天翼云
State Cloud

对象存储（经典版）I 型

(Object-Oriented Storage,OOS)

开发者文档 V6

天翼云科技有限公司

目录

1 产品简介

1.1 产品介绍

对象存储（经典版）I型（Object-Oriented Storage，OOS）为客户提供一种海量、弹性、廉价、高可用的存储服务。客户只需花极少的钱就可以获得一个几乎无限的存储空间，可以随时根据需要调整对资源的占用，并只需为真正使用的资源付费。

OOS 提供了基于 Web 门户和基于 HTTP REST 接口两种访问方式，用户可以在任何地方通过互联网对数据进行管理和访问。OOS 提供的 REST 接口与 Amazon S3 兼容，因此基于 OOS 的业务可以非常轻松的与 Amazon S3 对接。您也可以通过 OOS 提供的 SDK 来调用 OOS 服务。

用户可以根据需要，选择使用大陆的**对象存储网络**、**对象存储网络 2**，或**香港**节点。

1.2 产品优势

- 容量几乎没有上限，弹性扩容。
- 自动数据冗余和故障切换，确保高可用。
- 流量按需计费，节省带宽成本。
- 易于管理、维护和升级。

1.3 存储类型

OOS 提供两种类型的存储：标准存储和低频访问存储。用户可以根据不同业务场景选择不同的存储类型。

- **标准存储（STANDARD）：**访问时延低、吞吐量高，能够有效支持各种热点类型数据频繁访问。适用于各种音视频服务、图片服务、大型网站、大数据分析等应用的数据存储。标准存储是默认的存储类型。如果上载文件时未指定存储类型，OOS 默认使用标准存储。
- **低频访问存储（STANDARD_IA）：**适合长期保存不经常访问的数据。对于不经常访问但仍需要实时访问的数据，可以采用低频访问存储，例如各类移动应用、智能设备、企业数据的长期备份。低频访问存储的文件有最短存储时间，存储时间短于 30 天的文件被

提前删除或变更时，会产生一定费用。低频访问存储文件有最小计费大小，即如果文件大小低于 64KiB，会按照 64KiB 计算收费，文件大于等于 64KiB 按照实际存储收费。数据获取时会产生数据取回费用。

存储类型的对比

对比指标	标准存储类型	低频访问存储类型
数据持久性高达	99.99999999999%（13 个 9）	99.99999999999%（13 个 9）
服务设计的可用性	99.99%	99.9%
文件最小计费大小	按照文件实际大小计算	64KiB
最短存储时间	无最短存储时间要求	30 天
数据取回费用	不收取数据取回费用	按实际获取的数据量收取，单位 GiB
数据访问特点	实时访问	实时访问
图片处理	支持	支持
HTTPS 加密传输	支持	支持
修改存储类型	支持	支持

存储类型转换

支持文件的存储类型之间相互转：

- 标准存储转换为低频访问存储：**使用 PUT Bucket Lifecycle** 设置生命周期规则，或使用 **PUT Object - Copy** 修改请求头中的 x-amz-storage-class，可以将标准存储转换为低频访问存储。
- 低频访问存储转换为标准存储：使用 **PUT Object - Copy** 修改请求头中的 x-amz-storage-class，将低频访问存储修改为标准存储。

2 主要概念

面向对象存储的主要概念有：Account（账户）、Service（服务）、Bucket（存储桶）和Object（文件）。它们之间的关系如图 1 所示。在使用 OOS 之前，首先需要在天翼云网站注册一个 Account（账户）。注册成功之后，联系天翼云客服工作人员开通 OOS 服务，OOS 会为该账户提供服务（Service），在该服务下，用户可以创建 1 个或多个 Bucket（存储桶），每个存储桶中可以存储不限数量的 Object（文件）。

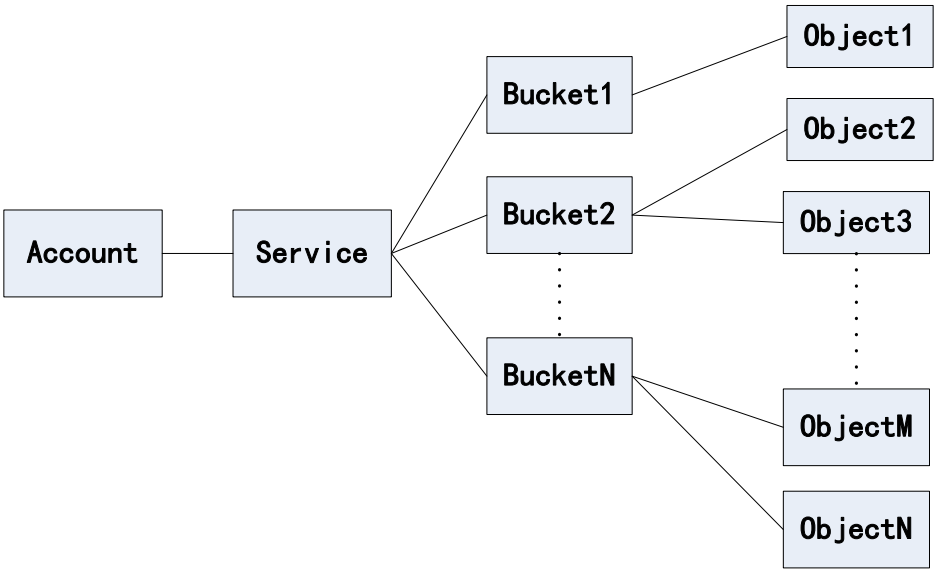


图 1　OOS 主要概念之间的关系

2.1 Account

在使用 OOS 之前，需要在天翼云网站注册一个 Account（账户）。注册时邮箱、密码和手机号码是必填项。正确填写所需信息并进行实名认证之后，联系天翼云客服人员（客服电话：400-810-9889）申请开通 OOS 服务。开通 OOS 服务成功之后，用户可以用该账户登录并使用 OOS 服务。

2.2 Service

Service 是 OOS 为注册成功用户提供的服务，该服务为用户提供弹性可扩展的存储空间，用户可以根据自己的业务需要建立 1 至 10 个存储桶（Bucket）。

2.3 Bucket

存储桶（Bucket）是存储文件（Object）的容器。OOS 的每个文件（Object）都必须包含在一个存储桶中。OOS 提供的是基于桶和文件的扁平化存储方式，桶中的所有文件都处于同一逻辑层级，去除了文件系统中的多层级树形目录结构。

您可以设置存储桶的属性，用来控制数据存储位置、访问权限、生命周期等，这些属性设置直接作用于该存储桶内的所有文件，因此您可以通过灵活的属性设置，来创建不同的存储桶，完成不同的管理功能。每个用户最多可以建立 10 个存储桶。用户只有对 Bucket 拥有相应的权限，才可以对其进行操作，这样保证了数据的安全性，防止非授权用户的非法访问。

2.3.1 Bucket 的命名规范

Put Bucket 用来创建一个存储桶（Bucket）。

存储桶（Bucket）的命名规范如下：

- 存储桶名称必须全局唯一。
- 存储桶名称长度介于 3 到 63 字节之间。
- 存储桶名称只能由小写字母、数字、短横线（-）和点（.）组成。
- 存储桶名称可以由一个或者多个小节组成，小节之间用点（.）隔开，各个小节需要：
  - 只能包含小写字母、数字和短横线（-）。
  - 必须以小写字母或者数字开始。
  - 必须以小写字母或者数字结束。
- 存储桶名称不能是一组或多组“数字.数字”的组合（如 192.162.0.1）。
- 存储桶名称中不能包含双点（..）、横线点（-.）和点横线（.-）。
- 不允许使用非法敏感字符，例如暴恐涉政相关信息等。

2.3.2 存储桶（Bucket）的基本操作

新建存储桶（Bucket）时需要输入存储桶的名称，选择操作权限（Bucket 的默认操作权限是 private）。存储桶创建成功后，存储桶名称不能进行修改。根用户和拥有存储桶相应权限的子用户可以更改存储桶的操作权限。

