于用户并与Server端连接的机器状态信息。
	log:ApplyConfigToMachineGroup	将配置应用到机器组。
	log:RemoveConfigFromMachineGroup	从机器组中删除配置。
	log:GetAppliedMachineGroups	获得机器组上已经被应用的机器列表。
	log:GetAppliedConfigs	获得机器组上已经被应用的配置名称。
ConsumerGroup	log:CreateConsumerGroup	在指定的Logstore上创建一个消费组。
	log:UpdateConsumerGroup	修改指定消费组属性。
	log:DeleteConsumerGroup	删除一个指定的消费组。
	log:ListConsumerGroup	查询指定Logstore的所有消费组。
	log:UpdateCheckPoint	更新指定消费组的某个Shard的checkpoint。
	log:HeartBeat	为指定消费者发送心跳到服务端。
	log:GetCheckPoint	获取指定消费组消费的某个或者所有Shard的checkpoint。
SavedSearch	log:CreateSavedSearch	创建快速查询。
	log:UpdateSavedSearch	更新快速查询。
	log:GetSavedSearch	查看指定快速查询。

资源类型	动作（Action）	说明
	log:DeleteSavedSearch	删除快速查询。
	log:ListSavedSearch	查看快速查询列表。
Dashboard	log:CreateDashboard	创建仪表盘。
	log:UpdateDashboard	更新仪表盘。
	log:GetDashboard	查看指定仪表盘。
	log:DeleteDashboard	删除仪表盘。
	log:ListDashboard	查看仪表盘列表。
Job	log:CreateJob	创建任务。例如创建告警、订阅。
	log:UpdateJob	更新任务。
App	log:CreateApp	日志服务APP的创建权限。
	log:UpdateApp	日志服务APP的更新权限。
	log:GetApp	日志服务APP的查看权限。
	log:DeleteApp	日志服务APP的删除权限。

16.4. 鉴权规则

当子账号通过日志服务OpenAPI对主账号的资源进行访问时，日志服务后台对RAM进行权限检查，以确保资源拥有者的确将相关资源的相关权限授予了调用者。本文档为您列举日志服务API发生子账号访问主账号资源时的鉴权规则。

Logstore

每个不同的日志服务API会根据涉及到的资源以及API的语义来确定需要检查哪些资源的权限。具体各类API的鉴权规则见下表。

Action	Resource
log:GetLogStore	acs:log:${regionName}:${projectOwnerAliUid}:**project**/${projectName}/**logstore**/${logstoreName}
log:ListLogStores	acs:log:${regionName}:${projectOwnerAliUid}:**project**/${projectName}/**logstore**/*
log:CreateLogStore	acs:log:${regionName}:${projectOwnerAliUid}:**project**/${projectName}/**logstore**/*
log:DeleteLogStore	acs:log:${regionName}:${projectOwnerAliUid}:**project**/${projectName}/**logstore**/${logstoreName}
log:UpdateLogStore	acs:log:${regionName}:${projectOwnerAliUid}:**project**/${projectName}/**logstore**/${logstoreName}

loghub

数据写入以及消费类API，其中获取数据游标API GetCursor以及获取数据API GetLogs共用同一个Action（log:GetCursorOrData）。

Action	Resource
log:GetCursorOrData	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}
log:ListShards	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}
log:PostLogStoreLogs	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}

config

Action	Resource
log:CreateConfig	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logtailconfig/*
log:UpdateConfig	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logtailconfig/${logtailConfigName}
log:DeleteConfig	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logtailconfig/${logtailConfigName}
log:GetConfig	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logtailconfig/${logtailConfigName}
log:ListConfig	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logtailconfig/*

machinegroup

Actions	Resources
log:CreateMachineGroup	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/machinegroup/*
log:UpdateMachineGroup	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/machinegroup/${machineGroupName}
log:DeleteMachineGroup	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/machinegroup/${machineGroupName}
log:GetMachineGroup	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/machinegroup/${machineGroupName}
log:ListMachineGroup	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/machinegroup/*
log:ListMachines	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/machinegroup/${machineGroupName}

config和machinegroup交互类API

Actions	Resources
log:ApplyConfigToGroup	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logtailconfig/${logtailConfigName} acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/machinegroup/${machineGroupName}
log:RemoveConfigFromGroup	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logtailconfig/${logtailConfigName} acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/machinegroup/${machineGroupName}
log:GetAppliedMachineGroups	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logtailconfig/${logtailConfigName}
log:GetAppliedConfigs	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/machinegroup/${machineGroupName}

