ALIBABA CLOUD

阿里云

可信账本数据库

API参考

文档版本：20200908

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.API概览

本文列举了可信账本数据库产品中所有开放的API。按照功能目录分类，将API链接和API描述罗列出来。

账本实例API

API	描述
DescribeLedger	调用DescribeLedger查询一个账本实例的详细信息。
DescribeLedgers	调用DescribeLedgers获取账本实例列表。
ModifyLedgerAttribute	调用ModifyLedgerAttribute修改账本属性。
DeleteLedger	调用DeleteLedger删除账本实例。

访问端点API

API	描述
ListVpcEndpoints	调用ListVpcEndpoints查看一个账本实例的访问端点列表。
CreateVpcEndpoint	调用CreateVpcEndpoint在VPC中创建某个账本实例的访问端点。
DeleteVpcEndpoint	调用DeleteVpcEndpoint在指定VPC中删除某个账本的访问端点。

时间锚点API

API	描述
ListTimeAnchors	调用ListTimeAnchors获取账本的时间锚点列表。

账本记录API

API	描述
ListJournals	调用ListJournals获取账本中记录列表。
GetJournal	调用GetJournal获取一条记录信息。

账本成员API

API	描述
GetMember	调用GetMember获取某个成员的信息。
ListMembers	调用ListMembers获取账本的成员列表。

API	描述
AcceptMember	调用AcceptMember接受某个账本的邀请，成为其成员。
InviteMembers	调用InviteMembers邀请其他阿里云账号加入到某个账本中。
EnableMember	调用EnableMember恢复某个成员
DisableMember	调用DisableMember禁用某个成员。
DeleteMember	调用DeleteMember删除账本中某个成员。
ModifyMemberKey	调用ModifyMemberKey修改一个账本成员的公钥。
ModifyMemberACLs	调用ModifyMemberACLs修改成员操作权限。

2.调用方式

可信账本数据库产品接口支持HTTP调用、CLI调用、SDK调用和OpenAPI Explorer调用。

HTTP调用

您可以通过向服务端发送HTTP Get请求调用云产品API。在使用HTTP方式进行调用时，请求结构如下：

```
http://Endpoint/?Action=xx&Parameters
```

其中，

- **Endpoint**：可信账本数据库API的服务端接入地址为：ledgerdb.aliyuncs.com。
- **Action**：要执行的操作，如调用CreateLedger创建一个可信账本实例。
- **Version**：要使用的API版本，可信账本数据库的API版本是2019-11-22。
- **Parameters**：请求参数，每个参数之间用“&”分隔。
  - 请求参数由公共请求参数和API自定义参数组成。公共参数中包含API版本号、身份验证和签名等信息，详情请参见公共参数。
  - 在使用HTTP方式调用API接口时，您需要进行签名计算以保证账号安全，详细说明请参见签名机制。

下面是一个调用 `CreateLedger` 接口创建一个可信账本实例的示例：

> **说明** 为了便于用户查看，本文档中的示例都做了格式化处理。

```
https://ledgerdb.aliyuncs.com/?Action=CreateLedger
&Format=xml
&Version=2019-11-22
&Signature=xxxx%xxxx%3D
&SignatureMethod=HMAC-SHA1
&SignatureNonce=15215528852396
&SignatureVersion=1.0
&AccessKeyId=key-test
&Timestamp=2012-06-01T12:00:00Z
...
```

SDK调用

阿里云可信账本数据库提供Java、Python、Go、.NET、Node.js、PHP、C++语言的SDK。阿里云SDK免去您手动签名的过程，方便使用。下表列举了各语言SDK的Github下载地址。

SDK	GitHub下载地址
Java SDK	下载地址
Python SDK	下载地址

SDK	GitHub下载地址
Go SDK	下载地址
.NET SDK	下载地址
PHP SDK	下载地址
C++	下载地址

CLI调用

阿里云命令行工具CLI（Alibaba Cloud CLI）是基于阿里云开放API建立的管理工具。借助此工具，您可以通过调用阿里云开放API来管理阿里云产品。该命令行工具与阿里云开放API一一对应，灵活性高且易于扩展。您可基于该命令行工具对阿里云原生API进行封装，扩展出您想要的功能。更多使用说明，请参见什么是阿里云CLI? 。

在阿里云CLI中，调用RPC API时，基本命令结构如下：

```
aliyun <product> <ApiName> [--parameter1 value1 --parameter2 value2]
```

代码示例：

```
aliyun ledgerdb CreateLedger
```

OpenAPI Explorer调用

OpenAPI Explorer是可视化的API调用工具。通过该工具，您可以通过网页或者命令行调用各云产品以及API市场上开放的API，查看每次的API请求和返回结果，并生成相应SDK调用示例。

您可以直接访问https://api.aliyun.com/调用API，也可以通过API文档中的调试功能进行调用。

3.签名机制

为保证API的安全调用，在调用API时阿里云会对每个API请求通过签名（Signature）进行身份验证。无论使用HTTP还是HTTPS协议提交请求，都需要在请求中包含签名信息。

概述

RPC API要按如下格式在API请求的Query中增加签名（Signature）：

```
https://Endpoint/?SignatureVersion=1.0&SignatureMethod=HMAC-SHA1&Signature=CT9X0VtwR86fNWSnsc6v8YGOjuE%3D&SignatureNonce=3ee8c1b8-83d3-44af-a94f-4e0ad82fd6cf
```

其中：

- **SignatureMethod**：签名方式，目前支持HMAC-SHA1。
- **SignatureVersion**：签名算法版本，目前版本是 1.0。
- **SignatureNonce**：唯一随机数，用于防止网络重放攻击。用户在不同请求间要使用不同的随机数值，建议使用通用唯一识别码（Universally Unique Identifier, UUID）。
- **Signature**: 使用AccessKey Secret对请求进行对称加密后生成的签名。

签名算法遵循RFC 2104 HMAC-SHA1规范，使用AccessSecret对编码、排序后的整个请求串计算HMAC值作为签名。签名的元素是请求自身的一些参数，由于每个API请求内容不同，所以签名的结果也不尽相同。可参考本文的操作步骤，计算签名值。

```
Signature = Base64( HMAC-SHA1( AccessSecret, UTF-8-Encoding-Of(StringToSign)) )
```

步骤一：构建待签名字符串

1. 使用请求参数构造规范化的请求字符串（Canonicalized Query String）。
    i. 按照参数名称的字典顺序对请求中所有的请求参数（包括公共请求参数和接口的自定义参数，但不包括公共请求参数中的Signature参数）进行排序。

> **说明** 当使用GET方法提交请求时，这些参数就是请求URI中的参数部分，即URI中“?”之后由“&”连接的部分。

ii. 对排序之后的请求参数的名称和值分别用UTF-8字符集进行URL编码。编码规则请参考下表：

字符	编码方式
A-Z、a-z和0-9以及“-”、“_”、“.”和“~”	不编码。
其它字符	编码成 `%XY` 的格式，其中 `XY` 是字符对应ASCII码的16进制表示。比如英文的双引号（”）对应的编码为 `%22` 。
扩展的UTF-8字符	编码成 `%XY%ZA...` 的格式。

英文空格	编码成 `%20` ，而不是加号（+）。
	该编码方式和一般采用的 `application/x-www-form-urlencoded` MIME格式编码算法（例如Java标准库中的 `java.net.URLEncoder` 的实现）存在区别。编码时可以先用标准库的方式进行编码，然后把编码后的字符串中的加号（+）替换成 `%20` ，星号（*）替换成 `%2A` ， `%7E` 替换回波浪号（~），即可得到上述规则描述的编码字符串。本算法可以用下面的 `percentEncode` 方法来实现：
	`private static final String ENCODING = "UTF-8";`
	`private static String percentEncode(String value) throws UnsupportedEncodingException`
	`{`
	`return value != null ? URLEncoder.encode(value, ENCODING).replace("+", "%20").replace("*", "%2A").replace("%7E", "~") : null;`
	`}`

iii. 将编码后的参数名称和值用英文等号（=）进行连接。

iv. 将等号连接得到的参数组合按步骤 i 排好的顺序依次使用“&”符号连接，即得到规范化请求字符串。

2. 将第一步构造的规范化字符串按照下面的规则构造成待签名的字符串。

```
StringToSign=
    HTTPMethod + “&” +
    percentEncode(“/”) + ” &” +
    percentEncode(CanonicalizedQueryString)
```

其中：

- HTTPMethod是提交请求用的HTTP方法，比如GET。
- percentEncode(“/”)是按照步骤1.1中描述的URL编码规则对字符 “/” 进行编码得到的值，即 `%2F` 。
- percentEncode(CanonicalizedQueryString)是对步骤1中构造的规范化请求字符串按步骤1.2中描述的URL编码规则编码后得到的字符串。

步骤二：计算签名值

1. 按照RFC2104的定义，计算待签名字符串（StringToSign）的HMAC值。

   > 说明 计算签名时使用的Key就是您持有的AccessKey Secret并加上一个“&”字符（ASCII:38），使用的哈希算法是SHA1。

2. 按照Base64编码规则把上面的HMAC值编码成字符串，即得到签名值（Signature）。
3. 将得到的签名值作为Signature参数添加到请求参数中。

   > 说明 得到的签名值在作为最后的请求参数值提交时要和其它参数一样，按照RFC3986的规则进行URL编码。

示例

以CreateLedger API为例，假设使用的AccessKey ID为 `testid` ， AccessKey Secret 为 `testsecret` 。签名前的请求URL如下：

```
http://ledgerdb.aliyuncs.com/?Action=CreateLedger&Timestamp=2018-12-23T12:46:24Z&Format=XML&AccessKeyId=testid&SignatureMethod=HMAC-SHA1&SignatureNonce=3ee8c1b8-83d3-44af-a94f-4e0ad82fd6cf&Version=2019-11-22&SignatureVersion=1.0
```

使用 `testsecret&` ，计算得到的签名值是：

```
OLeaidS1JvxuMvnyHOwuJ+uX5qY=
```

最后将签名作为Signature参数加入到URL请求中，最后得到的URL为：

```
http://ledgerdb.aliyuncs.com/?Action=CreateLedger&Timestamp=2018-12-23T12:46:24Z&Format=XML&AccessKeyId=testid&SignatureMethod=HMAC-SHA1&SignatureNonce=3ee8c1b8-83d3-44af-a94f-4e0ad82fd6cf&Version=2019-11-22&SignatureVersion=1.0&Signature=VyBL52idtt+oImX0NZC+2ngk15Q=
```

4.公共参数

本文介绍可信账本数据库产品每个接口都需要使用的请求参数和返回参数。

公共请求参数

公共请求参数表

名称	类型	是否必须	描述
Format	String	否	返回消息的格式。 取值：JSON（默认值）| XML
Version	String	是	API版本号，使用YYYY-MM-DD日期格式。取值：2019-11-22
AccessKeyId	String	是	访问服务使用的密钥ID。
Signature	String	是	签名结果串。
SignatureMethod	String	是	签名方式，取值：HMAC-SHA1
Timestamp	String	是	请求的时间戳，为日期格式。使用UTC时间按照ISO8601标准，格式为YYYY-MM-DDThh:mm:ssZ。 例如，北京时间2013年1月10日20点0分0秒，表示为2013-01-10T12:00:00Z。
SignatureVersion	String	是	签名算法版本，取值：1.0
SignatureNonce	String	是	唯一随机数，用于防止网络重放攻击。在不同请求间要使用不同的随机数值。
ResourceOwnerAccount	String	否	本次API请求访问到的资源拥有者账户，即登录用户名。

示例

```
http://ledgerdb.aliyuncs.com/?Action=CreateLedger
&TimeStamp=2019-11-22T10%3A33%3A56Z
&Format=xml
&AccessKeyId=testid
&SignatureMethod=Hmac-SHA1
&SignatureNonce=NwDAxvLU6tFE0DVb
&Version=2019-11-22
&SignatureVersion=1.0
&Signature=Signature
```

公共返回参数

API返回结果采用统一格式，调用成功返回的数据格式有XML和JSON两种，可以在发送请求时指定返回的数据格式，默认为JSON格式。每次接口调用，无论成功与否，系统都会返回一个唯一识别码RequestId。

- 返回 `2xx` HTTP状态码表示调用成功。
- 返回 `4xx` 或 `5xx` HTTP状态码表示调用失败。

公共返回参数示例如下：

- XML格式

```
<?xml version="1.0" encoding="utf-8"?>
  <!—结果的根结点-->
  <接口名称+Response>
    <!—返回请求标签-->
    <RequestId>4C467B38-3910-447D-87BC-AC049166F216</RequestId>
    <!—返回结果数据-->
  </接口名称+Response>
```

- JSON格式

```
{
  "RequestId":"4C467B38-3910-447D-87BC-AC049166F216",
  /*返回结果数据*/
}
```

5.获取AccessKey

您可以为阿里云主账号和子账号创建一个访问密钥（AccessKey）。在调用阿里云API时您需要使用AccessKey完成身份验证。

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 主账号Accesskey泄露会威胁您所有资源的安全。建议使用子账号（RAM用户）Accesskey进行操作，可以有效降低Accesskey泄露的风险。

操作步骤

1. 以主账号登录阿里云管理控制台。
2. 将鼠标置于页面右上方的账号图标，单击accesskeys。
3. 在安全提示页面，选择获取主账号还是子账号的Accesskey。

