ALIBABA CLOUD

阿里云

日志服务

SDK 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.概述

为了能让您更高效地使用日志服务，日志服务提供了多个语言版本（.NET、Java、Python、PHP、C等）的SDK（Software Development Kit），您可以根据业务需求选择语言版本使用。

使用前须知

不同语言的日志服务SDK具体实现细节会有所不同，但是它们都是日志服务API在不同语言上的封装，实现的功能也基本一致。具体包括如下几个方面：

- 实现对日志服务API接口的统一封装，让您不需要关心具体的API请求构建和响应解析。而且各个不同语言的接口使用也非常接近，方便您在不同语言间切换。更多信息，请参见接口规范。
- 实现日志服务API的数字签名逻辑，让您不需要关心API的签名逻辑细节，降低使用日志服务API的难度。更多信息，请参见请求签名。
- 实现日志服务日志的ProtoBuffer格式封装，让您在写入日志时不需要关心ProtoBuffer格式的具体细节。更多信息，请参见ProtoBuffer格式。
- 实现日志服务API中定义的压缩方法，让您不用关心压缩实现的细节。部分语言的SDK支持启用压缩模式写入日志（默认为使用压缩方式）。
- 提供统一的错误处理机制，让您可以使用语言所熟悉的方式处理请求异常。更多信息，请参见错误处理机制。
- 目前所有语言实现的SDK仅提供同步请求方式。

SDK列表

下表列举了日志服务不同语言的SDK的参考文档和GitHub源码。

SDK语言	参考文档	GitHub源码
Java	Java SDK	Log Service Java SDK、Log Service SDK for Java 0.6.0 API
.NET Core	.NET Core SDK	Log Service .NET Core SDK
.NET	.NET SDK	Log Service .NET SDK
PHP	PHP SDK	Log Service PHP SDK
Python	Python SDK	Log Service Python SDK、User Guide
Node.js	Node.js SDK	Log Service Node.js SDK
C	C SDK	Log Service C SDK
GO	Go SDK	Log Service Go SDK
iOS	iOS SDK	Log Service iOS SDK、Objective-C SDK
Android	Android SDK	Log Service Android SDK
C++	C++ SDK	Log Service C++ SDK
JavaScript SDK	JavaScript SDK	无

2.配置

使用SDK与日志服务的服务器端进行交互时需要指定一些基本配置，本文介绍SDK的基本配置信息。

目前，所有语言的SDK都定义了一个Client类作为入口类，这些基本配置信息在该入口类的构造时指定。

具体包括如下几项：

- 服务入口（Endpoint）：确认Client需要访问的服务入口。
- 阿里云访问密钥（AccessKey ID，AccessKey secret）：指定Client访问日志服务时使用的访问密钥。

下面详细说明这两个配置的使用方式。

服务入口（Endpoint）

当使用SDK时，首先需要明确访问的日志服务Project所在地域，例如华东1（杭州）、华北1（青岛）等，然后选择与其匹配的日志服务入口初始化Client。该服务入口与API中的服务入口定义一致，具体请参见服务入口。

- 当选择Client的Endpoint时，必须要保证您需要访问的Project的地域和Endpoint对应的地域一致，否则SDK将无法访问您指定的Project。
- 由于Client实例只能在构造时指定该服务入口，如果需要访问不同地域里的Project，则需要用不同的Endpoint构建不同的Client实例。
- 目前，所有API的服务入口均支持HTTPS协议和HTTP协议。
- 如果在阿里云ECS虚拟机内使用SDK，您还可以使用内网Endpoint来避免公网带宽开销，具体请参见服务入口。

访问密钥（AccessKey）

所有和日志服务端交互的请求都必须经过安全验证，而访问密钥就是用来对请求进行安全验证的关键因子，且以AccessKey ID和AccessKey secret方式成对出现。在Client构造时需要指定两个参数：访问密钥对（AccessKey ID，AccessKey secret）。所以，在使用SDK前，请在阿里云控制台密钥管理页面获取（或者创建）合适的密钥对。

> **说明**
>
> - 您的账号下可以拥有多组访问密钥对，但在构造Client时指定的AccessKey ID和AccessKey secret必须成对，否则无法通过服务端的安全验证。
> - 指定的访问密钥对必须处于**启用**状态，否则会被服务端拒绝请求。同样，您也可以到云控制台查看访问密钥的状态。

示例

如果您需要访问某个Project，且当前已经拥有一对处于**启用**状态的访问密钥对。如下所示：

```
AccessKeyId = "bq2sjzesjmo**************"
AccessKeySecret = "4fdO2fTDDnZPU/**************"
```

则可以使用如下实例化对应的Client：

- Java

```
String endpoint = "地域id.example.com";     //在实际使用中，请按照您实际的服务入口和接入方式编写。
String accessKeyId = "bq2sjzesjmo**************";     //用户访问密钥对中的AccessKey ID。
String accessKeySecret = "4fdO2fTDDnZPU/**************";//用户访问密钥对中的AccessKey secret。
Client client = new Client(endpoint, accessKeyId, accessKeySecret);
//use client to operate log service project.....
```