16.5. 通过STS实现跨账号访问日志服务资源

阿里云账号可以通过创建并授权用户角色的方式赋予其他云账号一定的资源权限，其他云账号扮演该角色，并为其名下的RAM用户授予AssumeRole权限之后，其他云账号或其子账号可以通过访问STS接口获取临时AK和Token函数，调用日志服务API接口。

背景信息

出于业务隔离或项目外包等需求，云账号A希望将部分日志服务业务授权给云账号B，由云账号B操作维护这部分业务。基本需求如下：

- 云账号B拥有向企业A的日志服务中写入数据和使用消费组的权限。
- 云账号B的指定RAM用户也拥有日志服务的写入和消费组权限。
- 云账号B可获取STS临时凭证，访问日志服务API接口，详情请参见STS。

赋权过程概述

1. 云账号A创建RAM角色，并指定云账号B扮演该角色并赋予日志服务的指定权限。
2. 云账号B创建RAM用户B1，并为其赋予 `AliyunSTSAssumeRoleAccess` （调用STS AssumeRole接口）的系统策略。
3. RAM账号B1调用STS AssumeRole接口访问日志服务API接口，操作日志服务资源。

步骤1：云账号A为云账号B创建RAM角色并授权

云账号A创建RAM角色，并指定云账号B扮演该角色并为角色赋予日志服务的指定权限。

可以通过RAM控制台创建RAM角色，请参见创建RAM用户及授权或通过RAM的API CreateRole创建RAM角色，请参见CreateRole，以下以控制台创建为例进行详细步骤说明。

1. 云账号A登录RAM控制台。
2. 云账号A创建RAM角色，并指定云账号B扮演该角色。
   i. 在左侧导航栏，单击**RAM角色管理**。
   ii. 在**RAM角色管理**页面，单击**创建RAM角色**。
   iii. 选中可信实体类型为**阿里云账号**，单击**下一步**。

iv. 输入**角色名称**和**备注**，**选择云账号**为**其他云账号**，填写云账号B的账号ID，单击**完成**。

> **说明** 将鼠标悬停在控制台右上角头像的位置，单击**安全信息管理**，即可查询主账号ID。

以上步骤中创建的RAM角色详情如下。

```
{
 "Statement": [
  {
   "Action": "sts:AssumeRole",
   "Effect": "Allow",
   "Principal": {
    "RAM": [
     "acs:ram::<云账号B的账号ID>:root"
    ]
   }
  }
 ],
 "Version": "1"
}
```

3. 创建权限策略。
   i. 在左侧导航栏的**权限管理**菜单下，单击**权限策略管理**。
   ii. 在**权限策略管理**页面，单击**创建权限策略**。
   iii. 在**新建自定义权限策略**页面，配置如下参数：

参数	说明
策略名称	输入策略名称。
备注	输入您需要备注的内容。
配置模式	选中**脚本配置**。

参数	说明

策略内容	此处配置的权限策略为云账号A授予云账号B的权限内容。
	如果仅需要写数据，具体权限如下：
	`{ "Version": "1", "Statement": [ { "Action": "log:PostLogStoreLogs", "Resource": "*", "Effect": "Allow" } ] }`
	如果需要通过协同消费库获取数据，具体权限如下：
	`{ "Version": "1", "Statement": [ { "Action": [ "log:GetCursorOrData", "log:CreateConsumerGroup", "log:ListConsumerGroup", "log:ConsumerGroupUpdateCheckPoint", "log:ConsumerGroupHeartBeat", "log:GetConsumerGroupCheckPoint", "log:UpdateConsumerGroup" ], "Resource": "*", "Effect": "Allow" } ] }`
	以上两类资源都是授权指定用户的所有Project和Logstore，如果您需要授权指定Project和Logstore，请参考如下内容：
	■ 授权指定Project：`acs:log:*:{projectOwnerAliUid}:project/*`。
	■ 授权指定Logstore：`acs:log:*:{projectOwnerAliUid}:project/{projectName}/logstore/{logstoreName}/*`。
	完整的资源说明请参见日志服务RAM资源。

iv. 单击**确定**。

4. 云账号A为RAM角色授权。
   i. 在左侧导航栏，单击**RAM角色管理**。
   ii. 在**RAM角色名称**列表下，找到目标RAM角色，单击**添加权限**。
   iii. **权限类型**选择**自定义策略**并在列表中选中步骤3创建的权限策略，单击**确定**。

iv. 单击**完成**，即完成授权。

步骤2：云账号B创建RAM用户B1并授权

云账号B创建RAM用户B1，并为其授予 `AliyunSTSAssumeRoleAccess` （调用STS AssumeRole接口）的系统策略。

1. 云账号B登录RAM控制台。
2. 在左侧导航栏中，选择**人员管理 > 用户**。
3. 在**用户**页面，单击**创建用户**。
4. 在**创建用户**页面，参考如下说明进行参数配置并单击**确定**。