4. 获取账号Accesskey。
   - 获取主账号AccessKey：
     a. 单击继续使用AccessKey。
     b. 在安全信息管理页面，单击创建AccessKey。
     c. 在手机验证页面，获取验证码，完成手机验证，单击确定。
     d. 在新建用户AccessKey页面，展开AccessKey详情，查看AccessKeyId和AccessKeySecret。可以单击保存AK信息，下载AccessKey信息。
   - 获取子账号AccessKey：
     a. 单击开始使用子用户AccessKey。
     b. 如果未创建RAM用户，在系统跳转的RAM访问控制台的新建用户页面，创建RAM用户。如果是获取已有RAM用户的Accesskey，则跳过此步骤。
     c. 在RAM访问控制台的左侧导航栏，选择人员管理>用户，搜索需要获取AccessKey的用户。

d. 单击用户登录名称，在用户详情页**认证管理**页签下的**用户AccessKey**区域，单击**创建新的AccessKey**。

e. 在**手机验证**页面，获取验证码，完成手机验证，单击**确定**。

f. 在**创建AccessKey**页面，查看AccessKeyId和AccessKeySecret。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。

6.子账号授权

在使用子账号（RAM账号）调用可信账本数据库API前，需要主账号通过创建授权策略对RAM账号进行授权。在授权策略中，使用资源描述符（Alibaba Cloud Resource Name，ARN）指定授权资源。

资源授权

默认子账号没有权限通过调用阿里云API去创建、修改云资源。使用子账号调用API时，您需要先创建一个授权策略，然后将这个授权策略关联给对应的子账号完成资源授权。

在创建授权策略时，您可以通过ARN（Alibaba Cloud Resource Name）指定要授权的资源。ARN是阿里云为每个资源定义的一个全局的阿里云资源名称。

ARN格式如下：

```
acs:service-name:region:account-id:resource-relative-id
```

其中：

- **acs**：Alibaba Cloud Service的首字母缩写，表示阿里云的公共云平台。
- **service-name**：阿里云云服务的名称，如ecs, oss, slb等。
- **region**：地域信息。如果不支持该项，可以使用通配符星号（*）来代替。
- **account-id**：账号ID，例如1234567890123456。
- **resource-relative-id**：资源描述。

示例如下：

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "ecs:[ECS RAM Action]",
        "ecs:DescribeInstances"
      ],
      "Resource": [
        "[ECS RAM Action Resource]",
        "acs:ecs:$regionid:15619224785*****:instance/i-bp1bzvz55uz27hf*****"
      ],
      "Effect": "Allow"
    }
  ]
}
```

可授权的可信账本数据库资源类型

在进行RAM子账号授权时，可信账本数据库资源的描述方式如下：

资源类型	授权策略中的资源描述方法
Ledger	acs:ledgerdb:$regionid:$accountid:ledger/$ledgerid
	acs:ledgerdb:$regionid:$accountid:ledger/*
Member	acs:ledgerdb:$regionid:$accountid:member/$memberid
	acs:ledgerdb:$regionid:$accountid:member/*

可授权的可信账本数据库接口

下表列举了可授权的API及其描述方式：

API	资源描述
CreateLedger	acs:ledgerdb:$regionid:$accountid:ledger/*
DeleteLedger	acs:ledgerdb:$regionid:$accountid:ledger/$ledgerid
DescribeLedger	acs:ledgerdb:$regionid:$accountid:ledger/$ledgerid
DescribeLedgers	acs:ledgerdb:$regionid:$accountid:ledger/*
ModifyLedgerAttribute	acs:ledgerdb:$regionid:$accountid:ledger/$ledgerid
AcceptMember	acs:ledgerdb:$regionid:$accountid:member/*
CreateMember	acs:ledgerdb:$regionid:$accountid:member/*
DeleteMember	acs:ledgerdb:$regionid:$accountid:member/$memberid
DisableMember	acs:ledgerdb:$regionid:$accountid:member/$memberid
EnableMember	acs:ledgerdb:$regionid:$accountid:member/$memberid
GetMember	acs:ledgerdb:$regionid:$accountid:member/$memberid
GetMemberKey	acs:ledgerdb:$regionid:$accountid:member/*
InviteMember	acs:ledgerdb:$regionid:$accountid:member/*
ListMembers	acs:ledgerdb:$regionid:$accountid:member/*

API	资源描述
ModifyMemberACLs	acs:ledgerdb:$regionid:$accountid:member/$memberid
ModifyMemberKey	acs:ledgerdb:$regionid:$accountid:member/*

7.账本实例API

7.1. DescribeLedger

调用DescribeLedger查询一个账本实例的详细信息。

调试

您可以在OpenAPI Explorer中直接运行该接口，免去您计算签名的困扰。运行成功后，OpenAPI Explorer可以自动生成SDK代码示例。

请求参数

名称	类型	是否必选	示例值	描述
Action	String	是	DescribeLedger	系统规定参数。取值：DescribeLedger。
LedgerId	String	是	l-c8cc7be3eea542axxxxxxxxxx	要查询的账本实例ID。

返回数据

名称	类型	示例值	描述
Ledger	Struct		账本详细信息。
CreateTime	String	1587515659345	账本创建时间。自1970年1月1日（00:00:00 GMT）以来的毫秒数。
JournalCount	Long	100	账本总记录数。
LastTimeAnchor	Struct		最新时间锚点信息。
JournalId	String	50	时间锚点所在的记录ID。
LedgerDigest	String	f1e17070c100bd86538550252d94540481f7b81b852c859c906a44e416efdb86	账本摘要。
LedgerDigestType	String	SHA256	账本摘要类型。取值：SHA256。

名称	类型	示例值	描述
LedgerVersion	String	50	账本版本。
Proof	String	{"hashInHex":"aecdd0b65b7c0bd50bac1ac5d3cfa06ed78ce7cb270b3ca1110b2a6549242d49","sequence":52516}	时间证据。
TimeStamp	String	1587095881000	时间戳。自1970年1月1日（00:00:00 GMT）以来的毫秒数。
LedgerDescription	String	这是我的第一个账本。	账本描述。
LedgerId	String	l-c8cc7be3eea542axxxxxxxxxx	账本实例ID。
LedgerName	String	账本1	账本名称。
LedgerStatus	String	NORMAL	账本状态。取值：INVALID、NORMAL、DELETING、DELETED。INVALID状态表示账本还处于创建中，NORMAL状态表示该账本可正常可用，DELETING状态表示该账本处于删除中，DELETED状态表示该账本已删除。
LedgerType	String	JOURNAL	账本类型。取值：KV、JOURNAL。KV类型的账本只能执行KV相关的数据操作，JOURNAL类型的账本只能执行JOURNAL相关的数据操作。
MemberCount	Long	10	账本成员数目。
OwnerAliUid	String	122435xxxxx	账本的创建者的阿里云账户ID。
RegionId	String	cn-hangzhou	账本所在的Region。
TimeAnchorCount	Long	10	账本中时间锚点数量。
UpdateTime	String	1587515659345	账本的最后更新时间。返回自1970年1月1日（00:00:00 GMT）以来的毫秒数。

名称	类型	示例值	描述
ZoneId	String	cn-hangzhou-h	账本实例所在的可用区ID。
RequestId	String	89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C	请求ID。

示例

请求示例

```
http(s)://[Endpoint]/?Action=DescribeLedger
&LedgerId=l-c8cc7be3eea542axxxxxxxxxx
&<公共请求参数>
```

正常返回示例

`XML` 格式

```
<RequestId>89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C</RequestId>
<Ledger>
  <JournalCount>100</JournalCount>
  <ZoneId>cn-hangzhou-h</ZoneId>
  <TimeAnchorCount>10</TimeAnchorCount>
  <CreateTime>1587515659345</CreateTime>
  <MemberCount>10</MemberCount>
  <LedgerId>l-c8cc7be3eea542axxxxxxxxxx</LedgerId>
  <LedgerDescription>这是我的第一个账本。</LedgerDescription>
  <LastTimeAnchor>
    <LedgerDigest>f1e17070c100bd86538550252d94540481f7b81b852c859c906a44e416efdb86</LedgerDigest>
    <JournalId>50</JournalId>
    <LedgerDigestType>SHA256</LedgerDigestType>
    <Proof>{"hashInHex":"aecdd0b65b7c0bd50bac1ac5d3cfa06ed78ce7cb270b3ca1110b2a6549242d49","sequence":52516}</Proof>
    <LedgerVersion>50</LedgerVersion>
    <TimeStamp>1587095881000</TimeStamp>
  </LastTimeAnchor>
  <OwnerAliUid>122435xxxxx</OwnerAliUid>
  <LedgerType>JOURNAL</LedgerType>
  <UpdateTime>1587515659345</UpdateTime>
  <RegionId>cn-hangzhou</RegionId>
  <LedgerStatus>NORMAL</LedgerStatus>
  <LedgerName>账本1</LedgerName>
</Ledger>
```

`JSON` 格式

```
{
  "RequestId": "89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C",
  "Ledger": {
    "JournalCount": 100,
    "ZoneId": "cn-hangzhou-h",
    "TimeAnchorCount": 10,
    "CreateTime": 1587515659345,
    "MemberCount": 10,
    "LedgerId": "l-c8cc7be3eea542axxxxxxxxxx",
    "LedgerDescription": "这是我的第一个账本。",
    "LastTimeAnchor": {
      "LedgerDigest": "f1e17070c100bd86538550252d94540481f7b81b852c859c906a44e416efdb86",
      "JournalId": 50,
      "LedgerDigestType": "SHA256",
      "Proof": "{\"hashInHex\":\"aecdd0b65b7c0bd50bac1ac5d3cfa06ed78ce7cb270b3ca1110b2a6549242d49\",\"sequence\":52516}",
      "LedgerVersion": 50,
      "TimeStamp": 1587095881000
    },
    "OwnerAliUid": "122435xxxxx",
    "LedgerType": "JOURNAL",
    "UpdateTime": 1587515659345,
    "RegionId": "cn-hangzhou",
    "LedgerStatus": "NORMAL",
    "LedgerName": "账本1"
  }
}
```


错误码

HttpCode	错误码	错误信息	描述
400	InvalidParameter	The specified parameter %s is invalid.	参数无效
400	MissingParameter	You must specify the parameter %s.	参数缺失
404	ResourceNotFound	The specified resource %s does not exist.	资源不存在
403	UnauthorizedOperation	You are not authorized to perform this operation. %s	无操作权限

访问错误中心查看更多错误码。

访问错误中心查看更多错误码。

7.2. DescribeLedgers

调用DescribeLedgers获取账本实例列表。

调试

您可以在OpenAPI Explorer中直接运行该接口，免去您计算签名的困扰。运行成功后，OpenAPI Explorer可以自动生成SDK代码示例。

请求参数

名称	类型	是否必选	示例值	描述
Action	String	是	DescribeLedgers	系统规定参数。取值：DescribeLedgers。
NextToken	String	否	xxxx	用来标记当前开始读取的位置。置空表示从头开始。
MaxResults	Integer	否	10	本次读取的最大数据量，默认值为10。

返回数据

名称	类型	示例值	描述
Ledgers	Array of Ledger		账本详细信息。
CreateTime	String	1587515659345	账本创建时间。自1970年1月1日（00:00:00 GMT）以来的毫秒数。
JournalCount	Long	100	账本总记录数。
LastTimeAnchor	Struct		最新时间锚点信息。
JournalId	String	50	时间锚点所在的记录ID。
LedgerDigest	String	f1e17070c100bd86538550252d94540481f7b81b852c859c906a44e416efdb86	账本摘要。

名称	类型	示例值	描述
LedgerDigestType	String	SHA256	账本摘要类型。取值：SHA256。
LedgerVersion	String	50	账本版本。
Proof	String	{"hashInHex":"aecdd0b65b7c0bd50bac1ac5d3cfa06ed78ce7cb270b3ca1110b2a6549242d49","sequence":52516}	时间证据。
TimeStamp	String	1587095881000	时间戳。自1970年1月1日（00:00:00 GMT）以来的毫秒数。
LedgerDescription	String	这是我的第一个账本。	账本描述。
LedgerId	String	l-c8cc7be3eea542axxxxxxxxxx	账本实例ID。
LedgerName	String	账本1	账本名称。
LedgerStatus	String	NORMAL	账本状态。取值：INVALID、NORMAL、DELETING、DELETED。INVALID状态表示账本还处于创建中，NORMAL状态表示该账本可正常可用，DELETING状态表示该账本处于删除中，DELETED状态表示该账本已删除。
LedgerType	String	JOURNAL	账本类型。取值：KV、JOURNAL。KV类型的账本只能执行KV相关的数据操作，JOURNAL类型的账本只能执行JOURNAL相关的数据操作。
MemberCount	Long	10	账本成员数目。
OwnerAliUid	String	122435xxxxx	账本创建者的阿里云账户ID。
RegionId	String	cn-hangzhou	账本所在的Region。
StorageClass	String	SSD	存储介质类型

名称	类型	示例值	描述
TimeAnchorCount	Long	10	账本中时间锚点数量。
UpdateTime	String	1587515659345	账本的最后更新时间。自1970年1月1日（00:00:00 GMT）以来的毫秒数。
ZoneId	String	cn-hangzhou-h	账本实例所在的可用区ID。
MaxResults	Integer	10	本次读取的最大数据量。
NextToken	String	xxxx	用来表示当前调用返回读取到的位置。空代表数据已经读取完毕。
RequestId	String	89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C	请求ID。

示例

请求示例

```
http(s)://[Endpoint]/?Action=DescribeLedgers
&<公共请求参数>
```

正常返回示例

`XML` 格式

```
<RequestId>89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C</RequestId>
<NextToken>xxxx</NextToken>
<MaxResults>10</MaxResults>
<Ledgers>
  <JournalCount>100</JournalCount>
  <ZoneId>cn-hangzhou-h</ZoneId>
  <TimeAnchorCount>10</TimeAnchorCount>
  <CreateTime>1587515659345</CreateTime>
  <MemberCount>10</MemberCount>
  <LedgerId>l-c8cc7be3eea542axxxxxxxxxx</LedgerId>
  <LedgerDescription>这是我的第一个账本。</LedgerDescription>
  <OwnerAliUid>122435xxxxx</OwnerAliUid>
  <LedgerType>JOURNAL</LedgerType>
  <UpdateTime>1587515659345</UpdateTime>
  <RegionId>cn-hangzhou</RegionId>
  <LedgerStatus>NORMAL</LedgerStatus>
  <LedgerName>账本1</LedgerName>
  <StorageClass>SSD</StorageClass>
  <LastTimeAnchor>
    <LedgerDigest>f1e17070c100bd86538550252d94540481f7b81b852c859c906a44e416efdb86</Ledger
Digest>
    <JournalId>50</JournalId>
    <LedgerDigestType>SHA256</LedgerDigestType>
    <Proof>{"hashInHex":"aecdd0b65b7c0bd50bac1ac5d3cfa06ed78ce7cb270b3ca1110b2a6549242d49","
sequence":52516}</Proof>
    <LedgerVersion>50</LedgerVersion>
    <TimeStamp>1587095881000</TimeStamp>
  </LastTimeAnchor>
</Ledgers>
```

`JSON` 格式