- .NET（C#）

```
String endpoint = "地域id.example.com";     // 在实际使用中，请按照您实际的服务入口和接入方式编写。
String accessKeyId = "bq2sjzesjmo**************";     //用户访问密钥对中的AccessKey ID。
String accessKeySecret = "4fdO2fTDDnZPU/**************";//用户访问密钥对中的AccessKey secret。
SLSClient client = new SLSClient(endpoint, accessKeyId, accessKeySecret);
//use client to operate sls project.....
```

- PHP

```
$endpoint = '地域id.example.com'; //在实际使用中，请按照您实际的服务入口和接入方式编写。
$accessKeyId = 'bq2sjzesjmo**************'; //用户访问密钥对中的AccessKey ID。
$accessKey = '4fdO2fTDDnZPU/**************';//用户访问密钥对中的AccessKey secret。
$client = new Aliyun_Sls_Client($endpoint, $accessKeyId, $accessKey);
//use client to operate sls project.....
```

- Python

```
# // 在实际使用中，请按照您实际的服务入口和接入方式编写。
endpoint = '地域id.example.com'
# 用户访问密钥对中的AccessKey ID。
accessKeyId = 'bq2sjzesjmo**************'
# 用户访问密钥对中的AccessKey secret。
accessKey = '4fdO2fTDDnZPU/**************'
client = LogClient(endpoint, accessKeyId, accessKey)
#use client to operate log project.....
```

3.错误处理

在使用SDK访问日志服务端时，可能会出现网络中断、网络延迟导致的请求失败。本文介绍SDK请求失败时的错误处理逻辑。

错误类型及处理原则

SDK可能出现的异常错误可以分成如下几类：

- 由日志服务端返回的错误。这类错误由日志服务端返回并由SDK处理。关于这类错误的详细细节可以参见各个API接口的具体说明和日志服务API的错误码，请参见通用错误码。
- 由SDK在向服务端发出请求时出现的网络错误。这类错误包括网络连接不通，服务端返回超时等。
- 由SDK自身产生的、与平台及语言相关的错误，如内存溢出等。

目前，各个语言SDK的实现都采取抛出异常的方式处理错误。具体原则如下：

- 由日志服务端返回的错误或者由SDK在向服务端发出请求时出现的网络错误将会被SDK处理并包装在统一的LogException类抛出给用户处理。
- 由SDK自身产生的、与平台及语言相关的错误不会被SDK处理，而是直接抛出平台及语言的Native Exception类给用户处理。

LogException

LogException类是SDK定义的、用于处理日志服务自身逻辑错误的异常类。它继承自各个语言的异常基类，提供如下异常信息：

- 错误代码（Error Code）：标示错误类型。如果是来自服务端的返回错误，则这个错误代码与API返回的错误代码一致。如果是SDK网络请求错误，则其错误代码为 `RequestError` 。具体请参见各个语言的完整API参考。
- 错误消息（Error Message）：标示错误消息。如果是来自服务端的响应错误，则这个错误消息与API返回的错误消息一致。如果是SDK网络请求错误，则其错误消息为 `request is failed.` 。具体请参见各个语言的完整API参考。
- 错误请求ID（Request Id）：标示当前错误对应于服务端的请求ID。该ID只有在服务端返回错误消息时有效，否则为空字符串。用户可以在遇到错误请求时记下该请求ID并提供给日志服务团队进行问题追踪和定位。

请求失败与重试

在使用SDK访问日志服务端时，有可能会因为网络临时中断、传输延时过程、服务端处理过慢等一系列原因导致请求失败。目前，这类错误都直接以异常抛出，日志服务内部并未对此做任何重试逻辑。所以，您在使用SDK时需要自己定义相应的处理逻辑（重试请求或者直接报错等）。

示例

假设您需要访问华东1（杭州）地域下名字为big-game的Project，且在出现网络异常时主动重试指定次数。各语言的代码片段如下：

- Java：

```
//其他代码
String accessId = "your_access_id";  //TODO：用您的真实阿里云AccessKeyId替代。
String accessKey = "your_access_key"; //TODO：用您的真实阿里云AccessKeySecret替代。
String project = "big-game";
String endpoint = "cn-hangzhou.sls.aliyuncs.com";
int max_retries = 3;
/*
 * 构建一个client
 */
Client client = new Client(accessId, accessKey, endpoint);
ListLogStoresRequest lsRequest = new ListLogStoresRequest(project);
for (int i = 0; i < max_retries; i++)
{
  try
  {
    ListLogStoresResponse res = client.ListLogStores(lsRequest)
    //TODO:处理返回的response
    break;
  }
  catch(LogException e)
  {
    if (e.GetErrorCode() == "RequestError")
    {
      if ( i == max_retries - 1)
      {
        System.out.println("request is still failed after all retries.");
        break;
      }
      else
        System.out.println("request error happens, retry it!");
    }
    else
    {
      System.out.println("error code :" + e.GetErrorCode());
      System.out.println("error message :" + e.GetErrorMessage());
      System.out.println("error requestId :" + e.GetRequestId());
      break;
    }
  }
  catch(Exception $ex)
  {
    System.out.println("unrecoverable exception when listing logstores.");
    break;
  }
}
//其他代码
```