参数	说明
用户账号信息	请输入**登录名称**及**显示名称**。
访问方式	选中**控制台密码登录**和**编程访问**。

5. 返回**用户**页面，在**用户登录名称/显示名称**列表中，找到目标RAM用户，单击**添加权限**。
6. 在**添加权限**页面，**被授权主体**会自动填入。选中**系统策略**，在**权限策略名称**列表下，单击需要授予RAM角色的权限策略**AliyunSTSAssumeRoleAccess**，单击**确定**。
7. 单击**完成**。

步骤3：RAM账号B1获取STS临时凭证访问日志服务

1. 调用STS AssumeRole接口获取临时AK和Token。更多信息，请参见AssumeRole。您可以选择以下方式调用该接口：

   通过STS SDK调用。更多信息，请参见STS SDK概览。
2. 调用日志服务接口，关于日志服务SDK请参见SDK参考。

示例代码

示例代码基于Java SDK，以云账号B通过STS向用户A的Project写入数据为例，详情请下载并参见示例代码。

17.公共资源说明

17.1. 数据加密

日志服务支持通过密钥管理服务KMS（Key Management Service）对数据进行加密存储，提供数据静态保护能力。本文主要介绍日志服务的数据加密机制以及使用KMS进行数据加密的操作步骤。

前提条件

已开通密钥管理服务，详情请参见开通密钥管理服务。

数据加密机制

日志服务通过KMS托管密钥，实现数据加密功能。数据加密机制如下：

- 日志服务支持的数据加密算法为AES算法和国密算法M4。
- KMS生成和管理您的主密钥CMK（Customer Master Key），并保障密钥的安全性。
- 日志服务支持如下两种加密类型：
  - 通过日志服务的服务密钥加密：日志服务为每个Logstore生成独立的数据加密密钥，用于数据加密。该加密密钥永不过期。
  - 通过用户自带密钥（BYOK）加密：您可以在KMS控制台上创建主密钥CMK，并授权日志服务相应的权限。日志服务调用KMS接口时，使用该CMK创建用于数据加密的密钥。当您的主密钥CMK被删除或禁用后，BYOK密钥失效。

> 注意
>
> - 仅调用CreateLogstore API创建Logstore时，可设置数据加密，即可配置加密算法和加密类型。配置后，不支持修改。
> - 由KMS BYOK生成的主密钥（CMK）失效后，Logstore上的所有读写请求都会失败。

BYOK授权

如果使用BYOK加密，则需先完成RAM授权。

1. 登录RAM控制台。
2. 创建RAM角色，操作步骤请参见步骤1：创建用户角色并指定受信云账号。
3. 修改RAM角色的信任策略，操作步骤请参见修改RAM角色的信任策略。

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

4. 为RAM角色授权AliyunKMSReadOnlyAccess、AliyunKMSCryptoUserAccess权限，操作步骤请参见为RAM角色授权。

5. 如果是使用RAM用户操作BYOK加密，还需对RAM用户授予PassRole权限，即创建自定义授权策略，并完成授权。操作步骤请参见创建自定义策略、为RAM用户授权。

```
{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ram:PassRole",
      "Resource": "acs:ram::*" #RAM角色的ARN，详情请参见如何获取RAM角色标识。
    }
  ]
}
```

配置数据加密

在调用CreateLogstore API创建Logstore时，添加如下encrypt_conf节点完成加密设置。

```
encrypt_conf = {
   "enable" : True,          # 是否启用加密。
  "encrypt_type" : "default"     # 加密算法，只支持default和m4。
  "user_cmk_info" :          # 可选字段。如果设置该字段，则表示使用自带密钥（BYOK），否则使用日志服务的服务密钥。
  {
      "cmk_key_id" : ""       # BYOK的主密钥ID，例如f5136b95-2420-ab31-xxxxxxxxx。
     "arn" : ""           # RAM角色的ARN，详情请参见如何获取RAM角色标识。
     "region_id" : ""        # 主密钥所在的地域ID。
  }
}
```