```
{
  "RequestId": "89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C",
  "NextToken": "xxxx",
  "MaxResults": 10,
  "Ledgers": {
    "JournalCount": 100,
    "ZoneId": "cn-hangzhou-h",
    "TimeAnchorCount": 10,
    "CreateTime": 1587515659345,
    "MemberCount": 10,
    "LedgerId": "l-c8cc7be3eea542axxxxxxxxxx",
    "LedgerDescription": "这是我的第一个账本。",
    "OwnerAliUid": "122435xxxxx",
    "LedgerType": "JOURNAL",
    "UpdateTime": 1587515659345,
    "RegionId": "cn-hangzhou",
    "LedgerStatus": "NORMAL",
    "LedgerName": "账本1",
    "StorageClass": "SSD",
    "LastTimeAnchor": {
      "LedgerDigest": "f1e17070c100bd86538550252d94540481f7b81b852c859c906a44e416efdb86",
      "JournalId": 50,
      "LedgerDigestType": "SHA256",
      "Proof": "{\"hashInHex\":\"aecdd0b65b7c0bd50bac1ac5d3cfa06ed78ce7cb270b3ca1110b2a6549242d49\",\"sequence\":52516}",
      "LedgerVersion": 50,
      "TimeStamp": 1587095881000
    }
  }
}
```


错误码

HttpCode	错误码	错误信息	描述
400	InvalidParameter	The specified parameter %s is invalid.	参数无效
400	MissingParameter	You must specify the parameter %s.	参数缺失

HttpCode	错误码	错误信息	描述
403	UnauthorizedOperation	You are not authorized to perform this operation. %s	无操作权限

访问错误中心查看更多错误码。

访问错误中心查看更多错误码。

7.3. ModifyLedgerAttribute

调用ModifyLedgerAttribute修改账本属性

调试

您可以在OpenAPI Explorer中直接运行该接口，免去您计算签名的困扰。运行成功后，OpenAPI Explorer可以自动生成SDK代码示例。

请求参数

名称	类型	是否必选	示例值	描述
Action	String	是	ModifyLedgerAttribute	系统规定参数。取值：ModifyLedgerAttribute。
LedgerId	String	是	l-c8cc7be3eea542axxxxxxxxxx	账本实例ID。
LedgerName	String	否	firstledger	账本名称。
LedgerDescription	String	否	我的第一个账本	账本描述。

返回数据

名称	类型	示例值	描述
RequestId	String	89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C	请求ID。

示例

请求示例

```
http(s)://[Endpoint]/?Action=ModifyLedgerAttribute
&LedgerId=l-c8cc7be3eea542axxxxxxxxxx
&<公共请求参数>
```

正常返回示例

XML 格式

```
<RequestId>89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C</RequestId>
```

JSON 格式

```
{
  "RequestId": "89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C"
}
```

错误码

HttpCode	错误码	错误信息	描述
400	InvalidParameter	The specified parameter %s is invalid.	参数无效
400	MissingParameter	You must specify the parameter %s.	参数缺失
404	ResourceNotFound	The specified resource %s does not exist.	资源不存在
403	UnauthorizedOperation	You are not authorized to perform this operation. %s	无操作权限

访问错误中心查看更多错误码。

访问错误中心查看更多错误码。

7.4. DeleteLedger

调用DeleteLedger删除账本实例。

调试

您可以在OpenAPI Explorer中直接运行该接口，免去您计算签名的困扰。运行成功后，OpenAPI Explorer可以自动生成SDK代码示例。

请求参数

名称	类型	是否必选	示例值	描述
Action	String	是	DeleteLedger	系统规定参数。取值：DeleteLedger。
LedgerId	String	是	l-c8cc7be3eea542axxxxxxxxxx	要删除的账本实例ID。

返回数据

名称	类型	示例值	描述
RequestId	String	473469C7-AA6F-4DC5-B3DB-A3DC0DE3C83E	请求ID。

示例

请求示例

```
http(s)://[Endpoint]/?Action=DeleteLedger
&LedgerId=l-c8cc7be3eea542axxxxxxxxxx
&<公共请求参数>
```

正常返回示例

`XML` 格式

```
<RequestId>473469C7-AA6F-4DC5-B3DB-A3DC0DE3C83E</RequestId>
```

`JSON` 格式

```
{
  "RequestId": "473469C7-AA6F-4DC5-B3DB-A3DC0DE3C83E"
}
```

错误码

HttpCode	错误码	错误信息	描述
400	InvalidParameter	The specified parameter %s is invalid.	参数无效
400	MissingParameter	You must specify the parameter %s.	参数缺失
404	ResourceNotFound	The specified resource %s does not exist.	资源不存在
403	UnauthorizedOperation	You are not authorized to perform this operation. %s	无操作权限

访问错误中心查看更多错误码。

访问错误中心查看更多错误码。

8.账本成员API

8.1. GetMember

调用GetMember获取某个成员的信息。

调试

您可以在OpenAPI Explorer中直接运行该接口，免去您计算签名的困扰。运行成功后，OpenAPI Explorer可以自动生成SDK代码示例。

请求参数

名称	类型	是否必选	示例值	描述
Action	String	是	GetMember	系统规定参数。取值：GetMember。
LedgerId	String	是	l-c8cc7be3eea542axxxxxxxxxx	操作的账本实例ID。
MemberId	String	否	m-04389ec1075443bcxxxxxxxxxx	要获取的成员ID。如果不指定，则获取当前阿里云账号对应的成员信息。

返回数据

名称	类型	示例值	描述
AliUid	String	13224xxxxxxxx	成员的阿里云账号UID。
CreateTime	Long	1587515659345	成员创建时间。自1970年1月1日（00:00:00 GMT）以来的毫秒数。
KeyType	String	SECP256K1	成员的公钥类型。取值：SECP256K1。
LedgerId	String	l-c8cc7be3eea542axxxxxxxxxx	成员所在的账本实例ID。
MemberId	String	m-04389ec1075443bcxxxxxxxxxx	成员ID。

名称	类型	示例值	描述
PublicKey	String	04d45eb3bcbd2a47aeb5de87b73fd6a324bb7b57a081fd474a82d8dc7e830d42868aeb15a9fa02ac09205b68ecfc78e8e67c70c67736aa0e867a9a7011b83fab4b	成员公钥，用于身份认证。
RequestId	String	89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C	请求ID。
Role	String	READER	成员角色。取值：ADMIN、WRITER、READER。ADMIN角色拥有账本所有操作权限，WRITER角色可以写入或者读取账本记录，READER角色只能读取账本记录。
State	String	ENABLED	成员状态。取值：ENABLED、DISABLED。处于ENABLED状态的成员可以正常使用该账本，处于DISABLED状态的成员无法使用该账本。
UpdateTime	Long	1587515659345	成员更新时间。自1970年1月1日（00:00:00 GMT）以来的毫秒数。

示例

请求示例

```
http(s)://[Endpoint]/?Action=GetMember
&LedgerId=l-c8cc7be3eea542axxxxxxxxxx
&<公共请求参数>
```

正常返回示例

`XML` 格式

```
<Role>READER</Role>
<MemberId>m-04389ec1075443bcxxxxxxxxxx</MemberId>
<RequestId>89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C</RequestId>
<KeyType>SECP256K1</KeyType>
<PublicKey>04d45eb3bcbd2a47aeb5de87b73fd6a324bb7b57a081fd474a82d8dc7e830d42868aeb15a9fa02
ac09205b68ecfc78e8e67c70c67736aa0e867a9a7011b83fab4b</PublicKey>
<State>ENABLED</State>
<CreateTime>1587515659345</CreateTime>
<UpdateTime>1587515659345</UpdateTime>
<LedgerId>l-c8cc7be3eea542axxxxxxxxxx</LedgerId>
<AliUid>13224xxxxxxxx</AliUid>
```

JSON 格式

```
{
  "Role": "READER",
  "MemberId": "m-04389ec1075443bcxxxxxxxxxx",
  "RequestId": "89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C",
  "KeyType": "SECP256K1",
  "PublicKey": "04d45eb3bcbd2a47aeb5de87b73fd6a324bb7b57a081fd474a82d8dc7e830d42868aeb15a9f
a02ac09205b68ecfc78e8e67c70c67736aa0e867a9a7011b83fab4b",
  "State": "ENABLED",
  "CreateTime": 1587515659345,
  "UpdateTime": 1587515659345,
  "LedgerId": "l-c8cc7be3eea542axxxxxxxxxx",
  "AliUid": "13224xxxxxxxx"
}
```

错误码

HttpCode	错误码	错误信息	描述
400	InvalidParameter	The specified parameter %s is invalid.	参数无效
400	MissingParameter	You must specify the parameter %s.	参数缺失
404	ResourceNotFound	The specified resource %s does not exist.	资源不存在
403	UnauthorizedOperation	You are not authorized to perform this operation. %s	无操作权限

访问错误中心查看更多错误码。

访问错误中心查看更多错误码。

8.2. ListMembers

调用ListMembers获取账本的成员列表。

调试

您可以在OpenAPI Explorer中直接运行该接口，免去您计算签名的困扰。运行成功后，OpenAPI Explorer可以自动生成SDK代码示例。

请求参数

名称	类型	是否必选	示例值	描述
Action	String	是	ListMembers	系统规定参数。取值：ListMembers。
LedgerId	String	是	l-c8cc7be3eea542axxxxxxxxxx	账本实例ID。
NextToken	String	否	xxxx	用来标记当前开始读取的位置。置空表示从头开始。
MaxResults	Integer	否	100	本次读取的最大数据量，默认值为1024。

返回数据

名称	类型	示例值	描述
MaxResults	Integer	100	本次读取的最大数据量。
Members	Array		成员列表。
AliUid	String	12256464xxx	成员的阿里云账号ID。
CreateTime	Long	1587515659345	成员创建时间。
KeyType	String	SECP256K1	成员的公钥类型。当前只支持SECP256K1。
LedgerId	String	l-c8cc7be3eea542axxxxxxxxx	成员所在的账本实例ID。

名称	类型	示例值	描述
MemberId	String	m-04389ec1075443bcxxxxxxxxx	成员ID。
PublicKey	String	04d45eb3bcbd2a47aeb5de87b73fd6a324bb7b57a081fd474a82d8dc7e830d42868aeb15a9fa02ac09205b68ecfc78e8e67c70c67736aa0e867a9a7011b83fab4b	成员公钥。十六进制表示。
Role	String	ADMIN	成员角色。从ADMIN、WRITER、READER中取值。ADMIN角色拥有账本所有操作权限，WRITER角色可以写入或者读取账本记录，READER角色只能读取账本记录。
State	String	ENABLED	成员状态。从ENABLED、DISABLED中取值。ENABLED状态的成员可以正常使用该账本，DISABLED状态的成员无法使用该账本。
UpdateTime	Long	1587515659642	成员更新时间。
NextToken	String	xxxx	用来表示当前调用返回读取到的位置。空代表数据已经读取完毕。
RequestId	String	89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C	请求ID。

示例

请求示例

```
http(s)://[Endpoint]/?Action=ListMembers
&LedgerId=l-c8cc7be3eea542axxxxxxxxxx
&<公共请求参数>
```

正常返回示例

`XML` 格式

```
<RequestId>89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C</RequestId>
<NextToken>xxxx</NextToken>
<MaxResults>100</MaxResults>
<Members>
  <Role>ADMIN</Role>
  <MemberId>m-04389ec1075443bcxxxxxxxxxx</MemberId>
  <KeyType>SECP256K1</KeyType>
  <PublicKey>04d45eb3bcbd2a47aeb5de87b73fd6a324bb7b57a081fd474a82d8dc7e830d42868aeb15a9fa
02ac09205b68ecfc78e8e67c70c67736aa0e867a9a7011b83fab4b</PublicKey>
  <State>ENABLED</State>
  <CreateTime>1587515659345</CreateTime>
  <UpdateTime>1587515659642</UpdateTime>
  <LedgerId>l-c8cc7be3eea542axxxxxxxxxx</LedgerId>
  <AliUid>12256464xxx</AliUid>
</Members>
```

`JSON` 格式