- .NET（C#）：

```
//其他代码
String accessId = "your_access_id";  //TODO：用您的真实阿里云AccessKeyId替代。
String accessKey = "your_access_key";  //TODO：用您的真实阿里云AccessKeySecret替代。
String project = "big-game";
String endpoint = "cn-hangzhou.sls.aliyuncs.com";
int max_retries = 3;
//创建一个Client
SLSClient client = new SLSClient(endpoint, accessId, accessKey);
ListLogstoresRequest request = new ListLogstoresRequest();
request.Project = project;
for (int i = 0; i < max_retries; i++)
{
  try
  {
    ListLogstoresResponse response = client.ListLogstores(request);
    //TODO:处理返回的response
    break;
  }
  catch(LogException e)
  {
    if (e.errorCode == "SLSRequestError")
    {
      if ( i == max_retries - 1)
      {
        Console.Writeline(“request is still failed after all retries.”);
        break;
      }
      else
      {
        Console.Writeline("request error happens, retry it!");
      }
    }
    else
    {
      Console.Writeline("error code :" + e.errorCode);
      Console.Writeline("error message :" + e.Message);
      Console.Writeline("error requestId :" + e.RequestId);
      break;
    }
  }
  catch(Exception $ex)
  {
    Console.Writeline("unrecoverable exception when listing logstores.");
    break;
  }
}
//其他代码
```

- PHP：

```
<?php
//其他代码
$endpoint = 'cn-hangzhou.sls.aliyuncs.com';
$accessId = 'your_access_id'; // TODO：用你的真实阿里云AccessKeyId替代
$accessKey = 'your_access_key'; //TODO：用你的真实阿里云AccessKeySecret替代
$maxRetries = 3;
// 构建一个sls client
$client = new Aliyun_Sls_Client($endpoint, $accessId, $accessKey);
$project = 'big-game';
$request = new Aliyun_Sls_Models_ListLogstoresRequest($project);
for($i = 0; $i < $maxRetries; ++$i)
{
  try
  {
    $response = $client->ListLogstores($request);
    //TODO: 处理返回的response
    break;
  }
  catch (Aliyun_Sls_Exception $e)
  {
    if ($e->getErrorCode()=='RequestError')
    {
      if ($i+1 == $maxRetries)
      {
        echo "error code :" . $e->getErrorCode() . PHP_EOL;
        echo "error message :" . $e->getErrorMessage() . PHP_EOL;
        break;
      }
      echo 'request error happens, retry it!' . PHP_EOL;
    }
    else
    {
      echo "error code :" . $e->getErrorCode() . PHP_EOL;
      echo "error message :" . $e->getErrorMessage() . PHP_EOL;
      echo "error requestId :" . $e->getRequestId() . PHP_EOL;
      break;
    }
  }
  catch (Exception $ex)
  {
    echo 'unrecoverable exception when listing logstores.' . PHP_EOL;
    var_dump($ex);
    break;
  }
}
//其他代码
```

- Python：

```
//其他代码
endpoint = 'cn-hangzhou.sls.aliyuncs.com'
accessId = 'your_access_id' # TODO：用你的真实阿里云AccessKeyId替代
accessKey = 'your_access_key' # TODO：用你的真实阿里云AccessKeySecret替代
maxRetries = 3
# 构建一个client
client = Client(endpoint, accessId, accessKey)
project = 'big-game'
lsRequest = ListLogstoresRequest(project)
for i in xrange(maxRetries):
  try:
    res = client.ListLogstores(lsRequest)
    # TODO:处理返回的response
    break
  except LogException as e:
    if e.get_error_code() == "RequestError":
      if i+1 == maxRetries:
        print "error code :" + e.get_error_code()
        print "error message :" + e.get_error_message()
        break
      else:
        print "request error happens, retry it!"
    else:
      print "error code :" + e.get_error_code()
      print "error message :" + e.get_error_message()
      print "error requestId :" + e.get_request_id()
      break
  except Exception as e:
    print 'unrecoverable exception when listing logstores.'
    break
//其他代码
```

4.接口规范

本文介绍SDK接口需要遵循的原则。

Request-Response原则

尽管不同语言的SDK实现有所不同，但其接口都遵循Request-Response原则，即对API的调用按照如下方式进行：

1. 利用请求参数构建相应的Request实例。
2. 调用SDK中的相应接口并传入上一步的Request实例。
3. SDK接口的返回结果以相应的Response实例返回给用户。