删除存储桶：只有根用户和拥有存储桶删除权限的子用户才能删除存储桶，且被删除的存储桶中不能包含任何文件。

查看属性：查看存储桶的属性时，用户可以更改 Bucket 的操作权限。

2.3.3 存储桶（Bucket）的数据位置和索引位置

在创建存储桶（Bucket）时，需要指定文件的数据位置和索引位置，数据位置指是存放文件数据的位置，索引位置是指存放文件数据索引信息的位置。

在创建存储桶（Bucket）时，对于数据位置用户可以：

- 选择**就近写入**，即 OOS 将数据写到离用户访问点最近的资源池中。
- 指定具体的资源池，OOS 会将数据按顺序写入用户指定的资源池。

创建存储桶（Bucket）时，需要指定数据索引位置，如果不指定，默认索引位置为指定数据位置的第一个。

用户也可以设置 OOS 数据调度策略，让 OOS 灵活处理数据位置存储：

- 允许自动调度：OOS 可以根据用户选择地区的实际使用情况，自动进行数据存储位置的调度，以便为用户提供更快的访问速度。
- 不允许自动调度：用户数据只能存放在指定的数据位置。

2.4 Object

对象（Object）是用户存储在 OOS 上的数据基本单元，也被称为 OOS 的文件。文件可以是文本、图片、音频、视频或者网页。OOS 支持的单个文件的大小从 1 字节到 5T 字节。

用户可以上传、下载、删除和共享文件。同时用户还可以对文件的组织形式进行管理，将文件移动或者复制到目标目录下。

2.5 通过 IPv6 访问 OOS

OOS 除了支持通过 IPv4 协议访问以外，还支持通过 IPv6 协议访问。当用户使用 IPv6 访问 Bucket 时：

- 客户端和网络必须都支持IPv6。
- 在Bucket Policy中设置ip地址的黑白名单时，可以使用IPv6。用户可以将Bucket Policy的Condition元素同时设置为包含IPv4和IPv6地址。例如

```
"Condition": {
    "IpAddress": {
        "ctyun:SourceIp": [
```

```
            "54.240.143.0/24",
            "2001:DB8:1234:5678::/64"
        ]
    }
}
```

- 当用户使用IPv6访问时，Bucket日志中记录的IP地址（Remote IP字段）是IPv6格式的。

2.6 统计

2.6.1 什么是统计分析

统计分析指用户可以查询指定 Bucket、指定数据位置的数据使用情况，根据统计分析数据，采取对应的措施。

2.6.2 基本概念

- **直接流量-互联网流量**：从互联网上传下载文件，并且未经对象存储网络内部调度产生的流量。
- **直接流量-非互联网流量**：从非互联网（例如内网）上传下载文件，并且未经对象存储网络内部调度产生的流量。
- **漫游流量-互联网流量**：从互联网上传下载文件，并且经过对象存储网络内部调度产生的流量。
- **漫游流量-非互联网流量**：从非互联网（例如内网）上传下载文件，并且经过对象存储网络内部调度产生的流量。
- **删除容量**：已删除文件的大小。

2.7 操作跟踪

2.7.1 什么是操作跟踪

操作跟踪用于记录 OOS 账户的管理事件，并将产生的跟踪日志保存到指定的 OOS 存储桶中。记录的信息包括用户的身份，请求时间，源 IP 地址，请求参数以及服务返回的响应结果等。

操作跟踪功能主要包括：

- **管理事件记录**：用户通过操作跟踪功能可以查看近 6 个月内的管理事件，包括：登录，退出，查看、创建、修改和删除资源。
- **跟踪日志**：当账户中发生一个管理事件时，OOS 会根据配置的跟踪参数与管理事件进行匹配，当与跟踪参数相匹配时，管理事件会以日志的形式存储到用户配置的存储桶中，即跟踪日志。

2.7.2 基本概念

2.7.2.1 管理事件

在账户中执行的 Bucket 操作、管理 API 操作、IAM 操作、操作跟踪均属于管理事件。

2.7.2.2 读事件

读事件为可以查看和读取资源，但不会对资源进行更改的操作。

2.7.2.3 写事件

写事件为可以修改资源的操作，包括创建、修改和删除操作。

2.8 访问控制

2.8.1 什么是 IAM

统一身份认证（Identity and Access Management，简称 IAM）是 OOS 为用户提供的用户身份管理与访问控制服务，您可以使用 IAM 创建、管理用户账号，并对这些账号进行权限分配，方便资源管理。

您只需为您在天翼云账户中的资源付费，无需为 IAM 单独付费。

2.8.2 基本概念

2.8.2.1 根用户

用户首次创建 CTYUN 账户时，最初使用的是一个对账户中所有服务和资源有完全访问权限的登录身份，此身份称为根用户。

2.8.2.2 IAM 用户

IAM 用户是 OOS 中的一个实体，该实体代表使用它与 OOS 进行交互的人员或应用程序，由 CTYUN 账户在 OOS 中创建的用户，也称为子用户。默认情况下，全新的 IAM 用户没有执行任何操作的权限，无权执行任何 OOS 操作或访问任何 OOS 资源。

2.8.2.3 用户组

用户组是用户的集合，IAM 可以将 IAM 用户添加到对应的用户组，通过对用户组进行授权管理 IAM 用户，用户组的权限会影响用户组内的 IAM 用户。建议具备相同权限的 IAM 用户添加到同一用户组，方便管理。同一个 IAM 用户可以同时加入多个用户组。

2.8.2.4 MFA

多因素认证（Multi-Factor Authentication，简称 MFA）是一种简单安全的二次认证方式，为用户增加了一层安全保护。

2.8.2.5 授权

通过给用户组和用户附加策略，用户就能获得策略中定义的权限，这一过程称为授权。

2.8.2.6 策略

策略是以 JSON 格式描述权限信息的集合，可以精确地描述涵盖的资源集、操作集以及授权条件。

支持系统策略和自定义策略：

- 系统策略：OOS 预先创建好的策略，用户可以根据自身需求，直接引用。对于系统策略，用户只能使用，不能修改。
- 自定义策略：用户自己创建的策略，用户可以对该类型策略进行修改和删除。

2.9 Endpoint 列表

对象存储网络、对象存储网络 2、香港节点的 Endpoint 不同。

2.9.1 对象存储网络

对象存储网络中的各个地区，使用统一的 OOS API、统计、操作跟踪和 IAM API 的 Endpoint。对象存储网络 Endpoint 列表如下：

- OOS API Endpoint: oos-cn.ctyunapi.cn，支持 HTTP 和 HTTPS。
- 统计 API Endpoint: oos-cn-mg.ctyunapi.cn，支持 HTTP 和 HTTPS。
- 操作跟踪 API Endpoint：oos-cn-cloudtrail.ctyunapi.cn，支持 HTTPS。
- IAM Endpoint: oos-cn-iam.ctyunapi.cn，支持 HTTPS。

说明：对于对象存储网络中的 OOS API，如果您的数据存储在某个资源池，建议您直接使用该资源池的 Endpoint。Endpoint 列表如下（Endpoint 列表仅为资源池 Endpoint 访问信息描述，与资源状态无关联）：

地区	OOS API Endpoint
郑州	oos-hazz.ctyunapi.cn
沈阳	oos-lnsy.ctyunapi.cn
四川成都	oos-sccd.ctyunapi.cn
乌鲁木齐	oos-xjwlmq.ctyunapi.cn
甘肃兰州	oos-gslz.ctyunapi.cn
山东青岛	oos-sdqd.ctyunapi.cn
贵州贵阳	oos-gzgy.ctyunapi.cn
湖北武汉	oos-hbwh.ctyunapi.cn
西藏拉萨	oos-xzls.ctyunapi.cn
安徽芜湖	oos-ahwh.ctyunapi.cn
广东深圳	oos-gdsz.ctyunapi.cn
江苏苏州	oos-jssz.ctyunapi.cn
上海 2	oos-sh2.ctyunapi.cn

2.9.2 对象存储网络 2

对象存储网络 2 中的各个地区，使用统一的 OOS API、统计、操作跟踪和 IAM API 的 Endpoint。对象存储网络 2 Endpoint 列表如下：

- OOS API Endpoint: oos-cn2.ctyunapi.cn，支持 HTTP 和 HTTPS。
- 统计 API Endpoint: oos-cn2-mg.ctyunapi.cn，支持 HTTP 和 HTTPS。
- 操作跟踪 API Endpoint：oos-cn2-cloudtrail.ctyunapi.cn，支持 HTTPS。
- IAM Endpoint: oos-cn2-iam.ctyunapi.cn，支持 HTTPS。

说明：对于对象存储网络 2 中的 OOS API，如果您的数据存储在某个资源池，建议您直接使用该资源池的 Endpoint。Endpoint 列表如下（Endpoint 列表仅为资源池 Endpoint 访问信息描述，与资源状态无关联）：

地区	OOS API Endpoint
内蒙古 1	oos-nm1.ctyunapi.cn
杭州 1	oos-hz1.ctyunapi.cn

2.9.3 香港节点

香港节点分为**香港精品网络**和**香港普通网络**两种方式，精品网和普通网 OOS API 的 Endpoint 不同，统计、操作跟踪和 IAM API 的 Endpoint 相同：