```
{
  "RequestId": "89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C",
  "NextToken": "xxxx",
  "MaxResults": 100,
  "Members": {
    "Role": "ADMIN",
    "MemberId": "m-04389ec1075443bcxxxxxxxxxx",
    "KeyType": "SECP256K1",
    "PublicKey": "04d45eb3bcbd2a47aeb5de87b73fd6a324bb7b57a081fd474a82d8dc7e830d42868aeb15
a9fa02ac09205b68ecfc78e8e67c70c67736aa0e867a9a7011b83fab4b",
    "State": "ENABLED",
    "CreateTime": 1587515659345,
    "UpdateTime": 1587515659642,
    "LedgerId": "l-c8cc7be3eea542axxxxxxxxxx",
    "AliUid": "12256464xxx"
  }
}
```

错误码

HttpCode	错误码	错误信息	描述

HttpCode	错误码	错误信息	描述
400	InvalidParameter	The specified parameter %s is invalid.	参数无效
400	MissingParameter	You must specify the parameter %s.	参数缺失
404	ResourceNotFound	The specified resource %s does not exist.	资源不存在
403	UnauthorizedOperation	You are not authorized to perform this operation. %s	无操作权限

访问错误中心查看更多错误码。

访问错误中心查看更多错误码。

8.3. AcceptMember

调用AcceptMember接受某个账本的邀请，成为其成员。

调试

您可以在OpenAPI Explorer中直接运行该接口，免去您计算签名的困扰。运行成功后，OpenAPI Explorer可以自动生成SDK代码示例。

请求参数

名称	类型	是否必选	示例值	描述
Action	String	是	AcceptMember	系统规定参数。取值：AcceptMember。
LedgerId	String	是	l-c8cc7be3eea542axxxxxxxxxx	账本实例ID。
KeyType	String	否	SECP256K1	用于身份认证的公钥类型。取值：SECP256K1。SECP256K1采用Secp256k1签名算法。
PublicKey	String	否	04d45eb3bcbd2a47aeb5de87b73fd6a324bb7b57a081fd474a82d8dc7e830d42868aeb15a9fa02ac09205b68ecfc78e8e67c70c67736aa0e867a9a7011b83fab4b	此Member在Ledger中的公钥内容。

返回数据

名称	类型	示例值	描述
MemberId	String	m-04389ec1075443bcxxxxxxxxxx	生成的成员ID。
RequestId	String	89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C	请求ID。

示例

请求示例

```
http(s)://[Endpoint]/?Action=AcceptMember
&LedgerId=l-c8cc7be3eea542axxxxxxxxxx
&<公共请求参数>
```

正常返回示例

`XML` 格式

```
<MemberId>m-04389ec1075443bcxxxxxxxxxx</MemberId>
<RequestId>89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C</RequestId>
```

`JSON` 格式

```
{
  "MemberId": "m-04389ec1075443bcxxxxxxxxxx",
  "RequestId": "89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C"
}
```

错误码

HttpCode	错误码	错误信息	描述
400	InvalidParameter	The specified parameter %s is invalid.	参数无效
400	MissingParameter	You must specify the parameter %s.	参数缺失
404	ResourceNotFound	The specified resource %s does not exist.	资源不存在
409	ResourceAlreadyExist	The resource %s already exists.	资源已经存在

HttpCode	错误码	错误信息	描述
403	UnauthorizedOperation	You are not authorized to perform this operation. %s	无操作权限

访问错误中心查看更多错误码。

访问错误中心查看更多错误码。

8.4. InviteMembers

调用InviteMembers邀请其他阿里云账号加入到某个账本中。

调试

您可以在OpenAPI Explorer中直接运行该接口，免去您计算签名的困扰。运行成功后，OpenAPI Explorer可以自动生成SDK代码示例。

请求参数

名称	类型	是否必选	示例值	描述
Action	String	是	InviteMembers	系统规定参数。取值：InviteMembers。
AliUids	String	是	1224xxxx,46573xxxxx	阿里云账号UID列表。多个云账号使用“,”分割。
LedgerId	String	是	l-c8cc7be3eea542axxxxxxxxxx	账本实例ID。

返回数据

名称	类型	示例值	描述
RequestId	String	89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C	请求ID。

示例

请求示例

```
http(s)://[Endpoint]/?Action=InviteMembers
&AliUids=1224xxxx,46573xxxxx
&LedgerId=l-c8cc7be3eea542axxxxxxxxxx
&<公共请求参数>
```

正常返回示例

XML 格式

```
<RequestId>89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C</RequestId>
```

JSON 格式

```
{"RequestId":"89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C"}
```

错误码

HttpCode	错误码	错误信息	描述
400	InvalidParameter	The specified parameter %s is invalid.	参数无效
400	MissingParameter	You must specify the parameter %s.	参数缺失
404	ResourceNotFound	The specified resource %s does not exist.	资源不存在
409	ResourceAlreadyExist	The resource %s already exists.	资源已经存在
403	UnauthorizedOperation	You are not authorized to perform this operation. %s	无操作权限

访问错误中心查看更多错误码。

访问错误中心查看更多错误码。

8.5. EnableMember

调用EnableMember恢复某个成员。

调试

您可以在OpenAPI Explorer中直接运行该接口，免去您计算签名的困扰。运行成功后，OpenAPI Explorer可以自动生成SDK代码示例。

请求参数

名称	类型	是否必选	示例值	描述
Action	String	是	EnableMember	系统规定参数。取值：EnableMember。
LedgerId	String	是	l-c8cc7be3eea542axxxxxxxxxx	操作的账本ID。

名称	类型	是否必选	示例值	描述
MemberId	String	是	m-04389ec1075443bcxxxxxxxxxx	被恢复的成员ID。

返回数据

名称	类型	示例值	描述
RequestId	String	89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C	请求ID。

示例

请求示例

```
http(s)://[Endpoint]/?Action=EnableMember
&LedgerId=l-c8cc7be3eea542axxxxxxxxxx
&MemberId=m-04389ec1075443bcxxxxxxxxxx
&<公共请求参数>
```

正常返回示例

`XML` 格式

```
<RequestId>89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C</RequestId>
```

`JSON` 格式

```
{"RequestId":"89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C"}
```

错误码

HttpCode	错误码	错误信息	描述
400	InvalidParameter	The specified parameter %s is invalid.	参数无效
400	MissingParameter	You must specify the parameter %s.	参数缺失
404	ResourceNotFound	The specified resource %s does not exist.	资源不存在
403	UnauthorizedOperation	You are not authorized to perform this operation. %s	无操作权限

访问错误中心查看更多错误码。

访问错误中心查看更多错误码。

8.6. DisableMember

调用DisableMember禁用某个成员。

调试

您可以在OpenAPI Explorer中直接运行该接口，免去您计算签名的困扰。运行成功后，OpenAPI Explorer可以自动生成SDK代码示例。

请求参数

名称	类型	是否必选	示例值	描述
Action	String	是	DisableMember	系统规定参数。取值：DisableMember。
LedgerId	String	是	l-c8cc7be3eea542axxxxxxxxxx	操作的账本实例ID。
MemberId	String	是	m-04389ec1075443bcxxxxxxxxxx	被禁用的成员ID。

返回数据

名称	类型	示例值	描述
RequestId	String	89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C	请求ID。

示例

请求示例

```
http(s)://[Endpoint]/?Action=DisableMember
&LedgerId=l-c8cc7be3eea542axxxxxxxxxx
&MemberId=m-04389ec1075443bcxxxxxxxxxx
&<公共请求参数>
```

正常返回示例

`XML` 格式

```
<RequestId>89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C</RequestId>
```

JSON 格式

```
{
  "RequestId": "89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C"
}
```

错误码

HttpCode	错误码	错误信息	描述
400	InvalidParameter	The specified parameter %s is invalid.	参数无效
400	MissingParameter	You must specify the parameter %s.	参数缺失
404	ResourceNotFound	The specified resource %s does not exist.	资源不存在
403	UnauthorizedOperation	You are not authorized to perform this operation. %s	无操作权限

访问错误中心查看更多错误码。

访问错误中心查看更多错误码。

8.7. ModifyMemberKey

调用ModifyMemberKey修改一个账本成员的公钥

调试

您可以在OpenAPI Explorer中直接运行该接口，免去您计算签名的困扰。运行成功后，OpenAPI Explorer可以自动生成SDK代码示例。

请求参数

名称	类型	是否必选	示例值	描述
Action	String	是	ModifyMemberKey	系统规定参数。取值：ModifyMemberKey。
KeyType	String	是	SECP256K1	公钥类型，当前仅支持SECP256K1
LedgerId	String	是	l-c8cc7be3eea542axxxxxxxxxx	账本实例ID

名称	类型	是否必选	示例值	描述
PublicKey	String	是	04d45eb3bcbd2a47aeb5de87b73fd6a324bb7b57a081fd474a82d8dc7e830d42868aeb15a9fa02ac09205b68ecfc78e8e67c70c67736aa0e867a9a7011b83fab4b	公钥十六进制编码字符串
MemberId	String	否	m-04389ec1075443bcxxxxxxxxxx	成员ID

返回数据

名称	类型	示例值	描述
RequestId	String	89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C	请求ID

示例

请求示例

```
http(s)://[Endpoint]/?Action=ModifyMemberKey
&KeyType=SECP256K1
&LedgerId=l-c8cc7be3eea542axxxxxxxxxx
&PublicKey=04d45eb3bcbd2a47aeb5de87b73fd6a324bb7b57a081fd474a82d8dc7e830d42868aeb15a9fa02ac09205b68ecfc78e8e67c70c67736aa0e867a9a7011b83fab4b
&<公共请求参数>
```

正常返回示例

`XML` 格式

```
<RequestId>89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C</RequestId>
```

`JSON` 格式

```
{
  "RequestId": "89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C"
}
```

错误码

HttpCode	错误码	错误信息	描述
400	InvalidParameter	The specified parameter %s is invalid.	参数无效
400	MissingParameter	You must specify the parameter %s.	参数缺失
404	ResourceNotFound	The specified resource %s does not exist.	资源不存在
403	UnauthorizedOperation	You are not authorized to perform this operation. %s	无操作权限

访问错误中心查看更多错误码。

访问错误中心查看更多错误码。

8.8. ModifyMemberACLs

调用ModifyMemberACLs修改成员操作权限

调试

您可以在OpenAPI Explorer中直接运行该接口，免去您计算签名的困扰。运行成功后，OpenAPI Explorer可以自动生成SDK代码示例。

请求参数

名称	类型	是否必选	示例值	描述
Action	String	是	ModifyMemberACLs	系统规定参数。取值：ModifyMemberACLs。
LedgerId	String	是	l-c8cc7be3eea542axxxxxxxxxx	账本实例ID。
MemberId	String	是	m-04389ec1075443bcxxxxxxxxxx	成员ID。
Role	String	是	WRITER	成员权限角色，目前有3种角色：ADMIN、WRITER、READER。ADMIN拥有账本所有操作权限，WRITER可以写入或者读取账本中的记录，READER只能读取账本记录。

返回数据

名称	类型	示例值	描述
RequestId	String	89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C	请求ID。

示例

请求示例

```
http(s)://[Endpoint]/?Action=ModifyMemberACLs
&LedgerId=l-c8cc7be3eea542axxxxxxxxxx
&MemberId=m-04389ec1075443bcxxxxxxxxxx
&Role=WRITER
&<公共请求参数>
```

正常返回示例

`XML` 格式

```
<RequestId>89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C</RequestId>
```

`JSON` 格式

```
{
  "RequestId": "89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C"
}
```

错误码

HttpCode	错误码	错误信息	描述
400	InvalidParameter	The specified parameter %s is invalid.	参数无效
400	MissingParameter	You must specify the parameter %s.	参数缺失
404	ResourceNotFound	The specified resource %s does not exist.	资源不存在
403	UnauthorizedOperation	You are not authorized to perform this operation. %s	无操作权限

访问错误中心查看更多错误码。

访问错误中心查看更多错误码。

8.9. DeleteMember

调用DeleteMember删除账本中某个成员。

调试

您可以在OpenAPI Explorer中直接运行该接口，免去您计算签名的困扰。运行成功后，OpenAPI Explorer可以自动生成SDK代码示例。

请求参数

名称	类型	是否必选	示例值	描述
Action	String	是	DeleteMember	系统规定参数。取值：DeleteMember。
LedgerId	String	是	l-c8cc7be3eea542axxxxxxxxxx	操作的账本实例ID。
MemberId	String	是	m-04389ec1075443bcxxxxxxxxxx	要删除的成员ID。

返回数据

名称	类型	示例值	描述
RequestId	String	89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C	请求ID。

示例

请求示例

```
http(s)://[Endpoint]/?Action=DeleteMember
&LedgerId=l-c8cc7be3eea542axxxxxxxxxx
&MemberId=m-04389ec1075443bcxxxxxxxxxx
&<公共请求参数>
```

正常返回示例

`XML` 格式

```
<RequestId>89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C</RequestId>
```

`JSON` 格式

```
{
  "RequestId": "89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C"
}
```

错误码

HttpCode	错误码	错误信息	描述
400	InvalidParameter	The specified parameter %s is invalid.	参数无效
400	MissingParameter	You must specify the parameter %s.	参数缺失
404	ResourceNotFound	The specified resource %s does not exist.	资源不存在
403	UnauthorizedOperation	You are not authorized to perform this operation. %s	无操作权限

访问错误中心查看更多错误码。

访问错误中心查看更多错误码。

9.账本记录API

9.1. GetJournal

调用GetJournal获取一条记录信息。

调试

您可以在OpenAPI Explorer中直接运行该接口，免去您计算签名的困扰。运行成功后，OpenAPI Explorer可以自动生成SDK代码示例。

请求参数

名称	类型	是否必选	示例值	描述
Action	String	是	GetJournal	系统规定参数。取值：GetJournal。
JournalId	Long	是	100	要获取的记录ID。
LedgerId	String	是	l-c8cc7be3eea542axxxxxxxxxx	操作的账本ID。

返回数据

名称	类型	示例值	描述
Journal	Struct		记录的详细信息。
ClientId	String	xxxx	记录写入者的客户端ID。
Clues	List	[clue1, clue2]	记录相关的线索列表。
JournalHash	String	04d45eb3bcbd2a47aeb5de87b73fd6a324bb7b57a081fd474a82d8dc7e830d42868aeb15a9fa02ac09205b68ecfc78e8e67c70c67736aa0e867a9a7011b83fab4b	记录Hash。
JournalId	String	100	记录ID。

名称	类型	示例值	描述
LedgerId	String	l-c8cc7be3eea542axxxxxxxxx	记录所在的账本实例ID。
MemberId	String	m-04389ec1075443bcxxxxxxxxx	记录写入者的成员ID。
PayloadJsonString	String	xxxxxxxxxxxxxxxxxxxx	记录内容。
PayloadType	String	CUSTOM	记录类型。取值如下： • CUSTOM：用户自定义操作。 • CREATE_LEDGER：创建账本。 • DELETE_LEDGER：删除账本。 • UPDATE_LEDGER：更新账本。 • CREATE_MEMBER：创建成员。 • DELETE_MEMBER：删除成员操作。 • UPDATE_MEMBER_KEY：更新成员公钥。 • UPDATE_MEMBER_PERMISSIONS：更新成员密钥。 • ENABLE_MEMBER：恢复成员。 • DISABLE_MEMBER：禁用成员。 • SET_TRUST_POINT：设置信任锚点。 • CREATE_TIMEANCHOR：创建时间锚点。 • KV_SET：Set一个KEY。 • KV_DELETE：Delete一个KEY。
Timestamp	Long	1587090122155	记录写入的时间。自1970年1月1日（00:00:00 GMT）以来的毫秒数。
RequestId	String	89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C	请求ID。

示例

请求示例