示例

以下代码片段展示了如何获取一个Project下的所有Logstore的名称。

- Java

```
// 其他代码。
String accessId = "your_access_id";  //用您的真实阿里云AccessKeyId替代。
String accessKey = "your_access_key"; //用您的真实阿里云AccessKeySecret替代。
String project = "your_project";    //用您的真实project名称替代。
String endpoint = "region_endpoint";//在实际使用中，请按照您实际的服务入口和接入方式编写。
//构建一个Client实例。
Client client = new Client(endpoint, accessId, accessKey);
//用请求参数“project”初始化ListLogstores的请求类。
ListLogStoresRequest lsRequest = new ListLogStoresRequest(project, 0,100, "");
//使用request实例调用ListLogstores接口，且返回参数为对应的Response实例。
ListLogStoresResponse res = client.ListLogStores(lsRequest);
//访问Response实例获取请求结果。
ArrayList<String> names = res.GetLogStores();
// 其他代码。
```

- .NET (C#)

```
// 其他代码。
String accessId = "your_access_id";  //用您的真实阿里云AccessKeyId替代。
String accessKey = "your_access_key"; //用您的真实阿里云AccessKeySecret替代。
String project = "your_project";    //用您的真实project名称替代。
String endpoint = "region_endpoint";//在实际使用中，请按照您实际的服务入口和接入方式编写。
//构建一个Client实例。
SLSClient client = new SLSClient(endpoint, accessId, accessKey);
//用请求参数“project”初始化ListLogstores的请求类。
ListLogStoresRequest lsRequest = new ListLogStoresRequest();
lsRequest.Project = project;
//使用 request实例调用ListLogstores接口，且返回参数为对应的Response实例。
ListLogStoresResponse res = client.ListLogStores(lsRequest);
//访问Response实例获取请求结果。
List<String> names = res.Logstores;
// 其他代码。
```

- PHP

```
// 其他代码。
$accessId = "your_access_id";  //用您的真实阿里云AccessKeyId替代。
$accessKey = "your_access_key"; //用您的真实阿里云AccessKeySecret替代。
$project = "your_project";    //用您的真实project名称替代。
$endpoint = "region_endpoint";//在实际使用中，请按照您实际的服务入口和接入方式编写。
//构建一个Client实例。
$client = new Aliyun_Sls_Client($endpoint, $accessId, $accessKey);
//用请求参数“project”初始化ListLogstores的请求类。
$request = new Aliyun_Sls_Models_ListLogstoresRequest($project);
//使用request实例调用ListLogstores接口，且返回参数为对应Response实例。
$response = $client->listLogstores($request);
//访问Response实例获取请求结果。
$names = $response->getLogstores();
// 其他代码。
```

- Python

```
// 其他代码。
accessId = 'your_access_id';  //用您的真实阿里云AccessKeyId替代。
accessKey = 'your_access_key'; //用您的真实阿里云AccessKeySecret替代。
project = 'your_project';    //用您的真实project名称替代。
endpoint = 'region_endpoint';//在实际使用中，请按照您实际的服务入口和接入方式编写。
# 构建一个client。
client = LogClient(endpoint, accessId, accessKey)
# 用请求参数“project”初始化ListLogstores的请求类。
lsRequest = ListLogstoresRequest(project)
# 使用request实例调用ListLogstores接口，且返回参数为对应的Response实例。
res = client.list_logstores(lsRequest)
# 访问Response实例获取请求结果。
names = res.get_logstores();
// 其他代码。
```

SDK实现了多组类似ListLogStores的接口，也定义了相应的Request和Response类。除去Request-Response风格的基础接口外，各个不同语言的SDK还会提供一些包装了这些基础接口的辅助接口，使您无需自己构建Request及解析最终Reponse内容。这类接口的细节请参见SDK参考。

5.Java SDK

本文介绍安装日志服务Java SDK及使用Java SDK完成常见操作的相关步骤。

前提条件

- 已开通日志服务。更多信息，请参见开通日志服务。
- 已创建并获取AccessKey。更多信息，请参见访问密钥。
- 已安装Java开发环境。

  日志服务Java SDK支持J2SE 6.0及以上的Java运行环境，您可以执行**java -version**命令检查您已安装的Java版本。如果未安装，可以从Java官方网站下载安装包并完成安装。

步骤1：安装Java SDK

您可以通过以下两种方式安装日志服务Java SDK：

- 方式一：在Maven项目中加入依赖项（推荐方式）。

  在Maven工程中使用日志服务Java SDK，只需在*pom.xml*中加入相应依赖即可。以0.6.60版本为例，在<dependencies>中加入如下内容：

  ```
  <dependency>
    <groupId>com.aliyun.openservices</groupId>
    <artifactId>aliyun-log</artifactId>
    <version>0.6.60</version>
    </dependency>
  ```

- 方式二：在Eclipse中导入JAR包。

  以0.6.60版本为例，步骤如下：

  i. 下载Java SDK，下载链接请参见Aliyun LOG Java SDK。
  ii. 将aliyun-log-0.6.60.jar拷贝到您的项目中。
  iii. 在Eclipse中选择您的工程，右击选择**Properties > Java Build Path > Add JARs**。
  iv. 选中步骤2中拷贝的JAR文件。