17.2. 数据模型

本文介绍采集到日志服务中的日志数据的数据模型。

为方便您理解日志服务中日志数据的数据模型并顺利使用，从如下基本概念展开介绍。

地域（Region）

地域为阿里云的服务节点。通过在不同的阿里云地域部署服务，实现您的服务距离您的客户更近，获得更低的访问延时及更好的用户体验。目前阿里云在全国各地拥有多个服务地域。

项目（Project）

项目为日志服务中的基本管理单元，用于资源隔离和控制。您可以通过项目来管理某一个应用的所有日志及相关的日志源。

日志库（Logstore）

日志库是日志服务中日志数据的收集、存储和消费单元。每个日志库隶属于一个项目，每个项目可以创建多个日志库。您可以根据实际需求为某一个项目生成多个日志库，其中常见的做法是为一个应用中的每类日志创建一个独立的日志库。例如：您有一个名为big-game的游戏应用，服务器上有三种日志：操作日志（operation_log）、应用程序日志（application_log）以及访问日志（access_log），您可以首先创建名为big-game的项目，然后在该项目下面为这三种日志创建三个日志库，分别用于它们的收集、存储和消费。

日志（Log）

日志为日志服务中处理的最小数据单元。日志服务采用半结构数据模式定义一条日志，具体数据模型如下：

数据域	描述	要求
time	日志中保留字段，用以表示日志产生的时间，一般由日志中的时间直接生成。	整型，Unix标准时间格式。单位为秒，表示从1970-1-1 00:00:00 UTC计算起的秒数。
topic	用户自定义字段，用以标记一批日志。例如访问日志可根据不同站点进行标记。	包括空字符串在内的任意字符串，长度不超过128字节。默认情况下，该字段为空字符串。
source	日志的来源地，例如产生该日志机器的IP地址。	任意不超过128字节的UTF-8编码字符串。默认该字段为空。
content	用以记录日志的具体内容。内容部分由一个或多个内容项组成，每一个内容项由Key-Value对组成。	Key为UTF-8编码字符串，包含字母、下划线和数字，且不以数字开头，其长度不超过128字节，且不可以使用如下关键字： • `__time__` • `__source__` • `__topic__` • `__partition_time__` • `__extract_others__` • `__extract_others__` Value为不超过1024*1024字节的任意UTF-8编码字符串。
tags	日志的标签，包括： • 用户自定义标签：您通过PutLogs写入数据时添加的标签。 • 服务端为您添加的标签，包括 `__client_ip__` 和 `__receive_time__` 。	字典格式，Kev和Value均为字符串类型。在控制台查询日志时，以 `__tag__:` 为前缀展示。

日志主题（Topic）

一个日志库内的日志可以通过日志主题（Topic）来划分。您可以在写入时指定日志主题，并在查询时指定查询的日志主题。例如，一个平台用户可以使用用户编号作为日志主题写入日志。这样在查询时可利用日志主题让不同用户仅看到自己的日志。如果不需要划分一个日志库内的日志，让所有日志使用相同的日志主题即可。

> **说明** 空字符串是一个有效的日志主题（Topic），且无论是写入还是查询日志时，默认的日志主题都是空字符串。所以，如果不需要使用日志主题，最简单的方式就是在写入和查询日志时都使用默认日志主题，即空字符串。

实际使用场景中，日志的格式多样。为了帮助理解，以下以一条Nginx原始访问日志如何映射到日志服务中日志数据模型为例说明。假设用户Nginx服务器的IP地址为10.10.10.1，如下为其一条原始日志：

```
10.1.1.1 - - [01/Mar/2012:16:12:07 +0800] "GET /Send?AccessKeyId=82251054** HTTP/1.1" 200 5 "-" "Mozilla/5.0 (X11; Linux i686 on x86_64; rv:10.0.2) Gecko/20100101 Firefox/10.0.2"
```

将该条原始日志映射到日志服务中日志数据模型，具体如下。

数据域	内容	说明
topic	None	沿用默认值（空字符串）。
time	1330589527	日志产生的精确时间（精确到秒），从原始日志中的时间戳转换而来。
source	10.10.10.1	使用服务器IP地址作为日志源。
content	Key-Value对	日志具体内容。

您可以自己决定如何提取日志原始内容并组合成Key-Value对，如下表所示：

key	value
ip	10.1.1.1
method	GET
status	200
length	5
ref_url	-
browser	Mozilla/5.0 (X11; Linux i686 on x86_64; rv:10.0.2) Gecko/20100101 Firef