```
http(s)://[Endpoint]/?Action=GetJournal
&JournalId=100
&LedgerId=l-c8cc7be3eea542axxxxxxxxxx
&<公共请求参数>
```

正常返回示例

XML 格式

```
<RequestId>89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C</RequestId>
<Journal>
  <MemberId>m-04389ec1075443bcxxxxxxxxxx</MemberId>
  <JournalId>100</JournalId>
  <ClientId>xxxx</ClientId>
  <Clues>[clue1, clue2]</Clues>
  <LedgerId>l-c8cc7be3eea542axxxxxxxxxx</LedgerId>
  <JournalHash>04d45eb3bcbd2a47aeb5de87b73fd6a324bb7b57a081fd474a82d8dc7e830d42868aeb15a
9fa02ac09205b68ecfc78e8e67c70c67736aa0e867a9a7011b83fab4b</JournalHash>
  <Timestamp>1587090122155</Timestamp>
  <PayloadType>xxxxxxxxxxxxxxxxxxxx</PayloadType>
  <PayloadJsonString>CUSTOM_PAYLOAD</PayloadJsonString>
</Journal>
```

JSON 格式

```
{
  "RequestId": "89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C",
  "Journal": {
    "MemberId": "m-04389ec1075443bcxxxxxxxxxx",
    "JournalId": 100,
    "ClientId": "xxxx",
    "Clues": "[clue1, clue2]",
    "LedgerId": "l-c8cc7be3eea542axxxxxxxxxx",
    "JournalHash": "04d45eb3bcbd2a47aeb5de87b73fd6a324bb7b57a081fd474a82d8dc7e830d42868aeb
15a9fa02ac09205b68ecfc78e8e67c70c67736aa0e867a9a7011b83fab4b",
    "Timestamp": 1587090122155,
    "PayloadType": "xxxxxxxxxxxxxxxxxxxx",
    "PayloadJsonString": "CUSTOM_PAYLOAD"
  }
}
```

错误码

HttpCode	错误码	错误信息	描述
400	InvalidParameter	The specified parameter %s is invalid.	参数无效

HttpCode	错误码	错误信息	描述
400	MissingParameter	You must specify the parameter %s.	参数缺失
404	ResourceNotFound	The specified resource %s does not exist.	资源不存在
403	UnauthorizedOperation	You are not authorized to perform this operation. %s	无操作权限

访问错误中心查看更多错误码。

访问错误中心查看更多错误码。

9.2. ListJournals

调用ListJournals获取账本中记录列表。

调试

您可以在OpenAPI Explorer中直接运行该接口，免去您计算签名的困扰。运行成功后，OpenAPI Explorer可以自动生成SDK代码示例。

请求参数

名称	类型	是否必选	示例值	描述
Action	String	是	ListJournals	系统规定参数。取值：ListJournals。
LedgerId	String	是	l-c8cc7be3eea542axxxxxxxxxx	账本ID。
Clue	String	否	clue1	线索ID。
MemberId	String	否	m-04389ec1075443bcxxxxxxxxxx	成员ID。
NextToken	String	否	xxxx	用来标记当前开始读取的位置。置空表示从头开始。
MaxResults	Integer	否	100	本次读取的最大数据量，默认值为1024。

返回数据

名称	类型	示例值	描述
Journals	Array		记录列表。
ClientId	String	xxxx	写入记录的客户端ID。
Clues	List	clue1	记录的线索ID列表。
JournalHash	String	04d45eb3bcbd2a47aeb5de87b73fd6a324bb7b57a081fd474a82d8dc7e830d42868aeb15a9fa02ac09205b68ecfc78e8e67c70c67736aa0e867a9a7011b83fab4b	记录hash。
JournalId	String	100	记录ID。
LedgerId	String	l-c8cc7be3eea542axxxxxxxxxx	账本实例ID。
MemberId	String	m-04389ec1075443bcxxxxxxxxxx	写入记录的成员ID。
PayloadJsonString	String	xxxxxxxxxxxxxxxxxxxx	记录数据。
PayloadType	String	CUSTOM	记录类型。取值如下： • CUSTOM：用户自定义操作。 • CREATE_LEDGER：创建账本。 • DELETE_LEDGER：删除账本。 • UPDATE_LEDGER：更新账本。 • CREATE_MEMBER：创建成员。 • DELETE_MEMBER：删除成员操作。 • UPDATE_MEMBER_KEY：更新成员公钥。 • UPDATE_MEMBER_PERMISSIONS：更新成员密钥。 • ENABLE_MEMBER：恢复成员。 • DISABLE_MEMBER：禁用成员。 • SET_TRUST_POINT：设置信任锚点。 • CREATE_TIMEANCHOR：创建时间锚点。 • KV_SET：Set一个KEY。 • KV_DELETE：Delete一个KEY。

名称	类型	示例值	描述
Timestamp	Long	1587090122155	写入记录的时间。单位为毫秒。
MaxResults	Integer	100	本次读取的最大数据量。
NextToken	String	xxxx	用来表示当前调用返回读取到的位置。空代表数据已经读取完毕。
RequestId	String	89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C	请求ID。

示例

请求示例

```
http(s)://[Endpoint]/?Action=ListJournals
&LedgerId=l-c8cc7be3eea542axxxxxxxxxx
&<公共请求参数>
```

正常返回示例

`XML` 格式

```
<RequestId>89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C</RequestId>
<NextToken>xxxx</NextToken>
<MaxResults>100</MaxResults>
<Journals>
  <MemberId>m-04389ec1075443bcxxxxxxxxxx</MemberId>
  <JournalId>100</JournalId>
  <ClientId>xxxx</ClientId>
  <LedgerId>l-c8cc7be3eea542axxxxxxxxxx</LedgerId>
  <JournalHash>04d45eb3bcbd2a47aeb5de87b73fd6a324bb7b57a081fd474a82d8dc7e830d42868aeb15a
9fa02ac09205b68ecfc78e8e67c70c67736aa0e867a9a7011b83fab4b</JournalHash>
  <Timestamp>1587090122155</Timestamp>
  <PayloadType>CUSTOM_PAYLOAD</PayloadType>
  <PayloadJsonString>xxxxxxxxxxxxxxxxxxxxx</PayloadJsonString>
</Journals>
<Journals>
  <Clues>clue1</Clues>
</Journals>
```

JSON 格式


```
{
  "RequestId": "89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C",
  "NextToken": "xxxx",
  "MaxResults": 100,
  "Journals": [
    {
      "MemberId": "m-04389ec1075443bcxxxxxxxxxx",
      "JournalId": 100,
      "ClientId": "xxxx",
      "LedgerId": "l-c8cc7be3eea542axxxxxxxxxx",
      "JournalHash": "04d45eb3bcbd2a47aeb5de87b73fd6a324bb7b57a081fd474a82d8dc7e830d42868a
eb15a9fa02ac09205b68ecfc78e8e67c70c67736aa0e867a9a7011b83fab4b",
      "Timestamp": 1587090122155,
      "PayloadType": "CUSTOM_PAYLOAD",
      "PayloadJsonString": "xxxxxxxxxxxxxxxxxxxx"
    },
    {
      "Clues": "clue1"
    }
  ]
}
```


错误码

HttpCode	错误码	错误信息	描述
400	InvalidParameter	The specified parameter %s is invalid.	参数无效
400	MissingParameter	You must specify the parameter %s.	参数缺失
404	ResourceNotFound	The specified resource %s does not exist.	资源不存在
403	UnauthorizedOperation	You are not authorized to perform this operation. %s	无操作权限

访问错误中心查看更多错误码。

访问错误中心查看更多错误码。

10.时间锚点API

10.1. ListTimeAnchors

调用ListTimeAnchors获取账本的时间锚点列表

调试

您可以在OpenAPI Explorer中直接运行该接口，免去您计算签名的困扰。运行成功后，OpenAPI Explorer可以自动生成SDK代码示例。

请求参数

名称	类型	是否必选	示例值	描述
Action	String	是	ListTimeAnchors	系统规定参数。取值：ListTimeAnchors。
LedgerId	String	是	l-c8cc7be3eea542axxxxxxxxxx	账本实例ID。
Reverse	Boolean	否	false	迭代方向。默认为false，按时间锚点的创建时间从旧到新迭代。当设置为true后，按时间锚点的创建时间就新到旧迭代。
NextToken	String	否	xxx	用来标记当前开始读取的位置，置空表示从头开始。
MaxResults	Integer	否	100	本次读取的最大数据量，默认值为10。

返回数据

名称	类型	示例值	描述
MaxResults	Integer	100	本次读取的最大数据量。
NextToken	String	xxxx	用来表示当前调用返回读取到的位置。空代表数据已经读取完毕。
RequestId	String	89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C	请求ID。
TimeAnchors	Array		时间锚点列表。

名称	类型	示例值	描述
JournalId	Long	57	记录ID。
LedgerDigest	String	04d45eb3bcbd2a47aeb5de87b73fd6a324bb7b57a081fd474a82d8dc7e830d42868aeb15a9fa02ac09205b68ecfc78e8e67c70c67736aa0e867a9a7011b83fab4b	账本摘要。
LedgerDigestType	String	SHA256	账本摘要类型。
LedgerVersion	Long	57	账本版本。
Proof	String	{"hashInHex":"de6eb06dfbbe534eedfdad435698d340f0dcba91b8bd4b76758317b00562daf6","sequence":49998}	时间证据。
TimeStamp	Long	1587515659	时间戳。单位秒。

示例

请求示例

```
http(s)://[Endpoint]/?Action=ListTimeAnchors
&LedgerId=l-c8cc7be3eea542axxxxxxxxxx
&<公共请求参数>
```

正常返回示例

`XML` 格式

```
<TimeAnchors>
  <LedgerDigest>04d45eb3bcbd2a47aeb5de87b73fd6a324bb7b57a081fd474a82d8dc7e830d42868aeb15a9fa02ac09205b68ecfc78e8e67c70c67736aa0e867a9a7011b83fab4b</LedgerDigest>
  <JournalId>57</JournalId>
  <LedgerDigestType>SHA256</LedgerDigestType>
  <Proof>{"hashInHex":"de6eb06dfbbe534eedfdad435698d340f0dcba91b8bd4b76758317b00562daf6","sequence":49998}</Proof>
  <LedgerVersion>57</LedgerVersion>
  <TimeStamp>1587515659</TimeStamp>
</TimeAnchors>
<RequestId>89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C</RequestId>
<NextToken>xxxx</NextToken>
<MaxResults>100</MaxResults>
```

JSON 格式

```
{
  "TimeAnchors": {
    "LedgerDigest": "04d45eb3bcbd2a47aeb5de87b73fd6a324bb7b57a081fd474a82d8dc7e830d42868aeb15a9fa02ac09205b68ecfc78e8e67c70c67736aa0e867a9a7011b83fab4b",
    "JournalId": 57,
    "LedgerDigestType": "SHA256",
    "Proof": "{\"hashInHex\":\"de6eb06dfbbe534eedfdad435698d340f0dcba91b8bd4b76758317b00562daf6\",\"sequence\":49998}",
    "LedgerVersion": 57,
    "TimeStamp": 1587515659
  },
  "RequestId": "89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C",
  "NextToken": "xxxx",
  "MaxResults": 100
}
```

错误码

HttpCode	错误码	错误信息	描述
400	InvalidParameter	The specified parameter %s is invalid.	参数无效
400	MissingParameter	You must specify the parameter %s.	参数缺失

HttpCode	错误码	错误信息	描述
404	ResourceNotFound	The specified resource %s does not exist.	资源不存在
403	UnauthorizedOperation	You are not authorized to perform this operation. %s	无操作权限

访问错误中心查看更多错误码。

访问错误中心查看更多错误码。

11.访问控制API

11.1. CreateVpcEndpoint

调用CreateVpcEndpoint在VPC中创建某个账本实例的访问端点。

调试

您可以在OpenAPI Explorer中直接运行该接口，免去您计算签名的困扰。运行成功后，OpenAPI Explorer可以自动生成SDK代码示例。

请求参数

名称	类型	是否必选	示例值	描述
Action	String	是	CreateVpcEndpoint	系统规定参数。取值：CreateVpcEndpoint。
LedgerId	String	是	l-c8cc7be3eea542axxxxxxxxxx	需要创建访问端点的账本实例ID。
VpcId	String	是	vpc-bp15nlkfxxxxxxxxxxxxx	访问端点所在的专有网络ID。
VSwitchId	String	是	vsw-bp1l50xxxxxxxxxxxx	访问端点所在的虚拟交换机ID。
ClientToken	String	否	123e4567-e89b-12d3-a456-426655440000	保证请求幂等性。从您的客户端生成一个参数值，确保不同请求间该参数值唯一。ClientToken只支持ASCII字符，且不能超过64个字符。

返回数据

名称	类型	示例值	描述
RequestId	String	89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C	请求ID。
VpcEndpointId	String	xxxxxxxxxxxxxx.ledgerdb.aliyuncs.com	访问端点ID。

示例

请求示例