- 香港精品网 OOS API Endpoint: oos-cnhk-hqnet.ctyunapi.cn，香港普通网 OOS API Endpoint：oos-cnhk-nqnet.ctyunapi.cn。支持 HTTP 和 HTTPS。
- 统计 API Endpoint: oos-cnhk-mg.ctyunapi.cn，支持 HTTP 和 HTTPS。
- 操作跟踪 API Endpoint：oos-cnhk-cloudtrail.ctyunapi.cn，支持 HTTPS。
- IAM Endpoint：oos-cnhk-iam.ctyunapi.cn，支持 HTTPS。

3 安全策略

为了保证用户数据的安全，OOS 提供以下安全策略来保障用户数据的安全性：用户签名验证、Bucket 权限控制、访问控制、Bucket Policy 安全策略、合规保留。

3.1 用户签名验证(V2)

为了防止用户数据被他人盗取，所有发送到 OOS 的非匿名请求都需要经过签名验证。OOS 通过使用 AccessKeyID/SecretAccessKey 对称加密的方法来验证某个请求的发送者身份。AccessKeyID 和 SecretAccessKey 在 OOS 服务开通后自动随机生成。当用户想以个人身份向 OOS 发送请求时，需要首先将发送的请求按照 OOS 指定的格式生成签名字符串。然后使用 SecretAccessKey 对签名字符串进行加密产生验证码。OOS 收到请求以后，会通过 AccessKeyID 找到对应的 SecretAccessKey，以同样的方法提取签名字符串和验证码，如果计算出来的验证码和提供的一样，即认为该请求是有效的。否则，OOS 将拒绝处理这次请求，并返回错误码。

3.1.1 Header 中包含签名

用户可以在 HTTP 请求中增加 Authorization（授权）的 Header 来包含签名信息，表明这个消息已被授权。如果用户的请求中没有 Authentication 字段，则认为是匿名访问。

验证码计算方法如下：

```
Authorization: "AWS " + AccessId + ":" + Signature
Signature =Base64( HMAC-SHA1( YourSecretAccessKey, UTF-8-Encoding-
Of( StringToSign ) ) ) ;
StringToSign = HTTP-VERB + "\n" +
Content-MD5 + "\n" +
Content-Type + "\n" +
Date + "\n" +
CanonicalizedAmzHeaders +"\n" +
CanonicalizedResource;

CanonicalizedAmzHeaders = [详见下列描述];
CanonicalizedResource = [ "/" + Bucket ] +
<HTTP-Request-URI > +
  [<sub-resource>]
```

注：sub-resource 如果存在，可参与签名的参数包括：acl、torrent、logging、location、policy、requestPayment、versioning、versions、versionId、notification、uploadId、uploads、partNumber、website、delete、lifecycle、tagging、cors、restore、inventory。

HMAC-SHA1 是由 RFC 2104 定义的算法。该算法需要输入两个字符串：一个 key 和一个 message。OOS 在验证请求时，使用您的 SecretAccessKey 作为 key，使用 UTF-8 编码的 StringToSign 作为 message。HMAC-SHA1 的输出同样是个字符串，被称为摘要。Signature 请求参数是这个摘要的 Base64 编码。

说明：

- **Content-MD5** 表示请求内容数据的 MD5 值。
- **Content-Type** 表示请求内容的类型。
- **Date** 表示此次操作的时间，且必须为 HTTP1.1 中支持的 GMT 格式。
- **CanonicalizedAmzHeaders** 表示 http 中的 object user meta 组合。
- **CanonicalizedResource** 表示用户要访问的 OOS 资源。

3.1.1.1 使用 Base64 编码

HMAC 请求签名必须使用 Base64 编码。Base64 编码将签名转换为简单的能附加到请求中的 ASCII 编码字符串。加号和斜杠这两个字符不能直接在 URL 中使用，必须经过编码。比如，如果授权编码包括加号，在 URL 中把它编码成为%2B。

3.1.1.2 StringToSign

StringToSign 中不包含 Content-Type, Date, Content-MD5 这些请求头的名字，只包含这些请求头的值。但是以“x-amz”开头的请求头的名字和值都包含在 StringToSign 中。

如果在请求中，Content-Type, Content-MD5 等请求头不存在，那么该位置用空串（""）来代替。

3.1.1.3 时间戳说明

在签名时必须使用时间戳，可以用 HTTP 的 Date 请求头，也可以用 x-amz-date 请求头。时间戳中的时间和 OOS 的系统时间不能相差 15 分钟以上，否则，OOS 会返回错误码 RequestTimeTooSkewed。

有些 HTTP 客户端不能设置 Date 请求头，如果你在签名时设置 Date 请求头有困难，那么你可以使用 x-amz-date 请求头来传送时间戳。x-amz-date 请求头需要符合 RFC 2616 的格式 (http://www.ietf.org/rfc/rfc2616.txt)。当请求中包含 x-amz-date 头时，OOS 在计算签名时会忽略 Date 头。所以如果请求中有 x-amz-date 头，在客户端计算签名时，Date 头的值应设置为空串。

3.1.1.4 构建 CanonicalizedAMZHeaders 的方法

所有以“x-amz-”为前缀的 HTTP Header 被称为 CanonicalizedAMZHeaders。它的构建方法如下：

1) 将所有以“x-amz-”为前缀的 HTTP 请求头的名字转换成小写字母。如'X-AMZ-Meta-Name: fred'转换成'x-amz-meta-name: fred'。

2) 将上一步得到的所有 HTTP 请求头按照字典序进行升序排列。

3) 如果有相同名字的请求头，则根据标准 RFC 2616, 4.2 章进行合并（两个值之间只用英文逗号分隔）。如有两个名为'x-amz-meta-name'的请求头，对应的值分别为'fred'和'barney'，则合并后为：'x-amz-meta-name:fred,barney'。

4) 删除分隔符（:）两端和值两端出现的任何空格。如'x-amz-meta-name : fred '转换成：'x-amz-meta-name:fred'。

5) 将所有的请求头和值用'\n'分隔符分隔拼成最后的 CanonicalizedAMZHeader。

3.1.1.5 构建 CanonicalizedResource 的方法

用户发送请求中访问的 OOS 目标资源被称为 CanonicalizedResource。它的构建方法如下：

1) 将 CanonicalizedResource 置成空字符串（“”）。

2) 如果请求通过 Host 请求头来指定 Bucket，那么增加 Bucket 名，并在其前面加上“/”(例如/bucketname)。对于 Bucket 名称在 path 中的请求，或者没有指定 Bucket 的请求，不做任何操作。

3) 在 path 中增加未编码的 HTTP 请求 URI，到请求参数处截止，不包含请求参数。

4) 如果请求包含子资源，例如?acl, ?website, ?logging，那么将所有的子资源按照字典序，从小到大排列并以’&’为分隔符生成子资源字符串。在 CanonicalizedResource 字符串尾添加“？”和子资源字符串。在构造 CanonicalizedResource 时，必须包含的子资源，OOS 支持的子资源包括：acl、torrent、logging、location、policy、requestPayment、versioning、versions、versionId、notification、uploadId、uploads、partNumber、website、delete、lifecycle、tagging、cors、restore、inventory。

如果请求通过参数指定了要覆盖的响应头，那么在 CanonicalizedResource 后加上该请求参数和值。当进行签名时，不要对这些值进行编码。但是当发送请求的时候，用户必须对这些参数值进行编码。GET 请求中的这些参数包括：response-content-type,response-content-language, response-expires, response-cache-control,response-content-disposition, response-content-encoding。

例如：/BucketName/ObjectName?response-content-type=ContentType

当发送批量删除文件请求时，delete 参数需要包含在 CanonicalizedResource 中。

3.1.1.6 示例

以下是使用 V2 签名的示例，示例中使用的访问密钥如下：

参数	值
AccessKeyID	3a7451ae6b635b4f5ded
SecretAccessKey	c458417af3507ca686128f54efb3a00d5ad7ff09

1. GET Object

从名为 example-bucket 的 Bucket 中 get 文件

请求	StringToSign
GET /photos/puppy.jpg HTTP/1.1 Host: example-bucket.oos-cn.ctyunapi.cn	GET\n \n

Date: Tue, 11 Jun 2024 01:32:55 GMT Content-Type: application/octet-stream Authorization: AWS 3a7451ae6b635b4f5ded:icJnqU3Zfm1sEOBCBwJPKymwWds=	application/octet-stream\n Tue, 11 Jun 2024 01:32:55 GMT\n /example-bucket/photos/puppy.jpg