Logs

由多条日志产生的集合。

LogGroup

一组日志的集合。

LogGroupList

一组LogGroup集合，用于结果返回。

编码方式

系统目前支持以下内容编码方式（将来可扩展），Restful API层通过Content-Type表示。

含义	描述	Content-Type
Protobuf	Protobuf对数据模型进行编码	application/x-protobuf

Protobuf格式请参见数据编码方式。

> 说明
>
> 由于Protobuf对Key-Value对不要求唯一性，因此需要避免出现该情况，否则行为为未定义。
>
> 对于同一个Message中的字段，在Protobuf编码时需要按照字段编号的顺序，否则数据可能会解析失败。

17.3. 数据编码方式

日志服务支持使用Protocol Buffer格式作为标准的日志写入格式。

Protocol Buffer格式用于结构化数据交换格式，当用户需要写入日志时，需要把原始日志数据序列化成如下格式的Protocol Buffer数据流，然后才能通过API写入服务端。

```
message Log
{
  required uint32 Time = 1;// UNIX Time Format
  message Content
  {
    required string Key = 1;
    required string Value = 2;
  }
  repeated Content Contents = 2;
}
message LogTag
{
  required string Key = 1;
  required string Value = 2;
}
message LogGroup
{
  repeated Log Logs= 1;
  optional string Reserved = 2; // reserved fields
  optional string Topic = 3;
  optional string Source = 4;
  repeated LogTag LogTags = 6;
}
message LogGroupList
{
  repeated LogGroup logGroupList = 1;
}
```

> 说明
> - 在使用Protobuf时要保证Key-Value对的唯一性，否则会出现行为未定义的错误。
> - 关于Protocol Buffer格式的更多信息请参见Github首页。
> - 关于日志服务写入日志的API的详细描述，请参见PutLogs。

17.4. 日志库

日志库（Logstore）是日志服务中日志数据的采集、存储和查询单元。

- 命名规则
  - 只能包含小写字母、数字、连字符（-）和下划线（_）。
  - 必须以小写字母或者数字开头和结尾。
  - 长度为3~63字符。
- 完整资源示例

```
{
  "logstoreName" : "access_log",
  "ttl": 1,
  "shardCount": 2,
  "autoSplit": true,
  "maxSplitShard": 64,
  "createTime":1439538649,
  "lastModifyTime":1439538649
}
```

- 示例参数说明

  资源示例中各参数含义如下表所示：

参数名称	类型	是否必须	描述
logstoreName	string	是	日志库名称，在该Project下唯一。
ttl	integer	是	日志数据生命周期（TTL），单位为天，最小为1天。
shardCount	integer	是	日志数据服务单元。
autoSplit	bool	否	是否自动分裂Shard。
maxSplitShard	int	否	自动分裂时最大的Shard个数，最小值为1，最大值为64。当autoSplit为true时必须指定。
createTime	integer	否	Logstore创建时间。
lastModifyTime	integer	否	Logstore更新时间。

17.5. 分区

Logstore读写日志必定保存在某一个分区（Shard）上。每个日志库（Logstore）分若干个分区，每个分区由MD5左闭右开区间组成，每个区间范围不会相互覆盖，并且所有的区间的范围是MD5整个取值范围。

Shard是每个Logstore下读写基本单元，每个Logstore会指定分区数目，每个分区能承载一定量的服务能力：

- 写入：5MB/s，500次/s。
- 读取：10MB/s，100次/s。

在向Shard读写数据过程中，读必须指定对应的Shard，而写的过程中可以使用Load-Balance方式。Load-Balance会根据后台系统负载自动均衡，保证写入的高可用性。

分区的完整资源的参数含义示例如下：

参数名称	类型	必须	描述
shardID	int	是	Logstore下Shard的唯一ID，由系统自动生成，类型为整型。

17.6. Logtail配置

Logtail配置是Logtail采集日志的策略集合。通过在创建Logtail配置时设置数据源、采集模式等参数，实现定制化的采集策略。本文档介绍API模式下Logtail配置相关的参数说明。