```
http(s)://[Endpoint]/?Action=CreateVpcEndpoint
&LedgerId=l-c8cc7be3eea542axxxxxxxxxx
&VpcId=vpc-bp15nlkfxxxxxxxxxxxxx
&VSwitchId=vsw-bp1l50xxxxxxxxxxxx
&<公共请求参数>
```

正常返回示例

`XML` 格式

```
<RequestId>89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C</RequestId>
<VpcEndpointId>xxxxxxxxxxxxxx.ledgerdb.aliyuncs.com</VpcEndpointId>
```

`JSON` 格式

```
{"RequestId":"89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C","VpcEndpointId":"xxxxxxxxxxxxxx.ledgerdb.aliyuncs.com"}
```

错误码

HttpCode	错误码	错误信息	描述
400	InvalidParameter	The specified parameter %s is invalid.	参数无效
400	MissingParameter	You must specify the parameter %s.	参数缺失
404	ResourceNotFound	The specified resource %s does not exist.	资源不存在
409	ResourceAlreadyExist	The resource %s already exists.	资源已经存在
403	UnauthorizedOperation	You are not authorized to perform this operation. %s	无操作权限

访问错误中心查看更多错误码。

访问错误中心查看更多错误码。

11.2. DeleteVpcEndpoint

调用DeleteVpcEndpoint在指定VPC中删除某个账本的访问端点。

调试

您可以在OpenAPI Explorer中直接运行该接口，免去您计算签名的困扰。运行成功后，OpenAPI Explorer可以自动生成SDK代码示例。

请求参数

名称	类型	是否必选	示例值	描述
Action	String	是	DeleteVpcEndpoint	系统规定参数。取值：DeleteVpcEndpoint。
LedgerId	String	是	l-c8cc7be3eea542axxxxxxxxxx	访问端点关联的账本实例ID。
VpcEndpointId	String	是	xxxxxxxxxxxxxx.ledgerdb.aliyuncs.com	要删除的访问端点ID。
VpcId	String	是	vpc-bp15nlkfxxxxxxxxxxxxx	访问端点所在的专有网络ID。
VSwitchId	String	是	vsw-bp1l50xxxxxxxxxxxx	访问端点所在的虚拟交换机ID。

返回数据

名称	类型	示例值	描述
RequestId	String	89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C	请求ID。

示例

请求示例

```
http(s)://[Endpoint]/?Action=DeleteVpcEndpoint
&LedgerId=l-c8cc7be3eea542axxxxxxxxxx
&VpcEndpointId=xxxxxxxxxxxxxx.ledgerdb.aliyuncs.com
&VpcId=vpc-bp15nlkfxxxxxxxxxxxxx
&VSwitchId=vsw-bp1l50xxxxxxxxxxxx
&<公共请求参数>
```

正常返回示例

`XML` 格式

```
<RequestId>89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C</RequestId>
```

`JSON` 格式

```
{
  "RequestId": "89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C"
}
```

错误码

HttpCode	错误码	错误信息	描述
400	InvalidParameter	The specified parameter %s is invalid.	参数无效
400	MissingParameter	You must specify the parameter %s.	参数缺失
404	ResourceNotFound	The specified resource %s does not exist.	资源不存在
403	UnauthorizedOperation	You are not authorized to perform this operation. %s	无操作权限

访问错误中心查看更多错误码。

访问错误中心查看更多错误码。

11.3. ListVpcEndpoints

调用ListVpcEndpoints查看一个账本实例的访问端点列表。

调试

您可以在OpenAPI Explorer中直接运行该接口，免去您计算签名的困扰。运行成功后，OpenAPI Explorer可以自动生成SDK代码示例。

请求参数

名称	类型	是否必选	示例值	描述
Action	String	是	ListVpcEndpoints	系统规定参数。取值：ListVpcEndpoints。
LedgerId	String	是	l-c8cc7be3eea542axxxxxxxxxx	账本实例ID。
NextToken	String	否	xxxxx	用来标记当前开始读取的位置，置空表示从头开始。
MaxResults	Integer	否	10	本次读取的最大数据量，默认值为10。

返回数据

名称	类型	示例值	描述
MaxResults	Integer	100	本次读取的最大数据量。
NextToken	String	xxxx	用来表示当前调用返回读取到的位置，空代表数据已经读取完毕。
RequestId	String	89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C	请求ID。
VpcEndpoints	Array		访问端点列表。
Address	String	10.34.23.144	访问端点在VPC中对应的ip地址。
CreateTime	Long	1587515659452	访问端点的创建时间。为自1970年1月1日（00:00:00 GMT）以来的毫秒数。
LedgerId	String	l-c8cc7be3eea542axxxxxxxxxx	所属的账本实例ID。
MemberId	String	m-04389ec1075443bcxxxxxxxxxx	所属的成员ID。
RegionId	String	cn-hangzhou	所在地域。
Status	String	NORMAL	访问端点状态。取值：CREATING、READY、DELETING。CREATING状态表示该访问端点正在创建过程中，READY状态表示该访问端点创建完成并且可以使用，DELETING状态表示该访问端点正在销毁中。
VSwitchId	String	vsw-bp1l50xxxxxxxxxxx	访问端点所在的虚拟交换机ID。
VpcEndpointId	String	xxxxxx.ledgerdb.aliyuncs.com	访问端点的ID。
VpcId	String	vpc-bp15nlkfxxxxxxxxxxxx	访问端点所在的虚拟网络ID。

示例

请求示例

```
http(s)://[Endpoint]/?Action=ListVpcEndpoints
&LedgerId=l-c8cc7be3eea542axxxxxxxxxx
&<公共请求参数>
```

正常返回示例

`XML` 格式

```
<RequestId>89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C</RequestId>
<NextToken>xxxx</NextToken>
<VpcEndpoints>
  <Status>NORMAL</Status>
  <MemberId>m-04389ec1075443bcxxxxxxxxxx</MemberId>
  <VpcId>vpc-bp15nlkfxxxxxxxxxxxxx</VpcId>
  <Address>10.34.23.144</Address>
  <VSwitchId>vsw-bp1l50xxxxxxxxxxx</VSwitchId>
  <CreateTime>1587515659452</CreateTime>
  <LedgerId>l-c8cc7be3eea542axxxxxxxxxx</LedgerId>
  <RegionId>cn-hangzhou</RegionId>
  <VpcEndpointId>xxxxxx.ledgerdb.aliyuncs.com</VpcEndpointId>
</VpcEndpoints>
<MaxResults>100</MaxResults>
```

`JSON` 格式

```
{
  "RequestId": "89E6B5F5-7511-46A7-9EDB-3C6F8AA4D48C",
  "NextToken": "xxxx",
  "VpcEndpoints": {
    "Status": "NORMAL",
    "MemberId": "m-04389ec1075443bcxxxxxxxxxx",
    "VpcId": "vpc-bp15nlkfxxxxxxxxxxxxx",
    "Address": "10.34.23.144",
    "VSwitchId": "vsw-bp1l50xxxxxxxxxxx",
    "CreateTime": 1587515659452,
    "LedgerId": "l-c8cc7be3eea542axxxxxxxxxx",
    "RegionId": "cn-hangzhou",
    "VpcEndpointId": "xxxxxx.ledgerdb.aliyuncs.com"
  },
  "MaxResults": 100
}
```


错误码

HttpCode	错误码	错误信息	描述
400	InvalidParameter	The specified parameter %s is invalid.	参数无效
400	MissingParameter	You must specify the parameter %s.	参数缺失
404	ResourceNotFound	The specified resource %s does not exist.	资源不存在
403	UnauthorizedOperation	You are not authorized to perform this operation. %s	无操作权限

访问错误中心查看更多错误码。

访问错误中心查看更多错误码。

12.客户端API

账本操作API

API	描述
statLedger	查询账本当前状态及相关统计数据

成员操作API

API	描述
updateMemberKey	更新账本成员公钥
getMember	查询账本成员
listMembers	批量查询账本成员
enableMember	激活账本成员
disableMember	禁用账本成员

记录（数据）操作API

API	描述
appendTransaction	新增记录
getTransaction	读取记录
existTransaction	查询记录是否存在
listTransactions	批量获取历史记录
getBlockInfo	获取区块信息

listTimeAnchors	批量查询可信时间戳
getLastTimeAnchor	查询最新的可信时间戳
getProof	获取记录证明

秘钥生成

API	描述
ECCK1KeyPair 秘钥对生成	秘钥对类，包含构造器和静态生成秘钥方法

客户端开源地址

https://github.com/aliyun/alibabacloud-ledgerdb-java-client

客户端构建及使用

Step1: build and install

```
$ git clone https://github.com/aliyun/alibabacloud-ledgerdb-java-client.git
$ cd alibabacloud-ledgerdb-java-client
$ mvn clean install -DskipTests
```

Step2: 在完成Step1后，在您的项目中添加依赖

```
<dependency>
   <groupId>com.antfin.ledgerdb</groupId>
   <artifactId>ledger-client</artifactId>
   <version>0.8.1</version>
</dependency>
```

客户端API Demo

```
import com.antfin.ledgerdb.client.LedgerDBLightClient;
import com.antfin.ledgerdb.client.OperationControl;
import com.antfin.ledgerdb.client.common.AppendTransactionResponse;
import com.antfin.ledgerdb.client.common.GetTransactionResponse;
import com.antfin.ledgerdb.client.common.SignerProfile;
import com.antfin.ledgerdb.client.crypto.ECCK1KeyPair;
import com.antfin.ledgerdb.client.crypto.Keys;
import com.antfin.ledgerdb.client.proto.Sender;
```

```
import org.bouncycastle.util.encoders.Hex;

import java.nio.charset.StandardCharsets;

public class LedgerStart{

 public static void main(String[] args) throws Exception {
  String ledgerId = "******";
  String memberId = "******";

  // VPC Endpoint可通过LedgerDB控制台提供的VPC配置功能获取
  String vpcEndpoint = "127.0.0.1";

  // 目前端口固定为10077
  int port = 10077;

  // ledgerUri参数的格式为 ledger://{VPC Endpoint}:10077/{ledgerID}
  String ledgerUri =
    "ledger://" + vpcEndpoint + ":" + port + "/" + ledgerId;
  LedgerDBLightClient client =
    new LedgerDBLightClient(vpcEndpoint, port);

  ECCK1KeyPair key = Keys.createEcKeyPair();
  byte[] publicKeyInBytes = key.getPublicKey();
  // 请将公钥通过LedgerDB控制台提供的密钥管理功能上传，用于后续对写入数据的签名校验

  SignerProfile profile = new SignerProfile(memberId, Sender.SenderType.REGULAR, key);
  OperationControl op = new OperationControl();
  op.addSignerProfile(profile);

  AppendTransactionResponse res =
    client.appendTransaction(ledgerUri, "hello ledger test".getBytes(StandardCharsets.US_ASCII), op)
;
  System.out.println(Hex.toHexString(res.getTxHash()));

  long sequence = res.getTotalSequence();
  System.out.println(sequence);

  OperationControl ocGet = new OperationControl();
  ocGet.addSignerProfile(profile);
  GetTransactionResponse getRes = client.getTransaction(ledgerUri, sequence, ocGet);
```

```
    byte[] payload =
        getRes.getPayload().getTx().getRequest().getCustomPayload().toByteArray();
    System.out.println("Content: " + new String(payload, StandardCharsets.US_ASCII));
  }
}
```

账本操作API

statLedger

方法定义

```
public StatLedgerResponse statLedger(String ledgerUri,OperationControl opControl)
```

示例

```
import com.antfin.ledgerdb.client.LedgerDBLightClient;
import com.antfin.ledgerdb.client.OperationControl;
import com.antfin.ledgerdb.client.common.SignerProfile;
import com.antfin.ledgerdb.client.common.StatLedgerResponse;
import com.antfin.ledgerdb.client.crypto.ECCK1KeyPair;
import com.antfin.ledgerdb.client.crypto.SignerKeyPair;
import com.antfin.ledgerdb.client.proto.Sender;
import org.bouncycastle.util.encoders.Hex;
import org.junit.Test;

public class StatLedgerDemo{

  String ledgerId = "*****";

  String memberId = "*****";

  String privateKeyInHex = "your private key in hex";

  String publicKeyInHex = "your public key in hex";

  byte[] privateKeyInBytes = Hex.decode(privateKeyInHex);

  byte[] publicKeyInBytes = Hex.decode(publicKeyInHex);

  String ledgerHost = "127.0.0.1";
```

```
  Integer ledgerPort = 10099;

  LedgerDBLightClient client = new LedgerDBLightClient(ledgerHost, ledgerPort);
  SignerKeyPair signerKeyPair = new ECCK1KeyPair(privateKeyInBytes, publicKeyInBytes);

  @Test
  public void test(){
    OperationControl op = new OperationControl();
    op.addSignerProfile(
       new SignerProfile(
          memberId,
          Sender.SenderType.REGULAR,
          signerKeyPair));
    String ledgerUri = "ledger://127.0.0.1:10099/*****";
    StatLedgerResponse response = client.statLedger(ledgerUri, op);
  }
}
```

成员操作API

updateMemberKey

方法定义

```
public UpdateMemberKeyResponse updateMemberKey(
        String ledgerUri,
        String memberId,
        byte[] publicKey,
        OperationControl operationControl)
```

示例

```
import com.antfin.ledgerdb.client.LedgerDBLightClient;
import com.antfin.ledgerdb.client.OperationControl;
import com.antfin.ledgerdb.client.common.SignerProfile;
import com.antfin.ledgerdb.client.common.UpdateMemberKeyResponse;
import com.antfin.ledgerdb.client.crypto.ECCK1KeyPair;
import com.antfin.ledgerdb.client.crypto.SignerKeyPair;
import com.antfin.ledgerdb.client.proto.Sender;
import org.bouncycastle.util.encoders.Hex;
import org.junit.Test;
```

```
public class UpdateMemberKeyDemo{

 String ledgerId = "*****";