注意：CanonicalizedResource 中包含 Bucket 名称，但是 HTTP 请求 URI 中没有，因为 Bucket 名称是在 Host 请求头中指定的。请求中没有 Content-MD5 请求头，所以 StringToSign 中是空行。

2. PUT Object

向名为 example-bucket 的 Bucket 中上传一个文件。

请求	StringToSign
PUT /photos/puppy.jpg HTTP/1.1 Host: example-bucket.oos-cn.ctyunapi.cn Date: Tue, 11 Jun 2024 01:43:59 GMT Content-Type: image/jpeg Content-MD5: ICy5YqxZB1uWSwcVLSNLcA== Content-Length: 94328 Authorization: AWS 3a7451ae6b635b4f5ded:MHUV0HaL8UiNe/VPNbWg06PppEI=	PUT\n ICy5YqxZB1uWSwcVLSNLcA==\n image/jpeg\n Tue, 11 Jun 2024 01:43:59 GMT\n /example-bucket/photos/puppy.jpg

注意：Content-Type、Content-MD5 请求头包含在请求中，也包含在 StringToSign 中。

3. List Objects

list 名为 example-bucket 的 Bucket 的文件。

请求	StringToSign
GET /?prefix=photos&max-keys=50&marker=puppy HTTP/1.1 Host: example-bucket.oos-cn.ctyunapi.cn Date: Tue, 11 Jun 2024 01:59:59 GMT Content-Type: application/octet-stream User-Agent: Mozilla/5.0 Authorization: AWS 3a7451ae6b635b4f5ded:kitekL1v232x7FYLUUi7y2kPC9g=	GET\n \n application/octet-stream\n Tue, 11 Jun 2024 01:59:59 GMT\n /example-bucket/

注意：CanonicalizedResource 的结尾要有斜杠/，查询字符串为空。

4. 获取 ACL

下面的例子是获取名为 example-bucket 的 Bucket 的访问控制权限配置信息。

请求	StringToSign
GET /?acl HTTP/1.1 Host: example-bucket.oos-cn.ctyunapi.cn Date: Tue, 11 Jun 2024 02:06:03 GMT Content-Type: application/octet-stream Authorization: AWS 3a7451ae6b635b4f5ded:7x+mp5y3YFS6BC9pdPiqsevbjb4=	GET\n \n application/octet-stream\n Tue, 11 Jun 2024 02:06:03 GMT\n /example-bucket/?acl

注意：在 CanonicalizedResource 中是如何包含子资源查询字符串参数的。

5. Delete Object

从名为 example-bucket 的 Bucket 中删除文件。Bucket 在 path 中指定，并使用 x-amz-date 请求头。

请求	StringToSign
DELETE /example-bucket/photos/puppy.jpg HTTP/1.1 Host: oos-cn.ctyunapi.cn Date: Tue, 11 Jun 2024 06:47:39 GMT x-amz-date: Tue, 11 Jun 2024 06:37:21 GMT Authorization: AWS 3a7451ae6b635b4f5ded:0kgBoDiPB3sQAy+Ole+oKcH+QRE=	DELETE\n \n \n \n x-amz-date:Tue, 11 Jun 2024 06:37:21 GMT\n /example-bucket/photos/puppy.jpg

注意：此请求使用 x-amz-date 请求头来替代 Date 请求头，在 StringToSign 中，实际的 Date 请求头被设置成空行。

6. 使用 CNAME 形式上传文件

下面的例子通过 CNAME 形式上传文件，并使用自定义元数据。其中：Bucket 为 example-bucket，Bucket 绑定的自定义域名为 oos11.ctyun.cn。

请求	StringToSign
PUT /example-bucket/db-backup.dat.gz HTTP/1.1 Host: oos11.ctyun.cn Date: Tue, 11 Jun 2024 07:18:11 GMT content-type: application/x-download Content-MD5: ICy5YqxZB1uWSwcVLSNLcA== X-Amz-Meta-ReviewedBy: joe X-Amz-Meta-FileChecksum: 0x02661779	PUT\n ICy5YqxZB1uWSwcVLSNLcA==\n application/x-download\n Tue, 11 Jun 2024 07:18:11 GMT\n x-amz-meta-checksumalgorithm:crc32\n x-amz-meta-filechecksum:0x02661779\n x-amz-meta-reviewedby:joe\n

X-Amz-Meta-ChecksumAlgorithm: crc32 Content-Disposition: attachment; file name=database.dat Content-Encoding: gzip Content-Length: 3 Authorization: AWS 3a7451ae6b635b4f5ded:Wdqh0EKuT5lUZioWfc0rk2a6Arg=	/example-bucket/db-backup.dat.gz

注意：“x-amz -”请求头被排序了，空白行被删除，转换为小写字符，多个同名的请求头使用逗号分隔的方式被加入。只有 Content-Type, Content-MD5 请求头被加入到了 StringToSign 中，但是其他的 Content-*请求头没有被加入。

7. List Buckets

请求	StringToSign
GET / HTTP/1.1 Host: oos-cn.ctyunapi.cn Date: Tue, 11 Jun 2024 03:35:03 GMT Authorization: AWS 3a7451ae6b635b4f5ded:MTxKel9VvMQGamBD1gQXJ5ttm5c=	GET\n \n \n Tue, 11 Jun 2024 03:35:03 GMT\n /

8. 文件名被编码

请求	StringToSign
GET /dictionary/fran/123%E5%92%8C123 HTTP/1.1 Host: example-bucket.oos-cn.ctyunapi.cn Date: Tue, 11 Jun 2024 05:35:27 GMT Authorization: AWS 3a7451ae6b635b4f5ded:owSmnJIMATp1GdDpXtw72QXJ7x0=	GET\n \n \n Tue, 11 Jun 2024 05:35:27 GMT\n /example-bucket/dictionary/fran/123%E5%92%8C123

StringToSign 中从请求 URI 获取的元素，是按原样获取的，包括 URL 编码和大小写。

3.1.2 使用查询参数验证

除了授权头以外，用户可以通过 URL 中加入查询参数来表达签名信息的方式，将该 URL 转给第三方实现授权访问，也称为预签名 URL。这对于使用第三方浏览器获取对象存储数据的用户非常有用，不需要代理请求。这种方式通过构造并加密一个终端用户浏览器可以检索到的预签名的请求来实现，同时设置过期时间来标明预签名的有效期。

预签名 URL 支持 GET 请求和 POST 请求（POST Object 除外）。

注意：使用预签名 URL 方式，有将您授权的数据在过期时间内曝露在互联网上的风险，建议您预先评估后使用。

URL 中包含签名的示例如下：

```
http://oos-cn.ctyunapi.cn/quotes/n-*elson?AWSAccessKeyId=44CF9590006BF252F707&Expires=1141889120&Signature=vjbyPxybdZaNmGa%2ByT272YEAiv4%3D
```

注意：在 URL 中实现签名，必须至少包含 Signature，Expires，AWSAccessKeyId 三个参数。

请求参数

参数	描述	是否必须
AWSAccessKeyId	用户的 AccessKeyID 信息，可以登录到可以登录到对象存储（经典版）I 型控制中心，点击“访问控制”>“安全凭证”>“密钥”查看；或者调用 ListAccessKey 查看。 类型：字符串。	是
Expires	签名过期时间，按照秒来计算的。服务器端超过这个时间的请求将被拒绝。 类型：字符串。	是
Signature	URL 是按照 HMAC-SHA1 的 StringToSign 的 Base64 编码方式编码。 类型：字符串。	是

Expires 是一个 UNIX 时间（自 UTC 时间 1970 年 1 月 1 号开始的秒数），用于标识该 URL 的超时时间。如果 OOS 接收到这个 URL 请求的时候晚于签名中包含的 Expires 参数时，则返回请求超时的错误码。例如：当前时间是 1141889060，开发者希望创建一个 60 秒后自动失效的 URL，则可以设置 Expires 时间为 1141889120。

URL 中包含签名的算法和 Header 中包含签名的算法基本一样，主要区别如下：

- 通过 URL 包含签名时，之前的 Date 参数换成 Expires 参数。
- 如果在 URL 和 Header 中同时包含签名，以 Header 中的签名为准。
- 对于 URL 预签名，如果传入的 Signature，Expires，AWSAccessKeyId 出现不止一次，以第一次为准。
- 对于 URL 预签名，请求先验证请求时间是否晚于 Expires 时间：
  - 如果请求时间晚于 Expires 时间，则返回错误响应码。
  - 如果请求时间不晚于 Expires 时间，则验证签名。