步骤2：创建日志服务Client

日志服务Client是日志服务的Java客户端，用于管理Project、Logstore等日志服务资源。使用Java SDK发起日志服务请求，您需要初始化一个Client实例。

```
String accessId = "your_access_id"; //阿里云访问密钥AccessKey ID。更多信息，请参见访问密钥。阿里云主账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM账号进行API访问或日常运维。
String accessKey = "your_access_key"; //阿里云访问密钥AccessKey Secret。
String host = "cn-hangzhou-intranet.log.aliyuncs.com"; //日志服务的域名。更多信息，请参见服务入口。此处以杭州为例，其它地域请根据实际情况填写。
Client client = new Client(host, accessId, accessKey); //创建日志服务Client。
```

Java SDK示例

日志服务Java SDK提供丰富的示例程序，方便参考或直接使用，更多信息，请参见aliyun-log-java-sdk。此处以创建Project和Logstore为例进行说明，示例代码如下所示：

```
//创建Project和Logstore。
String project = "my-project"; //待创建的Project的名称。
String logstore = "my-logstore";  //待创建的Logstore的名称。
int ttl_in_day = 3;  //数据保存时间。如果配置为3650，表示永久保存。
int shard_count = 10;  //Shard数量。
LogStore store = new LogStore(logstore, ttl_in_day, shard_count);
CreateLogStoreResponse res = client.CreateLogStore(project, store);
```

说明

- 为了提高您系统的IO效率，请尽量不要直接使用SDK写数据到日志服务。如何写数据，请参见Producer Library。
- 如果您要消费日志服务中的数据，请尽量不要直接使用Java SDK中拉取数据的接口。您可以通过消费组消费日志数据，更多信息，请参见通过消费组消费日志数据。该方式屏蔽了日志服务的实现细节，并且提供了负载均衡、按序消费等高级功能。

6.Python SDK

本文介绍安装日志服务Python SDK及使用Python SDK完成常见操作的相关步骤。

前提条件

- 已开通日志服务。更多信息，请参见开通日志服务。
- 已创建并获取AccessKey。更多信息，请参见访问密钥。
- 已安装Python开发环境。

  日志服务Python SDK支持Pypy2、Pypy3、Python2.6、Python2.7、Python3.3、Python3.4、Python3.5、Python3.6和Python3.7版本。您可以执行**python -V**命令检查您已安装的Python版本。如果未安装，请从Python官网下载安装包并完成安装。
- 已安装Python的包管理工具Pip。

  您可以执行**pip -V**命令检查您已安装的Pip版本。如果未安装，请从Pip官网下载安装包并完成安装。

  > **说明** 在Windows环境中，如果提示**不是内部或外部命令**，请在环境变量中编辑Path，增加Python和pip的安装路径。pip的安装路径一般为Python所在目录的*Scripts*文件夹。配置完成后，您可能需要重启电脑使环境变量生效。

步骤1：安装Python SDK

Python SDK支持所有运行Python的操作系统，例如Linux、macOS X和Windows。

1. 在命令行工具中，以管理员身份执行如下命令安装Python SDK。

   ```
   pip install -U aliyun-log-python-sdk
   ```

2. 验证安装结果。如果提示如下类似信息，表示安装Python SDK成功。

   ```
     Running setup.py install for aliyun-log-python-sdk ... done
   Successfully installed aliyun-log-python-sdk-0.6.48.8 certifi-2020.6.20 chardet-
   3.0.4 dateparser-1.0.0 elasticsearch-7.9.1 idna-2.10 jmespath-0.10.0 protobuf-3.
   13.0 python-dateutil-2.8.1 pytz-2020.4 regex-2020.10.28 requests-2.24.0 six-1.15
   .0 tzlocal-2.1 urllib3-1.25.11
   ```

步骤2：创建日志服务Client

LogClient是日志服务的Python客户端，用于管理Project、Logstore等日志服务资源。使用Python SDK发起日志服务请求，您需要初始化一个Client实例。

> **说明** 如果您要使用https连接，则需在endpoint中加入https://前缀，例如https://cn-hangzhou.log.aliyuncs.com。

```
from aliyun.log import LogClient
endpoint = 'cn-hangzhou.log.aliyuncs.com'  #日志服务的域名。更多信息，请参见服务入口。此处以杭州为例，其它地域请根据实际情况填写。
accessKeyId = 'your_access_id'  #阿里云访问密钥AccessKey ID。更多信息，请参见访问密钥。阿里云主账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM账号进行API访问或日常运维。
accessKey = 'your_access_key'  #阿里云访问密钥AccessKey Secret。
client = LogClient(endpoint, accessKeyId, accessKey)  #创建LogClient。
```

Python SDK示例

日志服务Python SDK提供丰富的示例程序，方便参考或直接使用，更多信息，请参见aliyun-log-python-sdk。此处以创建Project和Logstore为例进行说明，示例代码如下所示：

```
res = client.create_project("my-project", "description") #输入Project名称和描述。
res = client.create_logstore('my-project', 'my-logstore', ttl=30, shard_count=3)  #输入Project名称、Logstore名称、数据保存时长和Shard数据。如果ttl配置为3650，表示永久保存。
res.log_print()
```