 String memberId = "*****";

 String privateKeyInHex = "your private key in hex";

 String publicKeyInHex = "your public key in hex";

 byte[] privateKeyInBytes = Hex.decode(privateKeyInHex);

 byte[] publicKeyInBytes = Hex.decode(publicKeyInHex);

 String ledgerHost = "127.0.0.1";

 Integer ledgerPort = 10099;

 LedgerDBLightClient client = new LedgerDBLightClient(ledgerHost, ledgerPort);
 SignerKeyPair signerKeyPair = new ECCK1KeyPair(privateKeyInBytes, publicKeyInBytes);

 @Test
 public void test(){
  OperationControl op = new OperationControl();
  op.addSignerProfile(
     new SignerProfile(
        memberId,
        Sender.SenderType.REGULAR,
        signerKeyPair));
  String ledgerUri = "ledger://127.0.0.1:10099/*****";
  byte[] newPublicKey = Hex.decode("new public key in hex");
  UpdateMemberKeyResponse response = client.updateMemberKey(ledgerUri, memberId, newPublicK
ey, op);
 }
}
```

getMember

方法定义

```
public GetMemberResponse getMember(
    String ledgerUri,
    String memberId,
    OperationControl operationControl)
```

示例

```
import com.antfin.ledgerdb.client.LedgerDBLightClient;
import com.antfin.ledgerdb.client.OperationControl;
import com.antfin.ledgerdb.client.common.GetMemberResponse;
import com.antfin.ledgerdb.client.common.SignerProfile;
import com.antfin.ledgerdb.client.crypto.ECCK1KeyPair;
import com.antfin.ledgerdb.client.crypto.SignerKeyPair;
import com.antfin.ledgerdb.client.proto.Sender;
import org.bouncycastle.util.encoders.Hex;
import org.junit.Test;

public class GetMemberDemo{

 String ledgerId = "******";

 String memberId = "******";

 String privateKeyInHex = "your private key in hex";

 String publicKeyInHex = "your public key in hex";

 byte[] privateKeyInBytes = Hex.decode(privateKeyInHex);

 byte[] publicKeyInBytes = Hex.decode(publicKeyInHex);

 String ledgerHost = "127.0.0.1";

 Integer ledgerPort = 10099;

 LedgerDBLightClient client = new LedgerDBLightClient(ledgerHost, ledgerPort);
 SignerKeyPair signerKeyPair = new ECCK1KeyPair(privateKeyInBytes, publicKeyInBytes);

 @Test
 public void test(){
  OperationControl op = new OperationControl();
```

```
    op.addSignerProfile(
        new SignerProfile(
            memberId,
            Sender.SenderType.REGULAR,
            signerKeyPair));
    String ledgerUri = "ledger://127.0.0.1:10099/*****";
    GetMemberResponse response = client.getMember(ledgerUri, memberId, op);
  }
}
```

listMembers

方法定义

```
public ListMembersResponse listMembers(
        String ledgerUri,
        String lastMemberId,
        int limit,
        OperationControl operationControl)
```

示例

```
import com.antfin.ledgerdb.client.LedgerDBLightClient;
import com.antfin.ledgerdb.client.OperationControl;
import com.antfin.ledgerdb.client.common.ListMembersResponse;
import com.antfin.ledgerdb.client.common.SignerProfile;
import com.antfin.ledgerdb.client.crypto.ECCK1KeyPair;
import com.antfin.ledgerdb.client.crypto.SignerKeyPair;
import com.antfin.ledgerdb.client.proto.Sender;
import org.bouncycastle.util.encoders.Hex;
import org.junit.Test;

public class ListMembers{

  String ledgerId = "*****";

  String memberId = "*****";

  String privateKeyInHex = "your private key in hex";

  String publicKeyInHex = "your public key in hex";
```

```
  byte[] privateKeyInBytes = Hex.decode(privateKeyInHex);

  byte[] publicKeyInBytes = Hex.decode(publicKeyInHex);

  String ledgerHost = "127.0.0.1";

  Integer ledgerPort = 10099;

  LedgerDBLightClient client = new LedgerDBLightClient(ledgerHost, ledgerPort);
  SignerKeyPair signerKeyPair = new ECCK1KeyPair(privateKeyInBytes, publicKeyInBytes);

  @Test
  public void test(){
    OperationControl op = new OperationControl();
    op.addSignerProfile(
       new SignerProfile(
          memberId,
          Sender.SenderType.REGULAR,
          signerKeyPair));
    String ledgerUri = "ledger://127.0.0.1:10099/*****";
    ListMembersResponse response = client.listMembers(ledgerUri, "", 20, op);
  }
}
```

enableMember

方法定义

```
public EnableMemberResponse enableMember(
    String ledgerUri,
    String memberId,
    OperationControl op)
```

示例

```
import com.antfin.ledgerdb.client.LedgerDBLightClient;
import com.antfin.ledgerdb.client.OperationControl;
import com.antfin.ledgerdb.client.common.EnableMemberResponse;
import com.antfin.ledgerdb.client.common.SignerProfile;
import com.antfin.ledgerdb.client.crypto.ECCK1KeyPair;
import com.antfin.ledgerdb.client.crypto.SignerKeyPair;
import com.antfin.ledgerdb.client.proto.Sender;
```

```
import com.antfin.ledgerdb.client.proto.Sender;
import org.bouncycastle.util.encoders.Hex;
import org.junit.Test;

public class EnableMemberDemo{

 String ledgerId = "*****";

 String memberId = "*****";

 String privateKeyInHex = "your private key in hex";

 String publicKeyInHex = "your public key in hex";

 byte[] privateKeyInBytes = Hex.decode(privateKeyInHex);

 byte[] publicKeyInBytes = Hex.decode(publicKeyInHex);

 String ledgerHost = "127.0.0.1";

 Integer ledgerPort = 10099;

 LedgerDBLightClient client = new LedgerDBLightClient(ledgerHost, ledgerPort);
 SignerKeyPair signerKeyPair = new ECCK1KeyPair(privateKeyInBytes, publicKeyInBytes);

 @Test
 public void test(){
   OperationControl op = new OperationControl();
   op.addSignerProfile(
      new SignerProfile(
         memberId,
         Sender.SenderType.REGULAR,
         signerKeyPair));
   String ledgerUri = "ledger://127.0.0.1:10099/*****";
   String memberToEnable = "*****";
   EnableMemberResponse enableMemberResponse = client.enableMember(ledgerUri, memberToEnable, op);

 }
}
```

disableMember

方法定义

```
public DisableMemberResponse disableMember(
    String ledgerUri,
    String memberId,
    OperationControl op)
```

示例

```
import com.antfin.ledgerdb.client.LedgerDBLightClient;
import com.antfin.ledgerdb.client.OperationControl;
import com.antfin.ledgerdb.client.common.DisableMemberResponse;
import com.antfin.ledgerdb.client.common.SignerProfile;
import com.antfin.ledgerdb.client.crypto.ECCK1KeyPair;
import com.antfin.ledgerdb.client.crypto.SignerKeyPair;
import com.antfin.ledgerdb.client.proto.Sender;
import org.bouncycastle.util.encoders.Hex;
import org.junit.Test;

public class DisableMemberDemo{

 String ledgerId = "*****";

 String memberId = "*****";

 String privateKeyInHex = "your private key in hex";

 String publicKeyInHex = "your public key in hex";

 byte[] privateKeyInBytes = Hex.decode(privateKeyInHex);

 byte[] publicKeyInBytes = Hex.decode(publicKeyInHex);

 String ledgerHost = "127.0.0.1";

 Integer ledgerPort = 10099;

 LedgerDBLightClient client = new LedgerDBLightClient(ledgerHost, ledgerPort);
 SignerKeyPair signerKeyPair = new ECCK1KeyPair(privateKeyInBytes, publicKeyInBytes);
```

```
 @Test
 public void test(){
  OperationControl op = new OperationControl();
  op.addSignerProfile(
     new SignerProfile(
        memberId,
        Sender.SenderType.REGULAR,
        signerKeyPair));
  String ledgerUri = "ledger://127.0.0.1:10099/*****";
  String memberToDisable = "*****";
  DisableMemberResponse disableMemberResponse = client.disableMember(ledgerUri, memberToDis
able, op);
 }
}
```

记录（数据）操作API

appendTransaction

方法定义

```
public AppendTransactionResponse appendTransaction(
    String ledgerUri,
    byte[] data,
    OperationControl opControl)
```

示例

```
import com.antfin.ledgerdb.client.LedgerDBLightClient;
import com.antfin.ledgerdb.client.OperationControl;
import com.antfin.ledgerdb.client.common.AppendTransactionResponse;
import com.antfin.ledgerdb.client.common.SignerProfile;
import com.antfin.ledgerdb.client.crypto.ECCK1KeyPair;
import com.antfin.ledgerdb.client.crypto.SignerKeyPair;
import com.antfin.ledgerdb.client.proto.Sender;
import org.bouncycastle.util.encoders.Hex;
import org.junit.Test;

import java.nio.charset.StandardCharsets;

public class AppendTransactionDemo{
```

```
  String ledgerId = "*****";

  String memberId = "*****";

  String privateKeyInHex = "your private key in hex";

  String publicKeyInHex = "your public key in hex";

  byte[] privateKeyInBytes = Hex.decode(privateKeyInHex);

  byte[] publicKeyInBytes = Hex.decode(publicKeyInHex);

  String ledgerHost = "127.0.0.1";

  Integer ledgerPort = 10099;

  LedgerDBLightClient client = new LedgerDBLightClient(ledgerHost, ledgerPort);
  SignerKeyPair signerKeyPair = new ECCK1KeyPair(privateKeyInBytes, publicKeyInBytes);

  @Test
  public void test(){
    OperationControl op = new OperationControl();
    op.addSignerProfile(
        new SignerProfile(
            memberId,
            Sender.SenderType.REGULAR,
            signerKeyPair));
    String ledgerUri = "ledger://127.0.0.1:10099/*****";
    AppendTransactionResponse response =
        client.appendTransaction(ledgerUri, "Hello world".getBytes(StandardCharsets.UTF_8), op);

  }
}
```

getTransaction

方法定义

```
public GetTransactionResponse getTransaction(
    String ledgerUri,
    long txSequence,
    OperationControl opControl)
```

示例

```
import com.antfin.ledgerdb.client.LedgerDBLightClient;
import com.antfin.ledgerdb.client.OperationControl;
import com.antfin.ledgerdb.client.common.GetTransactionResponse;
import com.antfin.ledgerdb.client.common.SignerProfile;
import com.antfin.ledgerdb.client.crypto.ECCK1KeyPair;
import com.antfin.ledgerdb.client.crypto.SignerKeyPair;
import com.antfin.ledgerdb.client.proto.Sender;
import org.bouncycastle.util.encoders.Hex;
import org.junit.Test;

public class GetTransaction{

 String ledgerId = "*****";

 String memberId = "*****";

 String privateKeyInHex = "your private key in hex";

 String publicKeyInHex = "your public key in hex";

 byte[] privateKeyInBytes = Hex.decode(privateKeyInHex);

 byte[] publicKeyInBytes = Hex.decode(publicKeyInHex);

 String ledgerHost = "127.0.0.1";

 Integer ledgerPort = 10099;

 LedgerDBLightClient client = new LedgerDBLightClient(ledgerHost, ledgerPort);
 SignerKeyPair signerKeyPair = new ECCK1KeyPair(privateKeyInBytes, publicKeyInBytes);

 @Test
 public void test(){
  OperationControl op = new OperationControl();
```

```
    op.addSignerProfile(
        new SignerProfile(
            memberId,
            Sender.SenderType.REGULAR,
            signerKeyPair));
    String ledgerUri = "ledger://127.0.0.1:10099/*****";
    GetTransactionResponse response = client.getTransaction(ledgerUri, 10, op);
  }
}
```

existTransaction

方法定义

```
public ExistTransactionResponse existTransaction(
    String ledgerUri,
    long txSequence,
    OperationControl opControl)
```

示例

```
import com.antfin.ledgerdb.client.LedgerDBLightClient;
import com.antfin.ledgerdb.client.OperationControl;
import com.antfin.ledgerdb.client.common.ExistTransactionResponse;
import com.antfin.ledgerdb.client.common.SignerProfile;
import com.antfin.ledgerdb.client.crypto.ECCK1KeyPair;
import com.antfin.ledgerdb.client.crypto.SignerKeyPair;
import com.antfin.ledgerdb.client.proto.Sender;
import org.bouncycastle.util.encoders.Hex;
import org.junit.Test;

public class ExistTransactionDemo{

  String ledgerId = "******";

  String memberId = "******";

  String privateKeyInHex = "your private key in hex";

  String publicKeyInHex = "your public key in hex";

  byte[] privateKeyInBytes = Hex.decode(privateKeyInHex);
```

```
byte[] privateKeyInBytes = Hex.decode(privateKeyInHex);

byte[] publicKeyInBytes = Hex.decode(publicKeyInHex);

String ledgerHost = "127.0.0.1";

Integer ledgerPort = 10099;

LedgerDBLightClient client = new LedgerDBLightClient(ledgerHost, ledgerPort);
SignerKeyPair signerKeyPair = new ECCK1KeyPair(privateKeyInBytes, publicKeyInBytes);

@Test
public void test(){
  OperationControl op = new OperationControl();
  op.addSignerProfile(
     new SignerProfile(
        memberId,
        Sender.SenderType.REGULAR,
        signerKeyPair));
  String ledgerUri = "ledger://127.0.0.1:10099/*****";
  ExistTransactionResponse response = client.existTransaction(ledgerUri, 10, op);
 }
}
```

listTransactions

方法定义

```
public ListTransactionsResponse listTransactions(
    String ledgerUri,
    String clue,
    long beginSequence,
    int limit,
    OperationControl opControl)
```

示例