基础参数

Logtail配置的基础参数说明如下表所示：

Lotail配置的基础参数

参数名称	数据类型	是否必填	示例值	描述
configName	String	是	config-sample	Logtail配置的名称，在其所属Project内必须唯一。创建Logtail配置成功后，无法修改其名称。 命名规则如下： • 只能包括小写字母、数字、短划线（-）和下划线（_）。 • 必须以小写字母或者数字开头和结尾。 • 长度必须在2~128字符之间。
inputType	String	是	file	日志输入的方式。具体说明如下： • plugin：通过Logtail插件采集MySQL Binlog等日志。 • file：通过固定模式（正则模式、分隔符模式等）采集文本文件中的日志。 。
inputDetail	JSON Object	是	无	日志输入的相关配置。更多信息，请参见inputDetail参数。
outputType	String	是	LogService	日志输出的方式，只支持LogService，即只支持将数据上传到日志服务。
outputDetail	JSON Object	是	无	日志输出的相关配置。更多信息，请参见outputDetail参数。
logSample	String	否	无	日志样例。
createTime	int32	否		创建Logtail配置的时间。 服务端返回参数，不支持设置。
lastModifyTime	int32	否		最近一次修改Logtail配置的时间。 服务端返回参数，不支持设置。

inputDetail参数

inputDetail参数用于配置日志输入的相关信息。

- 基础参数

inputDetail基础参数

参数名称	数据类型	是否必填	示例值	描述
filterKey	Array	否	["ip"]	用于过滤日志的字段，只有该字段的值满足filterRegex参数中设置的正则表达式时，该日志才会被采集。
filterRegex	Array	否	["regex1"]	与filterKey对应的正则表达式，filterRegex的长度和filterKey的长度必须相同。

参数名称	数据类型	是否必填	示例值	描述
shardHashKey	Array	否		数据写入模式。默认按照负载均衡模式写入，开启后按照Shard模式写入。支持的值包括__topic__、__hostname__、__source__。
enableRawLog	Boolean	否		是否上传原始日志。
sensitive_keys	Brray	否	无	脱敏功能配置。更多信息，请参见sensitive_keys参数。
mergeType	String	否	topic	聚合方式，具体说明如下： ○ topic（默认）：根据Topic聚合。 ○ logstore：根据Logstore聚合。
delayAlarmBytes	int	否	209715200	采集进度落后的告警阈值，默认值为209715200，即200 MB。
adjustTimezone	Boolean	否		是否调整日志时区，仅在配置时间解析的情况下使用。
logTimezone	String	否	GMT+08:00	时区偏移量。例如日志时间为东八区，则该值为GMT+08:00。
priority	int	否	0	日志发送优先级。 ○ 0（默认）：低优先级。 ○ 1：高优先级。
advanced	JSON Object	否	无	扩展功能。更多信息，请参见。

sensitive_keys参数说明如下表所示：

- 参数配置

sensitive_keys参数

参数名称	数据类型	是否必填	示例值	描述
key	String	是	content	日志字段名称。
type	String	是	const	脱敏方式，具体说明如下： ■ const：将敏感内容替换成const字段取值内容。 ■ md5：将敏感内容替换为其对应的MD5值。
regex_begin	String	是	'password':'	敏感内容的前缀。
regex_content	String	是	[^']*	敏感内容的正则表达式。
all	Boolean	是	true	是否替换该字段中所有的敏感内容。 ■ （推荐）true：替换。 ■ false：部分替换。
const	String	否	"********"	当type设置为const时，必须配置。

- 配置示例

  例如日志中content字段的值为 `[{'account':'1812213231432969','password':'04a23f38'},{'account':'1812213685634','password':'123a'}]` ，现需要将 `password` 部分脱敏，则sensitive_keys配置为：

```
"key" : "content"
"type" : "const"
"regex_begin" : "'password':'"
"regex_content" : "[^']*"
"all" : true
"const" : "********"
```

- 日志样例

```
[{'account':'1812213231432969','password':'********'},{'account':'1812213685634','password':'********'}]
```

advanced参数说明如下表所示：

advanced参数说明

参数名称	数据类型	是否必填	示例	描述
exactly_once_concurrency	int	否	1	是否启用ExactlyOnce功能。ExactlyOnce功能用于指定单个文件运行的发送并发数。取值范围：0~64。 ○ 配置为0，表示不启用ExactlyOnce功能。
enable_log_position_meta	Boolean	否	true	是否在日志中添加该日志所属原始文件的元数据信息，即新增__tag__:__inode__字段和__file_offset__字段。 ○ true：添加。 ○ false：不添加。
specified_year		否	0	当原始日志的时间缺少年份信息时，您可以设置该参数，使用当前时间中的年份或指定年份补全日志时间。 ○ 配置为0，表示使用当前时间中的年份。 ○ 配置为具体的年份（例如2020），表示使用指定年份。