7..NET SDK

本文介绍安装日志服务.NET SDK及使用.NET SDK完成常见操作的相关步骤。

前提条件

- 已开通日志服务。更多信息，请参见开通日志服务。
- 已创建并获取AccessKey。更多信息，请参见访问密钥。
- 已安装.NET开发环境。

  日志服务.NET SDK支持.NET 3.5、4.0、4.5运行环境。推荐安装如下版本：

  - Microsoft .NET Framework 3.5、4.0、4.5。
  - Visual Studio 2010及以上版本。

步骤1：安装.NET SDK

您可以通过项目引入方式安装.NET SDK，此处以通过Visual Studio 2019安装.NET SDK为例。

1. 下载.NET SDK。
2. 解压安装包*aliyun-log-csharp-sdk-master.zip*到指定目录。
3. 使用Visual Studio 2019打开*aliyun-log-csharp-sdk-master\LOGSDKSample\LOGSDKSample.csproj*文件，执行调试安装。

步骤2：创建日志服务Client

LogClient是日志服务的C#客户端，用于管理Project、Logstore等日志服务资源。使用.Net SDK发起日志服务请求，您需要初始化一个Client实例。

```
String endpoint = "cn-hangzhou.log.aliyuncs.com", //日志服务的域名。更多信息，请参见服务入口。此处以杭州为例，其它地域请根据实际情况填写。
accesskeyId = "your accesskey id", //阿里云访问密钥AccessKey ID。更多信息，请参见访问密钥。阿里云主账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM账号进行API访问或日常运维。
accessKey = "your access key",  //阿里云访问密钥AccessKey Secret。
LogClient client = new LogClient(endpoint, accesskeyId, accessKey);
```

.NET SDK示例

日志服务.NET SDK提供丰富的示例程序，方便参考或直接使用，更多信息，请参见aliyun-log-csharp-sdk。此处以写入日志为例进行说明，示例代码如下所示：

```
PutLogsRequest putLogsReqError = new PutLogsRequest();
        putLogsReqError.Project = my-project;   //Project名称
        putLogsReqError.Topic = "topic";     //日志主题
        putLogsReqError.Logstore = my-logstore; //Logstore名称
        putLogsReqError.LogItems = new List<LogItem>();
        for (int i = 1; i <= 10; ++i)
        {
          LogItem logItem = new LogItem();
          logItem.Time = DateUtils.TimeSpan();
          for (int k = 0; k < 10; ++k)
            logItem.PushBack("error_" + i.ToString(), "invalid operation");
          putLogsReqError.LogItems.Add(logItem);
        }
        PutLogsResponse putLogRespError = client.PutLogs(putLogsReqError);
        Thread.Sleep(5000);
```

8..NET Core SDK

本文介绍安装日志服务.NET Core SDK及使用.NET Core SDK完成常见操作的相关步骤。

前提条件

- 已开通日志服务。更多信息，请参见开通日志服务。
- 已创建并获取AccessKey。更多信息，请参见访问密钥。
- 已创建Project。更多信息，请参见创建Project。
- 已安装NET Core开发环境。

  日志服务.NET Core SDK支持以下平台及版本：
  - .NET Core 2.0
  - .NET Framework (with .NET Core 1.x SDK) 4.6.2
  - .NET Framework (with .NET Core 2.0 SDK) 4.6.1
  - Mono 5.4
  - Xamarin.iOS 10.14
  - Xamarin.Mac 3.8
  - Xamarin.Android 8.0
  - Universal Windows Platform 10.0.16299

步骤1：安装.NET Core SDK

您可以通过项目引入方式安装.NET Core SDK，此处以通过Visual Studio 2019安装.NET Core SDK为例。

1. 下载.NET Core SDK。
2. 解压安装包*aliyun-log-dotnetcore-sdk-master.zip*到指定目录。
3. 使用Visual Studio 2019打开*aliyun-log-dotnetcore-sdk-master\Aliyun.Api.LogService.Examples\Aliyun.Api.LogService.Examples.csproj*文件，执行调试安装。

步骤2：创建日志服务Client

ILogServiceClient是日志服务的.NET Core客户端，用于管理Project、Logstore等日志服务资源。使用.NET Core SDK发起日志服务请求，您需要初始化一个Client实例。

```
public static ILogServiceClient BuildSimpleClient()
{
  return LogServiceClientBuilders.HttpBuilder
    .Endpoint("endpoint", "projectName")  // 日志服务的域名和已创建的Project名称。更多域名信息，请参见服务入口。
    .Credential("accessKeyId", "accessKey") // 阿里云访问密钥AccessKey ID和钥AccessKey Secret。更多信息，请参见访问密钥。阿里云主账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM账号进行API访问或日常运维。
    .Build();
}
```