3.2 用户签名验证（V4）

OOS 除了支持 **3.1 用户签名验证（V2）**中所述的签名算法外，还兼容 Amazon S3 Version 4 签名算法。V4 签名比 V2 签名安全性更高。

3.2.1 Signature Version 4 的工作原理

客户端按照如下步骤创建签名：

1. 创建规范请求。
2. 使用规范请求和其他信息创建待签名的字符串。
3. 使用 SecretAccessKey 派生签名密钥。然后将该签名密钥与在上一步中创建的字符串结合使用来创建签名。
4. 将生成的签名添加到 HTTP 请求头中或者作为参数添加到 URL 中。

OOS 服务收到请求后，将执行相同步骤来计算请求中发送的签名。之后，OOS 会将计算得到的签名与用户在请求中发送的签名进行比较。如果签名匹配，则处理请求。如果签名不匹配，则拒绝请求。

3.2.2 认证方法

用户可以使用以下方法来发送身份验证信息：

- HTTP 授权请求头- 使用 HTTP Authorization 请求头是验证 OOS 请求的最常用方法。所有 OOS REST 操作（使用 POST 请求的基于浏览器的上传除外）都需要此请求头。
- 查询字符串参数- 使用 URL 查询参数来提供请求信息，包括身份验证信息。由于请求签名是 URL 的一部分，因此这种类型的 URL 通常称为预签名 URL。

OOS 还支持使用基于浏览器的 HTTP POST 请求的上传方式。通过 HTTP POST 请求，用户可以直接通过浏览器将内容上传到 OOS。

3.2.3 使用 Authorization 请求头验证

3.2.3.1 概述

Authorization 请求头包含以下信息（增加换行是为了方便阅读，实际为空字符串）：

```
Authorization: AWS4-HMAC-SHA256
Credential=2a948fd3f00ba0925806/20180501/region/s3/aws4_request,
SignedHeaders=host;range;x-amz-date,
Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024
```

组成字段说明如下：

部分	描述
AWS4-HMAC-SHA256	用于签名的算法，固定值。
Credential	用户 AccessKeyID 和范围信息，范围信息包括请求日期、区域、服务、终止字符串 aws4_request，格式如下： <*AccessKeyID*>/<*date*>/<*region*>/<*service*>/aws4_request 其中： ● date 格式为 YYYYMMDD。

	● region： ■ 对于 oos api：访问域名为 oos-*xx*.ctyunapi.cn，region 为 *xx*。 ■ 对于统计 api：访问域名为 oos-*xx*-mg.ctyunapi.cn，region 为 *xx*-mg。 ■ 对于操作跟踪 api：访问域名为 oos-*xx*-cloudtrail.ctyunapi.cn，region 为 *xx*。 ■ 对于 iam api：访问域名为 oos-*xx*-iam.ctyunapi.cn，region 为 *xx*。 **说明**：各资源池的详细访问域名详见 **Endpoint 列表**。 ● service： ■ 若使用 OOS API 服务，service 为 s3。 ■ 若使用统计分析服务，service 为 s3。 ■ 若使用操作跟踪服务，service 为 cloudtrail。 ■ 若使用 IAM 服务，service 为 sts。
SignedHeaders	已签名请求头的列表。该列表只需包含请求头名字，用分号分隔，必须全部小写，并按字符顺序对其进行排序，示例如下： host;range;x-amz-date
Signature	计算出的 256 位签名信息，以 64 个小写十六进制字符串形式表示。

有两种签名计算方式：

- **签名负载方式**：用户可以选择计算整个负载（即请求体）的 checksum，并将其包含在签名计算中。这种方式提高了安全性，但用户需要读取两次负载或将其缓冲在内存中。我们建议用户包含负载 checksum 以增强安全性。
- **无签名负载方式**：在签名计算中不包括负载的 checksum。将值设置为文本字符串 UNSIGNED_PAYLOAD，直接作为签名计算。

上述两种方式，都必须携带 x-amz-content-sha256 请求头，如果选择签名负载方式，请将 x-amz-content-sha256 请求头的值设置为负载的 checksum 值，否则将值设置为文本字符串 UNSIGNED-PAYLOAD。

3.2.3.2 签名过程

签名过程如下图所示：

下表描述了图中显示的功能。用户需要为这些函数实现代码。

功能	描述
Lowercase()	将字符串转换为小写。
Hex()	小写 16 进制编码。
SHA256Hash()	安全散列算法（SHA）加密散列函数。
HMAC-SHA256()	使用签名密钥，根据 SHA256 算法计算出的签名值。
Trim()	删除任何前导或尾随空格。
UriEncode()	URI 编码每个字节。UriEncode（）必须强制执行以下规则：

- 除下列字符外，其他字符进行 URI 编码：'A' - 'Z'，'a' - 'z'，'0' - '9'，' - '，'.'，'_'和'~'。
- 空格字符是保留字符，必须编码为“%20”（而不是“+”）。
- 每个 URI 编码字节由'%'和两位十六进制值组成。
- 十六进制值中的字母必须为大写，例如“%1A”。
- 除了文件名之外，对正斜杠字符'/'进行编码。例如，如果文件名称为 photos/Jan/sample.jpg，则不对名称中的正斜杠进行编码。

重要：

建议用户编写自己的自定义 UriEncode 函数，以确保您的编码可以正常工作。

以下是 Java 中的示例 UriEncode（）函数。

```
public static String UriEncode(CharSequence input, boolean
encodeSlash) {
        StringBuilder result = new StringBuilder();
        for (int i = 0; i < input.length(); i++) {
            char ch = input.charAt(i);
            if ((ch >= 'A' && ch <= 'Z') || (ch >= 'a' && ch <= 'z')
|| (ch >= '0' && ch <= '9') || ch == '_' || ch == '-' || ch == '~' ||
ch == '.') {
                result.append(ch);
            } else if (ch == '/') {
                result.append(encodeSlash ? "%2F" : ch);
            } else {
                result.append(toHexUTF8(ch));
            }
        }
        return result.toString();
    }
```

1．创建规范请求

将请求的内容（包括主机、操作、请求头等）组织为标准规范格式。规范请求是用于创建待签字符串的输入之一。伪代码如下：

```
CanonicalRequest =
  HTTPRequestMethod + '\n' +
  CanonicalURI + '\n' +
  CanonicalQueryString + '\n' +
  CanonicalHeaders + '\n' +
  SignedHeaders + '\n' +
  HexEncode(Hash(RequestPayload))
```

- *HTTPMethod* 是 HTTP 方法，例如 GET，PUT，HEAD 和 DELETE。
- *CanonicalURI* 是 URI 的绝对路径，以域名后面的“/”开头，直到字符串的末尾或者问号字符（'? '）截止。例如/example-bucket/myphoto.jpg
- *CanonicalQueryString* 是 URI 编码后的查询字符串参数。用户需要单独对参数名称和值进行 URI 编码。并需要按参数名称的字母顺序，对参数进行排序。排序在编码后进行。以下 URI 示例中的查询字符串是 prefix=somePrefix&marker=someMarker&max-keys=20：

```
http://oos-cn.ctyunapi.cn/example-bucket?prefix=somePrefix&marker=someMarker&max-keys=20
```

CanonicalQueryString 的构造方式如下（为了便于阅读，添加了换行符）：

```
UriEncode("marker")+"="+UriEncode("someMarker")+"&"+
UriEncode("max-keys")+"="+UriEncode("20") + "&" +
UriEncode("prefix")+"="+UriEncode("somePrefix")
```

当请求的目标是子资源时，相应的查询参数的值设置为空字符串（“”）。例如，下面的请求用于设置 Bucket 的 ACL 权限：

```
http://oos-cn.ctyunapi.cn/example-bucket?acl
```

在这种情况下，CanonicalQueryString 为：

```
UriEncode("acl") + "=" + ""
```

如果 URI 中不包含“？”，则请求中不存在查询字符串，此时将 CanonicalQueryString 设置为空字符串（“”），但仍需要包含“\ n”。

- *CanonicalHeaders* 是请求头的列表：
  - 各个请求头名称和值由换行符（“\ n”）分隔。
  - 请求头名称必须为小写。

- 如果有相同名字的请求头，则根据标准 RFC 2616, 4.2 章进行合并（两个值之间只用英文逗号分隔）。如有两个名为'x-amz-meta-name'的请求头，对应的值分别为'fred'和'barney'，则合并后为：'x-amz-meta-name:fred,barney'。
- 删除分隔符（:）两端和值两端出现的任何空格。如'x-amz-meta-name : fred '转换成：'x-amz-meta-name:fred'。
- 需要按字母顺序对请求头名称进行排序。