- 文本日志的Logtail配置
  - 基础配置

参数名称	数据类型	是否必填	示例	描述
logType	String	是	common_reg_log	日志的采集模式。具体说明如下： ■ json_log：JSON模式。 ■ apsara_log：飞天模式。 ■ common_reg_log：完整正则模式。 ■ delimiter_log：分隔符模式。

参数名称	数据类型	是否必填	示例	描述
logPath	String	是	/var/log/httpd/	日志文件所在的目录。
filePattern	String	是	access*.log	日志所在的文件名。
topicFormat	String	是	none	Topic生成方式，支持以下四种类型： ■ none：不生成日志主题。 ■ default：将日志文件路径作为日志主题。 ■ group_topic：将应用该Logtail配置的机器组topic属性作为日志主题。 ■ 文件路径正则表达式：将日志文件路径的某一部分作为日志主题，如*/var/log/(.*).log*。 更多信息，请参见日志主题。
timeFormat	String	否	%Y/%m/%d %H:%M:%S	日志时间格式。更多信息，请参见时间格式。
preserve	Boolean	否	true	如果一个日志文件在指定时间内没有任何更新，则认为该文件已超时。 ■ true（默认）：永不超时。 ■ false：如果日志文件在30分钟内没有更新，则认为已超时，并不再监控该文件。
preserveDepth	integer	否	1	当设置preserve为false时，指定监控不超时目录的深度，取值范围为1~3。
fileEncoding	String	否	utf8	日志文件编码格式，取值为utf8、gbk。
discardUnmatch	Boolean	否	true	是否丢弃匹配失败的日志。
maxDepth	int	否	100	设置日志目录被监控的最大深度。最大深度范围：0~1000，0代表只监控本层目录。
delaySkipBytes	int	否	0	采集落后时是否丢弃落后数据的阈值。 ■ 0（默认）：不丢弃。 ■ 其他值：当采集落后超过该值时，则直接丢弃落后的数据。
dockerFile	Boolean	否	false	采集的目标文件是否为容器内文件，默认为false。
dockerIncludeLabel	JSON Object	否	无	容器Label白名单，采集包含白名单中Label的Docker容器日志，为空表示全部采集。
dockerExcludeLabel	JSON Object	否	无	容器Label黑名单，不采集包含黑名单中Label的Docker容器日志，为空表示全部采集。

参数名称	数据类型	是否必填	示例	描述
dockerIncludeEnv	JSON Object	否	无	容器环境变量白名单，采集包含白名单中的环境变量的日志，为空表示全部采集。
dockerExcludeEnv	JSON Object	否	无	容器环境变量黑名单，采集不包含黑名单中的环境变量的日志，为空表示全部采集。

- 完整正则和极简模式特有配置

参数名称	数据类型	是否必填	示例值	描述
key	Array	是		日志内容提取结果的key列表。
logBeginRegex	String	否		行首正则表达式。
regex	String	否		提取字段的正则表达式。

```
{
  "configName": "logConfigName",
  "outputType": "LogService",
  "inputType": "file",
  "inputDetail": {
    "logPath": "/logPath",
    "filePattern": "*",
    "logType": "common_reg_log",
    "topicFormat": "default",
    "discardUnmatch": false,
    "enableRawLog": true,
    "fileEncoding": "utf8",
    "maxDepth": 10,
    "key": [
      "content"
    ],
    "logBeginRegex": ".*",
    "regex": "(.*)"
  },
  "outputDetail": {
    "projectName": "test-project",
    "logstoreName": "test-logstore"
  }
}
```