.NET Core SDK示例

日志服务.NET SDK提供丰富的示例程序，方便参考或直接使用，更多信息，请参见aliyun-log-dotnetcore-sdk。此处以抛出业务异常为例进行说明，示例代码如下所示：

```
public async Task Caller()
{
  try
  {
    return await Wrapper();
  } catch (LogServiceException e)
  {
    // 捕获LogServiceException异常后，获取如下信息：
    Console.WriteLine($@"
      RequestId (请求ID): {e.RequestId}
      ErrorCode (错误码): {e.ErrorCode}
      ErrorMessage (错误消息): {e.ErrorMessage}");
    throw;
  }
}
private async Task<IResponse> Wrapper()
{
  var response = await client.GetLogsAsync(...);
  return response
    // 如果请求返回结果不成功，则抛出LogServiceException异常。
    .EnsureSuccess()
    .Result;
}
```

9.Node.js SDK

本文介绍安装日志服务Node.js SDK及使用Node.js SDK完成常见操作的相关步骤。

前提条件

- 已开通日志服务。更多信息，请参见开通日志服务。
- 已创建并获取AccessKey。更多信息，请参见访问密钥。
- 已安装Node.js开发环境。

步骤1：安装Node.js SDK

1. 下载Node.js SDK。
2. 执行如下命令安装Node.js SDK。

```
npm install aliyun-sdk
```

如果使用npm遇到网络问题，建议使用淘宝提供的npm镜像cnpm。

步骤2：搭建项目

本文以使用Express搭建项目为例。

1. 安装Express。更多信息，请参见安装Express。
2. 安装morgan。
3. 创建app.js文件并写入如下代码。

```
var express = require('express')
var morgan = require('morgan')
var app = express()
const logger = morgan(function (tokens, req, res) {
 return [
  tokens.method(req, res),
  tokens.url(req, res),
  tokens.status(req, res),
  tokens.res(req, res, 'content-length'), '-',
  tokens['response-time'](req, res), 'ms'
 ].join(' ')
})
app.use(logger)
app.get('/', (req, res) => res.send('Hello World!'))
app.listen(3000, () => console.log('Example app listening on port 3000!'))
```

4. 执行如下命令启动项目。

```
node app.js
```

步骤3：初始化

在app.js文件中写入如下代码，完成Node.js SDK初始化。

```
 var sls = new ALY.SLS({
  "accessKeyId": "11****ut",      //阿里云访问密钥AccessKey ID。更多信息，请参见访问密钥。阿里云主账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM账号进行API访问或日常运维。
  "secretAccessKey": "TS****7Y",  //阿里云访问密钥AccessKey Secret。
  endpoint: 'http://cn-hangzhou.log.aliyuncs.com',  //日志服务的域名。更多信息，请参见服务入口。此处以杭州为例，其它地域请根据实际情况填写。
  apiVersion: '2015-06-01'                  //SDK版本号，固定值。
 })
```

示例

日志服务Node.js SDK提供丰富的示例程序，方便参考或直接使用。更多信息，请参见aliyun-sdk-js。此处以创建Logstore为例进行说明，示例代码如下所示：

```
sls.createLogstore({
  projectName: projectName,          //待创建的Logstore所属的Project名称。
  logstoreDetail:{
    logstoreName:"test_logstore",     //设置Logstore名称。
    ttl:3,                  //设置数据保存时长，如果ttl配置为3650，表示永久保存。
    shardCount:2                //设置Shard数量。
  }
})
```

10.PHP SDK

本文介绍安装日志服务PHP SDK及使用PHP SDK完成常见操作的相关步骤。

前提条件

- 已开通日志服务。更多信息，请参见开通日志服务。
- 已创建并获取AccessKey。更多信息，请参见访问密钥。
- 已安装PHP开发环境。

  日志服务PHP SDK支持PHP 5.2.1及以上版本。您可以执行**php -v**命令检查您已安装的PHP版本。

  > **说明** 日志服务PHP SDK暂不支持依赖管理工具Composer。

步骤1：安装PHP SDK

1. 下载最新的PHP SDK包。
2. 解压*aliyun-log-php-sdk-master.zip*包到指定目录。PHP SDK是一个软件开发包，不需要额外的安装操作。PHP SDK包中除了SDK代码外，还包括一组第三方依赖包和一个Autoloader类，用于简化您的调用流程。

步骤2：创建日志服务Client

Aliyun_Log_Client是日志服务的PHP客户端，用于管理Project、Logstore等日志服务资源。使用PHP SDK发起日志服务请求，您需要初始化一个Client实例。

```
$endpoint = 'cn-hangzhou.log.aliyuncs.com'; //日志服务的域名。更多信息，请参见服务入口。此处以杭州为例，其它地域请根据实际情况填写。
$accessKeyId = '11****TY';    //阿里云访问密钥AccessKey ID。更多信息，请参见访问密钥。阿里云主账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM账号进行API访问或日常运维。
$accessKey = 'YT****ED';      //阿里云访问密钥AccessKey Secret。
$client = new Aliyun_Log_Client($endpoint, $accessKeyId, $accessKey); //创建日志服务Client。
```

PHP SDK示例

日志服务PHP SDK提供丰富的示例程序，方便参考或直接使用，更多信息，请参见aliyun-log-php-sdk。此处以创建Logstore为例进行说明，示例代码如下所示：

```
$req2 = new Aliyun_Log_Models_CreateLogstoreRequest(test-project,test-logstore,3,2); //配置Project名称、Logstore名称、数据保存时长和Shard数量。其中如果数据保存时长配置为3650，表示永久保存。
$res2 = $client -> createLogstore($req2);
```