示例如下：

```
Lowercase(Trim(<HeaderName1>))+":"+Trim(<value>)+"\n"
Lowercase(Trim(<HeaderName2>))+":"+Trim(<value>)+"\n"
...
Lowercase(Trim(<HeaderNameN>))+":"+Trim(<value>)+"\n"
```

CanonicalHeaders 列表必须包括以下内容：

- HTTP Host 标头
- 如果存在 Content-Type 请求头，则必须将其添加到 CanonicalHeaders 列表中。
- 所有 x-amz-*请求头。例如，如果您使用的是临时安全凭据，则需要在请求中包含 x-amz-security-token。必须将此标题添加到 CanonicalHeader 列表中。

以下是 CanonicalHeaders 的示例，请求头名称为小写并已排序。

```
host:oos-cn.ctyunapi.cn
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130708T220855Z
```

- *SignedHeaders* 是按字母顺序排序的，以分号分隔的小写请求头名称列表。列表中的请求头与用户在 CanonicalHeaders 字符串中包含的请求头相同。例如，对于前面的示例，SignedHeaders 的值为：

```
host;x-amz-content-sha256;x-amz-date
```

- *RequestPayload* 是请求负载的 SHA256 哈希的十六进制值。

  如果请求中没有负载，则计算空字符串的哈希值，如下所示：

```
Hex(SHA256Hash(""))
```

  哈希返回以下值：

  e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855

例如：当用户使用 PUT 请求上传文件时，用户可以在请求体中提供文件数据。使用 GET 请求检索文件时，没有请求体，所以计算空字符串的哈希。

2．创建待签名字符串

待签名的字符串格式如下：

```
"AWS4-HMAC-SHA256" + "\n" +
timeStampISO8601Format + "\n" +<Scope> + "\n" +
Hex(SHA256Hash(<CanonicalRequest>))
```

常量字符串 AWS4-HMAC-SHA256 指定用户使用的哈希算法 HMAC-SHA256。

timeStamp 是 ISO 8601 格式的当前 UTC 时间（例如 20180501T000000Z）。

Scope 是将生成的签名绑定到指定日期，OOS 区域和服务。

```
date.Format(<YYYYMMDD>) + "/" + <region> + "/" + <service> + "/aws4_request"
```

3．计算签名

使用 SecretAccessKey 作为初始哈希操作的密钥，对请求日期、区域和服务执行一系列加密哈希操作（HMAC 操作），从而派生签名密钥。伪代码如下：

```
DateKey = HMAC-SHA256("AWS4"+"<SecretAccessKey>", "<YYYYMMDD>") DateRegionKey = HMAC-
SHA256(<DateKey>, "< region>") DateRegionServiceKey = HMAC-SHA256(<DateRegionKey>,
"<service>") SigningKey = HMAC-SHA256(<DateRegionServiceKey>, "aws4_request")
```

最终签名是使用签名密钥作为密钥，对待签名字符串计算得到 HMAC-SHA256 哈希值。伪代码如下：

```
HMAC-SHA256(SigningKey, StringToSign)
```

4．将签名信息添加到请求头

在计算签名后，将其添加到请求的 HTTP 请求头或查询字符串中。

将签名信息添加到 Authorization 标头的伪代码如下：

```
Authorization: <algorithm> Credential=<ak>/<credential_scope>,
SignedHeaders=<SignedHeaders>, Signature=<signature>
```

3.2.3.3 示例

以下是使用 V4 签名的示例。示例中使用的访问密钥如下：

参数	值
AccessKeyID	2a948fd3f00ba0925806
SecretAccessKey	ef2017c2e5ffa0b1761717ecbca021da16501384

Bucket 名称：example-bucket。

访问的域名是 oos-cn.ctyunapi.cn, region 是 cn。

1. 示例：GET 文件

从存储桶 example-bucket 中获取文件 test.txt 的前 10 个字节。请求如下：

```
GET /test.txt HTTP/1.1
x-amz-content-sha256: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
Authorization: SignatureToBeCalculated
x-amz-date: 20190220T060724Z
Range: bytes=0-9
Host: example-bucket.oos-cn.ctyunapi.cn
```

由于此 GET 请求不提供任何请求体内容，因此该 x-amz-content-sha256 请求头的值是空请求体的哈希值。以下步骤显示 Authorization 请求头的计算的方法。

1) StringToSign

a. 创建规范请求

```
GET
/test.txt

host:example-bucket.oos-cn.ctyunapi.cn
range:bytes=0-9
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20190220T060724Z

host;range;x-amz-content-sha256;x-amz-date
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

其中，最后一行是空请求体的 hash 值。第三行是空，因为此请求不包含请求参数。

b. 待签名字符串

```
AWS4-HMAC-SHA256
20190220T060724Z
20190220/cn/s3/aws4_request
a6417debbe1fe886b8ed84dca872475f7f09b01961af10d30fa601bc0986ba36
```

2) 生成签名密钥

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<SecretAccessKey>","20190220"),"cn"),"s3"),"aws4_request")
```

3) 计算后的签名

```
dcefeb864c1ffad98f8f0307af32ceb584b38dc2a9c7a65459363cdb03fc6f12
```

4) Authorization 请求头

```
Authorization: AWS4-HMAC-SHA256
Credential=2a948fd3f00ba0925806/20190220/cn/s3/aws4_request, SignedHeaders=host;range;x-
amz-content-sha256;x-amz-date,
Signature=dcefeb864c1ffad98f8f0307af32ceb584b38dc2a9c7a65459363cdb03fc6f12
```

2. 示例：PUT 文件

在存储桶 example-bucket 中上传文件 test.txt。

```
PUT /test.txt HTTP/1.1
x-amz-content-sha256: 7509e5bda0c762d2bac7f90d758b5b2263fa01ccbc542ab5e3df163be08e6ca9
Authorization:SignatureToBeCalculated
x-amz-date: 20190220T070722Z
x-amz-storage-class: STANDARD
Host: example-bucket.oos-cn.ctyunapi.cn
Content-Length: 12

hello world!
```

以下步骤显示 Authorization 请求头的计算的方法。

1) StringToSign

a. 创建规范请求

```
PUT
/test.txt
```

```
content-length:12
host:example-bucket.oos-cn.ctyunapi.cn
x-amz-content-sha256:7509e5bda0c762d2bac7f90d758b5b2263fa01ccbc542ab5e3df163be08e6ca9
x-amz-date:20190220T070722Z
x-amz-storage-class:STANDARD

content-length;host;x-amz-content-sha256;x-amz-date;x-amz-storage-class
7509e5bda0c762d2bac7f90d758b5b2263fa01ccbc542ab5e3df163be08e6ca9
```

其中，第三行是空，因为此请求不包含请求参数。最后一行是请求体的 hash 值，它必须与 x-amz-content-sha256 请求头的值相同。

b. 待签名字符串

```
AWS4-HMAC-SHA256
20190220T070722Z
20190220/cn/s3/aws4_request
013accc1b2460f530908e106224c57d9fcf9ed74986f5399e27196b73824ddf3
```

2) 生成签名密钥

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<SecretAccessKey>","20190220"),"cn"),"s3"),"aws4_request")
```

3) 计算后的签名

```
5c4e3bc9b2589f2d451a7570cb1283637691f95671525fb0223a1fd158f5fee1
```

4) Authorization 请求头

```
Authorization: AWS4-HMAC-SHA256
Credential=2a948fd3f00ba0925806/20190220/cn/s3/aws4_request,
SignedHeaders=contentlength;host;x-amz-content-sha256;x-amz-date;x-amz-storage-class,
Signature=5c4e3bc9b2589f2d451a7570cb1283637691f95671525fb0223a1fd158f5fee1
```

3. 示例：列出存储桶中的文件

列出存储桶 example-bucket 中的文件，prefix 设置为“t”，最多返回 2 个文件。请求如下：

```
GET /?max-keys=2&prefix=t HTTP/1.1
x-amz-content-sha256: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
Authorization: SignatureToBeCalculated
x-amz-date: 20190220T085955Z
```

```
Host: example-bucket.oos-cn.ctyunapi.cn
```

以下步骤显示 Authorization 请求头的计算的方法。

1) StringToSign

a. 创建规范请求

```
GET
/
max-keys=2&prefix=t
host:example-bucket.oos-cn.ctyunapi.cn
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20190220T085955Z

host;x-amz-content-sha256;x-amz-date
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