- JSON模式特有配置

参数名称	数据类型	是否必填	示例值	描述
timeKey	String	否		指定时间字段的key名称。

- 分隔符模式特有配置

参数名称	数据类型	是否必填	示例值	描述
separator	String	否	,	根据您的日志格式选择正确的分隔符。更多信息，请参见附录：分隔符简介及日志样例。
quote	String	是	\	当日志字段内容中包含分隔符时，需要指定引用符进行包裹，被引用符包裹的内容会被日志服务解析为一个完整字段。请根据您的日志格式选择正确的引用符。更多信息，请参见附录：分隔符简介及日志样例。
key	Array	是	["ip", "time"]	日志内容提取结果的key列表。
timeKey	String	是	time	指定时间字段key名称，必须在key列表里面。
autoExtend	Boolean	否	true	如果日志中分割出的字段数少于配置的Key数量，是否上传已解析的字段。

```
{
  "configName": "logConfigName",
  "logSample": "testlog",
  "inputType": "file",
  "outputType": "LogService",
  "inputDetail": {
    "logPath": "/logPath",
    "filePattern": "*",
    "logType": "delimiter_log",
    "topicFormat": "default",
    "discardUnmatch": true,
    "enableRawLog": true,
    "fileEncoding": "utf8",
    "maxDepth": 999,
    "separator": ",",
    "quote": "\"",
    "key": [
      "ip",
      "time"
    ],
    "autoExtend": true
  },
  "outputDetail": {
    "projectName": "test-project",
    "logstoreName": "test-logstore"
  }
}
```

- 飞天模式特有配置

参数名称	数据类型	是否必填	示例值	描述
logBeginRegex	string	否		行首正则表达式。

- 插件配置

  inputDetail插件模式特有的配置。

参数名称	数据类型	是否必填	示例值	描述
plugin	JONS Object	是		使用Logtail插件采集日志时，需配置此参数。更多信息，请参见使用Logtail插件采集数据。

Plugin插件模式（docker标准输出类型）配置示例

```
{
  "configName": "logConfigName",
  "outputType": "LogService",
  "inputType": "plugin",
  "inputDetail": {
    "plugin": {
      "inputs": [
        {
          "detail": {
            "ExcludeEnv": null,
            "ExcludeLabel": null,
            "IncludeEnv": null,
            "IncludeLabel": null,
            "Stderr": true,
            "Stdout": true
          },
          "type": "service_docker_stdout"
        }
      ]
    }
  },
  "outputDetail": {
    "projectName": "test-project",
    "logstoreName": "test-logstore"
  }
}
```

outputDetail参数

用于配置日志输出的Project和Logstore。

参数名称	数据类型	是否必填	示例值	描述
projectName	String	是	my-project	Project名称，必须为请求的Project名。
logstoreName	String	是	my-logstore	Logstore名称。

17.7. Logtail机器组

本文介绍了机器及机器组的配置参数及完整资源示例。

机器

当机器安装Logtail并正常启动后，会根据Logtail配置中的用户信息自动关联到当前用户。

每台机器属性如下：

```
{
  "ip" : "testip1",
  "machine-uniqueid" : "testuuid1",
  "userdefined-id" : "testuserdefinedid1",
  "lastHeartbeatTime":1397781420
}
```

机器属性参数如下：

参数名称	类型	描述
ip	string	机器hostname对应的IP地址。
uuid	string	机器标示的唯一主键，由Logtail上传。
userdefined-id	string	用户自定义机器标示，由Logtail上传。
lastHeartbeatTime	integer	机器的最后心跳时间（从epoch时间开始的秒数）。

机器组

日志服务通过一个Logtail采集配置来采集多台服务器上的日志，您可以将这些服务器加入到同一个机器组，并将Logtail采集配置应用到该机器组。您可以通过如下两种方法定义一个机器组。

- IP地址：在机器组中添加服务器的IP地址，通过IP地址识别服务器。
- 自定义标识：定义属于机器组的一个标识，在对应服务器上配置对应标识进行关联。

机器组命名规则：

- 只能包括小写字母、数字、连字符（-）和下划线（_）
- 必须以小写字母或者数字开头和结尾
- 长度必须在2~128字节以内

完整资源示例

```
{   "groupName" : "testgroup",
  "groupType" : "",
  "groupAttribute" : {
    "externalName" : "testgroup",
     “groupTopic": "testgrouptopic"
  },
  "machineIdentifyType": "ip",
  "machineList" : [
    "ip1",
    "ip2"
    …
  ],
  "createTime": 1431705075,
  "lastModifyTime" : 1431705075
}
```

机器组参数说明如下：

属性名称	类型	必须	描述
groupName	string	是	机器组名称，Project下唯一。
groupType	string	否	机器组类型，默认为空。

属性名称	类型	必须	描述
machineIdentifyType	string	是	机器标识类型，分为IP和userdefined两种。
groupAttribute	object	是	机器组的属性，默认为空。
machineList	array	是	具体的机器标识，可以是IP或userdefined-id。
createTime	int	否	该机器组创建时间。
lastModifyTime	int	否	该机器组更新时间。

groupAttribute说明如下：

属性名称	类型	是否必须	描述
groupTopic	string	否	机器组的Topic，默认为空。
externalName	string	否	机器组所依赖的外部管理标识，默认为空。