```
import com.antfin.ledgerdb.client.LedgerDBLightClient;
import com.antfin.ledgerdb.client.OperationControl;
import com.antfin.ledgerdb.client.common.ListTransactionsResponse;
import com.antfin.ledgerdb.client.common.SignerProfile;
import com.antfin.ledgerdb.client.crypto.ECCK1KeyPair;
import com.antfin.ledgerdb.client.crypto.SignerKeyPair;
```

```
import com.antfin.ledgerdb.client.crypto.SignerKeyPair;
import com.antfin.ledgerdb.client.proto.Sender;
import org.bouncycastle.util.encoders.Hex;
import org.junit.Test;

public class ListTransactionsDemo{

 String ledgerId = "*****";

 String memberId = "*****";

 String privateKeyInHex = "your private key in hex";

 String publicKeyInHex = "your public key in hex";

 byte[] privateKeyInBytes = Hex.decode(privateKeyInHex);

 byte[] publicKeyInBytes = Hex.decode(publicKeyInHex);

 String ledgerHost = "127.0.0.1";

 Integer ledgerPort = 10099;

 LedgerDBLightClient client = new LedgerDBLightClient(ledgerHost, ledgerPort);
 SignerKeyPair signerKeyPair = new ECCK1KeyPair(privateKeyInBytes, publicKeyInBytes);

 @Test
 public void test(){
  OperationControl op = new OperationControl();
  op.addSignerProfile(
     new SignerProfile(
        memberId,
        Sender.SenderType.REGULAR,
        signerKeyPair));
  String ledgerUri = "ledger://127.0.0.1:10099/*****";
  ListTransactionsResponse response =
     client.listTransactions(ledgerUri, "", 1, 10, op);
 }
}
```

setTrustPoint

方法定义

```
public SetTrustPointResponse setTrustPoint(
    String ledgerUri,
    long sequence,
    OperationControl op)
```

示例

```
import com.antfin.ledgerdb.client.LedgerDBLightClient;
import com.antfin.ledgerdb.client.OperationControl;
import com.antfin.ledgerdb.client.common.SetTrustPointResponse;
import com.antfin.ledgerdb.client.common.SignerProfile;
import com.antfin.ledgerdb.client.crypto.ECCK1KeyPair;
import com.antfin.ledgerdb.client.crypto.SignerKeyPair;
import com.antfin.ledgerdb.client.proto.Sender;
import org.bouncycastle.util.encoders.Hex;
import org.junit.Test;

public class SetTrustPointDemo{

 String ledgerId = "*****";

 String memberId = "*****";

 String privateKeyInHex = "your private key in hex";

 String publicKeyInHex = "your public key in hex";

 byte[] privateKeyInBytes = Hex.decode(privateKeyInHex);

 byte[] publicKeyInBytes = Hex.decode(publicKeyInHex);

 String ledgerHost = "127.0.0.1";

 Integer ledgerPort = 10099;

 LedgerDBLightClient client = new LedgerDBLightClient(ledgerHost, ledgerPort);
 SignerKeyPair signerKeyPair = new ECCK1KeyPair(privateKeyInBytes, publicKeyInBytes);
```

```
  @Test
  public void test(){
    OperationControl op = new OperationControl();
    op.addSignerProfile(
        new SignerProfile(
            memberId,
            Sender.SenderType.REGULAR,
            signerKeyPair));
    String ledgerUri = "ledger://127.0.0.1:10099/*****";
    SetTrustPointResponse response = client.setTrustPoint(ledgerUri, 10, op);
  }
}
```

getTrustPoint

方法定义

```
public GetTrustPointResponse getTrustPoint(
    String ledgerUri,
    OperationControl op)
```

示例

```
import com.antfin.ledgerdb.client.LedgerDBLightClient;
import com.antfin.ledgerdb.client.OperationControl;
import com.antfin.ledgerdb.client.common.GetTrustPointResponse;
import com.antfin.ledgerdb.client.common.SignerProfile;
import com.antfin.ledgerdb.client.crypto.ECCK1KeyPair;
import com.antfin.ledgerdb.client.crypto.SignerKeyPair;
import com.antfin.ledgerdb.client.proto.Sender;
import org.bouncycastle.util.encoders.Hex;
import org.junit.Test;

public class GetTrustPoint{

  String ledgerId = "*****";

  String memberId = "*****";

  String privateKeyInHex = "your private key in hex";

  String publicKeyInHex = "your public key in hex";
```

```
  byte[] privateKeyInBytes = Hex.decode(privateKeyInHex);

  byte[] publicKeyInBytes = Hex.decode(publicKeyInHex);

  String ledgerHost = "127.0.0.1";

  Integer ledgerPort = 10099;

  LedgerDBLightClient client = new LedgerDBLightClient(ledgerHost, ledgerPort);
  SignerKeyPair signerKeyPair = new ECCK1KeyPair(privateKeyInBytes, publicKeyInBytes);

  @Test
  public void test(){
    OperationControl op = new OperationControl();
    op.addSignerProfile(
        new SignerProfile(
            memberId,
            Sender.SenderType.REGULAR,
            signerKeyPair));
    String ledgerUri = "ledger://127.0.0.1:10099/*****";
    GetTrustPointResponse getTrustPointResponse = client.getTrustPoint(ledgerUri, op);
  }
}
```

getLastTimeAnchor

方法定义

```
public GetLastTimeAnchorResponse getLastTimeAnchor(
    String ledgerUri,
    OperationControl op)
```

示例

```
import com.antfin.ledgerdb.client.LedgerDBLightClient;
import com.antfin.ledgerdb.client.OperationControl;
import com.antfin.ledgerdb.client.common.GetLastTimeAnchorResponse;
import com.antfin.ledgerdb.client.common.SignerProfile;
import com.antfin.ledgerdb.client.crypto.ECCK1KeyPair;
import com.antfin.ledgerdb.client.crypto.SignerKeyPair;
import com.antfin.ledgerdb.client.proto.Sender;
```

```
import org.bouncycastle.util.encoders.Hex;
import org.junit.Test;

public class GetLastTimeAnchorDemo{

 String ledgerId = "******";

 String memberId = "******";

 String privateKeyInHex = "your private key in hex";

 String publicKeyInHex = "your public key in hex";

 byte[] privateKeyInBytes = Hex.decode(privateKeyInHex);

 byte[] publicKeyInBytes = Hex.decode(publicKeyInHex);

 String ledgerHost = "127.0.0.1";

 Integer ledgerPort = 10099;

 LedgerDBLightClient client = new LedgerDBLightClient(ledgerHost, ledgerPort);
 SignerKeyPair signerKeyPair = new ECCK1KeyPair(privateKeyInBytes, publicKeyInBytes);

 @Test
 public void test(){
  OperationControl op = new OperationControl();
  op.addSignerProfile(
     new SignerProfile(
        memberId,
        Sender.SenderType.REGULAR,
        signerKeyPair));
  String ledgerUri = "ledger://127.0.0.1:10099/******";
  GetLastTimeAnchorResponse response = client.getLastTimeAnchor(ledgerUri, op);
 }
}
```

listTimeAnchors

方法定义

```
public ListTimeAnchorsResponse listTimeAnchors(
    String ledgerUri,
    long startSequence,
    int limit,
    boolean reverse,
    OperationControl op)
```

示例

```
import com.antfin.ledgerdb.client.LedgerDBLightClient;
import com.antfin.ledgerdb.client.OperationControl;
import com.antfin.ledgerdb.client.common.ListTimeAnchorsResponse;
import com.antfin.ledgerdb.client.common.SignerProfile;
import com.antfin.ledgerdb.client.crypto.ECCK1KeyPair;
import com.antfin.ledgerdb.client.crypto.SignerKeyPair;
import com.antfin.ledgerdb.client.proto.Sender;
import org.bouncycastle.util.encoders.Hex;
import org.junit.Test;

public class ListTimeAnchorsDemo{

 String ledgerId = "******";

 String memberId = "******";

 String privateKeyInHex = "your private key in hex";

 String publicKeyInHex = "your public key in hex";

 byte[] privateKeyInBytes = Hex.decode(privateKeyInHex);

 byte[] publicKeyInBytes = Hex.decode(publicKeyInHex);

 String ledgerHost = "127.0.0.1";

 Integer ledgerPort = 10099;

 LedgerDBLightClient client = new LedgerDBLightClient(ledgerHost, ledgerPort);
 SignerKeyPair signerKeyPair = new ECCK1KeyPair(privateKeyInBytes, publicKeyInBytes);

 @Test
```

```
  public void test(){
    OperationControl op = new OperationControl();
    op.addSignerProfile(
        new SignerProfile(
            memberId,
            Sender.SenderType.REGULAR,
            signerKeyPair));
    String ledgerUri = "ledger://127.0.0.1:10099/*****";
    ListTimeAnchorsResponse response =
        client.listTimeAnchors(ledgerUri, 10, 10, false, op);
  }
}
```

getBlockInfo

方法定义

```
public GetBlockInfoResponse getBlockInfo(
    String ledgerUri,
    long blockSequence,
    OperationControl op)
```

示例

```
import com.antfin.ledgerdb.client.LedgerDBLightClient;
import com.antfin.ledgerdb.client.OperationControl;
import com.antfin.ledgerdb.client.common.GetBlockInfoResponse;
import com.antfin.ledgerdb.client.common.SignerProfile;
import com.antfin.ledgerdb.client.crypto.ECCK1KeyPair;
import com.antfin.ledgerdb.client.crypto.SignerKeyPair;
import com.antfin.ledgerdb.client.proto.Sender;
import org.bouncycastle.util.encoders.Hex;
import org.junit.Test;

public class GetBlockInfoDemo{

  String ledgerId = "*****";

  String memberId = "*****";

  String privateKeyInHex = "your private key in hex";
```

```
String publicKeyInHex = "your public key in hex";

byte[] privateKeyInBytes = Hex.decode(privateKeyInHex);

byte[] publicKeyInBytes = Hex.decode(publicKeyInHex);

String ledgerHost = "127.0.0.1";

Integer ledgerPort = 10099;

LedgerDBLightClient client = new LedgerDBLightClient(ledgerHost, ledgerPort);
SignerKeyPair signerKeyPair = new ECCK1KeyPair(privateKeyInBytes, publicKeyInBytes);

@Test
public void test(){
  OperationControl op = new OperationControl();
  op.addSignerProfile(
     new SignerProfile(
        memberId,
        Sender.SenderType.REGULAR,
        signerKeyPair));
  String ledgerUri = "ledger://127.0.0.1:10099/*****";
  GetBlockInfoResponse response = client.getBlockInfo(ledgerUri, 5, op);
 }
}
```

getProof

方法定义

```
public GetProofResponse getProof(String ledgerUri, long txSequence, OperationControl op)
```

示例

```
import com.antfin.ledgerdb.client.LedgerDBLightClient;
import com.antfin.ledgerdb.client.OperationControl;
import com.antfin.ledgerdb.client.common.GetBlockInfoResponse;
import com.antfin.ledgerdb.client.common.GetProofResponse;
import com.antfin.ledgerdb.client.common.SignerProfile;
import com.antfin.ledgerdb.client.crypto.ECCK1KeyPair;
import com.antfin.ledgerdb.client.crypto.SignerKeyPair;
import com.antfin.ledgerdb.client.proto.Sender;
import org.bouncycastle.util.encoders.Hex;
import org.junit.Test;
public class GetProofDemo {
   String ledgerId = "******";
   String memberId = "******";
   String privateKeyInHex = "your private key in hex";
   String publicKeyInHex = "your public key in hex";

   byte[] privateKeyInBytes = Hex.decode(privateKeyInHex);
   byte[] publicKeyInBytes = Hex.decode(publicKeyInHex);
   String ledgerHost = "127.0.0.1";
   Integer ledgerPort = 10099;
   LedgerDBLightClient client = new LedgerDBLightClient(ledgerHost, ledgerPort);
   SignerKeyPair signerKeyPair = new ECCK1KeyPair(privateKeyInBytes, publicKeyInBytes);
   @Test
   public void test() {
      OperationControl op = new OperationControl();
      op.addSignerProfile(new SignerProfile( memberId,  Sender.SenderType.REGULAR, signerKeyPair));
      String ledgerUri = "ledger://127.0.0.1:10099/*****";
      GetProofResponse response = client.getProof(ledgerUri, 5, op);
   }
}
```

ECCK1KeyPair 秘钥对生成

```
import com.antfin.ledgerdb.client.crypto.ECCK1KeyPair;
import com.antfin.ledgerdb.client.crypto.Keys;
import org.bouncycastle.util.encoders.Hex;
import org.junit.Test;
import java.security.InvalidAlgorithmParameterException;
import java.security.KeyPair;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
public class KeyGenTest{
   @Test
   public void testGen() throws
         InvalidAlgorithmParameterException,
         NoSuchAlgorithmException,
         NoSuchProviderException{

    ECCK1KeyPair key = Keys.createEcKeyPair();
    // 64 bytes
    byte[] publicKeyInBytes = key.getPublicKey();
    // 32 bytes
    byte[] privateKeyInBytes = key.getPrivateKey();
    System.out.println(Hex.toHexString(publicKeyInBytes));
    System.out.print(Hex.toHexString(privateKeyInBytes));
  }
  @Test
  public void testConstructors() throws
        InvalidAlgorithmParameterException,
        NoSuchAlgorithmException,
        NoSuchProviderException{

    ECCK1KeyPair key = Keys.createEcKeyPair();
    // 64 bytes
    byte[] publicKeyInBytes = key.getPublicKey();
    // 32 bytes
    byte[] privateKeyInBytes = key.getPrivateKey();
    // construct from private key in bytes
    ECCK1KeyPair keyConstructed = new ECCK1KeyPair(privateKeyInBytes);
    // construct from private key in bytes and public key in bytes
    // public key should be 64 bytes
    ECCK1KeyPair keyConstructed2 = new ECCK1KeyPair(privateKeyInBytes, publicKeyInBytes);}
}
```