其中，最后一行是空请求体的 hash 值。

b. 待签名字符串

```
AWS4-HMAC-SHA256
20190220T085955Z
20190220/cn/s3/aws4_request
3b6553685b6c201cd38cb1077fe657b0f55b355e7ae011e31fa244d009c4d43a
```

2) 生成签名密钥

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<SecretAccessKey>","20190220"),"cn"),"s3"),"aws4_request")
```

3) 计算后的签名

```
72c3758e3b8f27a1a9d9d38b4c143329d3094bc8156d28581bfdd5b7663d6ca8
```

4) Authorization 请求头

```
Authorization: AWS4-HMAC-SHA256
Credential=2a948fd3f00ba0925806/20190220/cn/s3/aws4_request, SignedHeaders=host;x-
amzcontent-sha256;x-amz-date,
Signature=72c3758e3b8f27a1a9d9d38b4c143329d3094bc8156d28581bfdd5b7663d6ca8
```

3.2.4 使用查询参数验证

3.2.4.1 概述

用户可以使用查询字符串参数验证身份信息，此方法也称为预签名 URL。预签名 URL 的用例场景是用户可以授予对 OOS 资源的临时访问权限。例如，用户可以在网站上包含预先签名的 URL，或者在命令行客户端（例如 Curl）中使用它来下载文件。

预签名 URL 支持 GET、DELETE、PUT、HEAD、POST 请求（POST Object 除外）。

注意：使用预签名 URL 方式，有将您授权的数据在过期时间内曝露在互联网上的风险，建议您预先评估后使用。

以下是预签名 URL 的示例。

```
https://oos-cn.ctyunapi.cn/example-bucket/test.txt
?X-Amz-Algorithm=AWS4-HMAC-SHA256 &X-Amz-Credential=<AccessKeyID>/20180721/<oos-region>/s3/aws4_request
&X-Amz-Date=20180721T201207Z
&X-Amz-Expires=86400
&X-Amz-SignedHeaders=host
&X-Amz-Signature=<signature-value>
```

预签名 URL 方式需要注意：

- 预签名 URL 方式必须包含的字段：X-Amz-Date、X-Amz-Algorithm、X-Amz-Signature、X-Amz-SignedHeaders、X-Amz-Credential 和 X-Amz-Expires。字段的顺序可以互换，如果缺少其中一个，返回错误信息。
- 如果访问时间晚于请求中 X- Amz-Expires 时间或者设置的时间格式错误，返回错误信息。
- 如果在 URL 和 Header 中同时包含签名，以 Header 中的签名为准。
- 该 X-Amz-CredentialURL 中的值仅显示可读性“/”字符。在实践中，它应编码为%2F。例如：

```
&X-Amz-Credential=<AccessKeyID>%2F20180721%2F<oos-region>%2Fs3%2Faws4_request
```

下表介绍了 URL 中提供身份验证信息的查询参数。

查询字符串参数	描述

X-Amz-Algorithm	确定 OOS 签名的版本以及用于计算签名的算法。取值为 AWS4-HMAC-SHA256。
X-Amz-Credential	用户的 accessKeyId 和范围信息，范围信息包括请求日期、区域、服务、终止字符串 aws4_request，格式如下： <AccessKeyID>/<date>/<region>/<service>/aws4_request 其中： ● date 格式为 YYYYMMDD。 ● region： ■ 对于 oos api：访问域名为 oos-*xx*.ctyunapi.cn，region 为 *xx*。 ■ 对于统计 api：访问域名为 oos-*xx*-mg.ctyunapi.cn，region 为 *xx*-mg。 ■ 对于操作跟踪 api：访问域名为 oos-*xx*-cloudtrail.ctyunapi.cn，region 为 *xx*。 ■ 对于 iam api：访问域名为 oos-*xx*-iam.ctyunapi.cn，region 为 *xx*。 **说明**：各资源池的详细访问域名详见 **Endpoint 列表**。 ● service： ■ 若使用 OOS API 服务，service 为 s3。 ■ 若使用统计分析服务，service 为 s3。 ■ 若使用操作跟踪服务，service 为 cloudtrail。 ■ 若使用 IAM 服务，service 为 sts。
X-Amz-Date	日期和时间格式必须遵循 ISO 8601 标准，并且必须使用“*yyyyMMdd*T*HHmmss*Z”格式进行格式化。例如，如果日期和时间是“08/01/2018 15：32：41.982-700”，则必须首先将其转换为 UTC（协调世界时），然后提交为“20180801T083241Z”。

X-Amz-Expires	提供生成的预签名 URL 有效的时间段（以秒为单位）。例如，86400（24 小时）。该值是整数。您可以设置的最小值为 1，最大值为 604800（七天）。
X-Amz-SignedHeaders	列出用于计算签名的标头。签名计算中需要以下标头： ● HTTP host 标头。 ● x-amz-*请求头。
X-Amz-Signature	提供签名以验证您的请求。此签名必须与 OOS 计算的签名相匹配。否则，OOS 拒绝该请求。

3.2.4.2 签名过程

下图说明了签名计算过程。

下表描述了图中显示的功能。用户需要为这些功能实现代码。

功能	描述
Lowercase()	将字符串转换为小写。

Hex()	小写十六进制编码。
SHA256Hash()	安全散列算法（SHA）加密散列函数。
HMAC-SHA256()	使用提供的签名密钥的 SHA256 算法计算 HMAC。这是最终的签名。
Trim()	删除任何前导或尾随空格。

UriEncode()	URI 编码每个字节。UriEncode（）必须强制执行以下规则：
	● URI 编码除了下面字符之外的每个字节：'A' - 'Z'，'a' - 'z'，'0' - '9'，'-'，'.'，'_'和'~'。
	● 空格字符是保留字符，必须编码为“%20”（而不是“+”）。
	● 每个 URI 编码字节由'%'和两位十六进制值组成。
	● 十六进制值中的字母必须为大写，例如“%1A”。
	● 除了文件名之外，对正斜杠字符'/'进行编码。例如，如果文件名称为 photos/Jan/sample.jpg，则不对键名称中的正斜杠进行编码。
	重要：
	建议您编写自己的自定义 UriEncode 函数，以确保您的编码可以正常工作。
	以下是 Java 中的示例 UriEncode（）函数。
	`public static String UriEncode(CharSequence input, boolean encodeSlash) {`
	`StringBuilder result = new StringBuilder();`
	`for (int i = 0; i < input.length(); i++) {`
	`char ch = input.charAt(i);`
	`if ((ch >= 'A' && ch <= 'Z') || (ch >= 'a' && ch <= 'z') || (ch >= '0' && ch <= '9') || ch == '_' || ch == '-' || ch == '~' || ch == '.') {`
	`result.append(ch);`
	`} else if (ch == '/') {`
	`result.append(encodeSlash ? "%2F" : ch);`
	`} else {`
	`result.append(toHexUTF8(ch));`

```
            }
        }
        return result.toString();
    }
```

使用参数的签名过程与使用请求头的签名过程类似，如下所示：

- 由于创建预签名 URL 的时候，并不知道有效负载的内容，所以设置常量 UNSIGNED-PAYLOAD。
- 规范查询字符串（Canonical Query String）必须包括除了 X-Amz-Signature 之外的所有上述查询字符串。
- 规范标头必须包括 HTTP Host 标头。如果用户想包含 x-amz-*请求头，这些标头都将参与签名计算。用户可以选择其他的请求头是否参与签名计算。安全起见，尽可能多的请求头参与签名计算。

3.2.4.3 生成签名的示例

通过创建预签名 URL 的方式，与其他人共享 example-bucket 中 test.txt 文件，过期时间设置为 7 天（604800 秒）。以 GET 请求为例：

```
GET
http://oos-cn.ctyunapi.cn/example-bucket/test.txt
?X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=2a948fd3f00ba0925806/20240906/cn/s3/aws4_request&X-Amz-
Date=20240906T235141Z&X-Amz-Expires=604800
&X-Amz-SignedHeaders=host
&X-Amz-Signature=<signature-value>
```

以下步骤首先说明如何计算签名和构建预签名 URL。示例中使用的访问密钥如下：

参数	值
AccessKeyID	2a948fd3f00ba0925806
SecretAccessKey	ef2017c2e5ffa0b1761717ecbca021da16501384

1. StringToSign

a. 创建规范请求（以 GET 请求为例）