11.Go SDK

本文介绍安装日志服务Go SDK及使用Go SDK完成常见操作的相关步骤。

前提条件

- 已开通日志服务。更多信息，请参见开通日志服务。
- 已创建并获取AccessKey。更多信息，请参见访问密钥。
- 已安装Golang环境。

步骤1：安装Go SDK

执行以下命令安装Go SDK：

```
go get -u github.com/aliyun/aliyun-log-go-sdk
```

> **说明** 安装过程中，界面不会打印提示，请耐心等待。如果安装超时，请再次执行以上命令。

步骤2：创建日志服务Client

日志服务Client是日志服务的Go客户端，用于管理Project、Logstore等日志服务资源。使用Go SDK发起日志服务请求，您需要初始化一个Client实例。

```
AccessKeyID = "your_accesskey_id"        //阿里云访问密钥AccessKey ID。更多信息，请参见访问密钥。阿里云主账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM账号进行API访问或日常运维。
AccessKeySecret = "your_accesskey_secret"   //阿里云访问密钥AccessKey Secret。
Endpoint = "cn-hangzhou.log.aliyuncs.com"   //日志服务的域名。更多信息，请参见服务入口。此处以杭州为例，其它地域请根据实际情况填写。
Client = sls.CreateNormalInterface(Endpoint,AccessKeyID,AccessKeySecret,"")  //创建Client。
```

Go SDK示例

日志服务Go SDK提供丰富的示例程序，方便参考或直接使用，更多信息，请参见aliyun-log-go-sdk。此处以创建Project和Logstore为例进行说明，示例代码如下所示：

```
project, err := client.CreateProject("project-name","description")  //输入Project名称和描述。
  if err != nil {
    fmt.Println(err)
  }
  fmt.Println(project)
  err = client.CreateLogStore("project-name","logstore-name",2,2,true,16)  //输入Project名称、Logstore名称、数据保存时长、Shard数量、开启自动分裂Shard功能和最大分裂数。如果数据保存时长配置为3650，表示永久保存。
  if err != nil {
    fmt.Println(err)
  }
```

12.Android SDK

阿里云日志服务Android SDK主要解决Android平台用户数据采集问题，目前提供写入日志数据的功能。

通过Github

https://github.com/aliyun/aliyun-log-android-sdk

13.C SDK

阿里云日志服务C SDK主要解决各种平台的日志接入问题，比如兼容MIPS芯片、OpenWrt系统等。

C SDK 使用 cURL 作为网络库，*apr/apr-util* 库解决内存管理以及跨平台问题。您只需要基于源码进行简单编译便可以使用。

此外我们还有C版本的Producer Library和Producer Lite Library，为您提供跨平台、简洁、高性能、低资源消耗的一站式日志采集方案。

> **说明** C SDK只支持数据写入，不支持其他创建资源、获取数据等接口。

Github项目地址以及更多详细说明参见：

- C Producer Library（推荐服务端使用）
- C Producer Lite Library（推荐IOT、智能设备使用）
- C 原生API（推荐二次开发使用）

14.iOS SDK

本文介绍安装日志服务iOS SDK的示例代码。

Swift

Swift示例代码如下所示。更多信息，请参见aliyun-log-ios-sdk。

```
/*
  构建日志服务客户端。
  endPoint为服务访问入口。更多信息，请参见服务入口。
  accessKeyID和accessKeySecret为阿里云访问密钥。更多信息，请参见访问密钥。
*/
let myClient = try! LOGClient(endPoint: "",
                    accessKeyID: "",
                    accessKeySecret: "",
                    projectName:"")
/* 创建日志组。 */
let logGroup = try! LogGroup(topic: "mTopic",source: "mSource")
  /* 写入一条日志。 */
  let log1 = Log()
      try! log1.PutContent("K11", value: "V11")
      try! log1.PutContent("K12", value: "V12")
      try! log1.PutContent("K13", value: "V13")
  logGroup.PutLog(log1)
 /* 写入一条日志。 */
  let log2 = Log()
      try! log2.PutContent("K21", value: "V21")
      try! log2.PutContent("K22", value: "V22")
      try! log2.PutContent("K23", value: "V23")
  logGroup.PutLog(log2)
/* 发送日志。 */
myClient.PostLog(logGroup,logStoreName: ""){ response, error in
    // handle response however you want
    if error?.domain == NSURLErrorDomain && error?.code == NSURLErrorTimedOut {
      print("timed out") // note, `response` is likely `nil` if it timed out
    }
  }
```

Objective-C

更多信息，请参见AliyunLogObjc。

15.C++ SDK

阿里云日志服务C++ SDK帮助您在C++ 程序中调用日志服务的API接口，仅用于Linux平台。

日志服务C++ SDK支持日志服务全量API接口，提供资源创建、数据读写等多种功能。

GitHub地址：https://github.com/aliyun/aliyun-log-cpp-sdk