```
GET
/example-bucket/test.txt
X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Credential=2a948fd3f00ba0925806%2F20240906%2Fcn%2Fs3%2Faws4_request&X-Amz-
Date=20240906T235141Z&X-Amz-Expires=604800&X-Amz-SignedHeaders=host
host:oos-cn.ctyunapi.cn

host
UNSIGNED-PAYLOAD
```

b. 待签名字符串

```
AWS4-HMAC-SHA256
20240906T235141Z
20240906/cn/s3/aws4_request
9e0b6407d893f03ea8ed79710b98a0b19bf9060b744f0e14212f32d1ac04ba62
```

2. StringKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<SecretAccessKey>","20240906"),"cn"),"s3"),"aws4_request")
```

3. 签名

```
66628b60cb4cc78d37c76b204d6a019572ed3887d84488c72f0643d850ad4915
```

4. 将签名信息添加到请求头

```
http://oos-cn.ctyunapi.cn/example-bucket/test.txt?X-Amz-Date=20240906T235141Z&X-Amz-
Algorithm=AWS4-HMAC-SHA256&X-Amz-
Signature=66628b60cb4cc78d37c76b204d6a019572ed3887d84488c72f0643d850ad4915&X-Amz-
SignedHeaders=host&X-Amz-Credential=2a948fd3f00ba0925806/20240906/cn/s3/aws4_request&X-
Amz-Expires=604800
```

3.3 Bucket 权限控制

OOS 提供 Bucket 级别的权限控制，Bucket 目前有 3 种访问权限：private（私有），public-read（公共读）和 public-read-write（公共读写）：

- private（私有）

  只有根用户和具有相应权限的子用户可以对该存储桶内的文件进行读/写/删除操作（包括 Get、Put 和 Delete Object），其他人（包括匿名访问）只有通过 Bucket Policy 授权或分享链接才可访问该存储桶内的文件。

- public-read（公共读）

  只有根用户和具有相应权限的子用户可以对该存储桶内的文件进行写/删除操作（包括 Put 和 Delete Object）。任何人（包括匿名访问）都可以对该存储桶内的文件进行读操作，这有可能造成您数据的外泄以及费用激增，请慎用该权限。

- public-read-write（公共读写）

  任何人（包括匿名访问）都可以对该存储桶内的文件进行读/写/删除操作（包括 Get、Put 和 Delete Object）。这有可能造成您数据的外泄以及费用激增，若被人恶意写入违法信息还可能会侵害您的合法权益，除特殊场景外，不建议您配置公共读写权限。

  注意：如果想使用访问权限为公共读写的存储桶，请联系天翼云客服评估审核后开通此功能。

说明：新建 Bucket 时，默认权限是 private，用户可以根据需要修改为其他权限。

3.4 访问控制

访问控制（Identity and Access Management，简称 IAM）是 OOS 为用户提供的用户身份与权限管理服务，您可以使用 IAM 创建、管理用户账号，并对这些账号进行权限分配，方便资源管理。

天翼云账号管理员可以：

- 创建、管理子用户账号。
- 控制子用户账号内资源具有的操作权限。
- 按需为子用户分配不同权限，从而避免与其他子用户共享资源使用、访问密钥的使用等，降低账号的信息安全风险。

- 多重身份认证：通过多因素操作认证（MFA）,在进行 IAM 相关操作时，可以使用 MFA，为操作增加一份安全保障。

3.5 Bucket Policy 安全策略

3.5.1 介绍

Bucket policy 用于定义 OOS 资源的访问权限。Bucket Policy 可以用于：

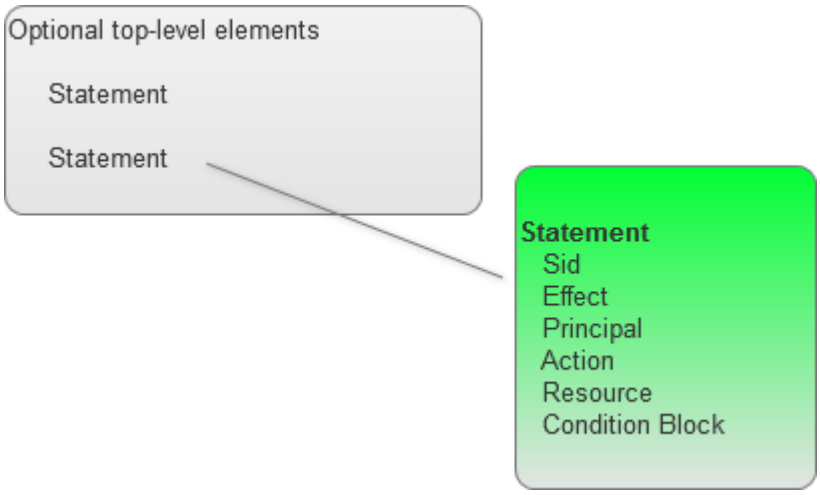
- 允许/拒绝 Bucket 级别的权限。
- 允许/拒绝 Object 级别的权限。

Policy 本身是一个 JSON 字符串，限制在 20KiB。一个 Bucket Policy 包括：

- 可选的 Bucket Policy 基本信息。
- 一个或多个独立的 statements。

每个 statement 包括一个权限的核心信息。如果一个 Bucket Policy 包括多个 statements，那么 statements 之间是逻辑或关系。

注意：如果多个 statement 之间有重叠或者冲突，那么含 deny 语句优先级最高。



3.5.2 Bucket Policy 元素

下面介绍 statements 中的各个元素。

3.5.2.1 Version

Version 是 Bucket Policy 语言的版本，它是个可选项，目前 OOS API 只允许填写 2012-10-17。

```
"Version":"2012-10-17"
```

3.5.2.2 Id

Id 是 Bucket Policy 的一个可选标识。推荐使用 UUID 来指明 Bucket Policy。

3.5.2.3 Statement

Statement 是个重要的元素，它可以包含多个元素。Statement 元素可以包含一组独立的 statements，每个独立的 Statement 是一个包含在大括号（{}）内的严格的 JSON 块。

```
"Statement":[{...},{...},{...}]
```

1. Sid

Sid(statement ID)是 statement 的一个可选标识，它可以看作是 policy id 的子 id。

```
"Sid" : "1"
```

2. Effect

Effect 是一个必填元素，用于表示允许访问或拒绝访问，有效值只能是 Allow 或 Deny。

```
"Effect":"Allow"
```

3. Principal

Principal 用于指定被允许或被拒绝访问的用户，Principal 可以为：

- "Principal": "*"、或"Principal": {"CTYUN":"*"}：表示所有用户。
- "Principal": { "CTYUN": "arn:ctyun:iam::*accountId*:root" }：表示根用户。
- "Principal": { "CTYUN": "arn:ctyun:iam::*accountId*:user/*user-name*" }：表示子用户。

可以指定根用户和多个子用户,如下表示方法：

```
"Principal": {
  "CTYUN": [
    "arn:ctyun:iam::accountId:root",
    "arn:ctyun:iam::accountId:user/user-name1",
    "arn:ctyun:iam::accountId:user/user-name2"
  ]
}
```

4. Action

Action 用于指定被允许或拒绝访问的操作，必填项。只有拥有相关权限的用户能执行 Bucket 相关写操作，以及 Bucket 子资源的相关操作。Action 中可以配置以下权限。

与 Bucket 相关的 OOS 权限列表：

权限关键字	OOS 操作
oos:ListBucket	GET Bucket（List Objects）、HEAD Bucket
oos:ListBucketMultipartUploads	List Multipart Uploads
oos:DeleteMultipleObjects	DeleteMultipleObjects

文件操作的 OOS 权限列表：

权限	OOS 操作
oos:AbortMultipartUpload	Abort Multiple Upload
oos:DeleteObject	DELETE Object
oos:GetObject	GET Object、HEAD Object
oos:ListMultipartUploadParts	List Part
oos:PutObject	PUT Object、PUT Object-Copy、POST Object、Initiate Mulitipart Upload、Upload Part、Compelete Multipart Upload、Upload Part －Copy

可使用通配符 (*) 来访问 OOS 产品提供的所有操作。例如，以下 Action 元素适用于所有 OOS 操作。

```
"Action": "oos:*"
```

还可以使用通配符 (*) 作为操作名称的一部分。例如下面的 Action 元素，适用于 PutObject, GetObject, DeleteObject

```
"Action": "oos:*Object"
```

5. Resource

Resource 是指 Policy 覆盖的文件或文件集合，可以使用通配符*和？。

当 Action 中有桶级别权限时（如 oos:ListBucket），Resource 需配置到具体 Bucket 名称，如 "Resource":"arn:ctyun:oos:::example-bucket"。

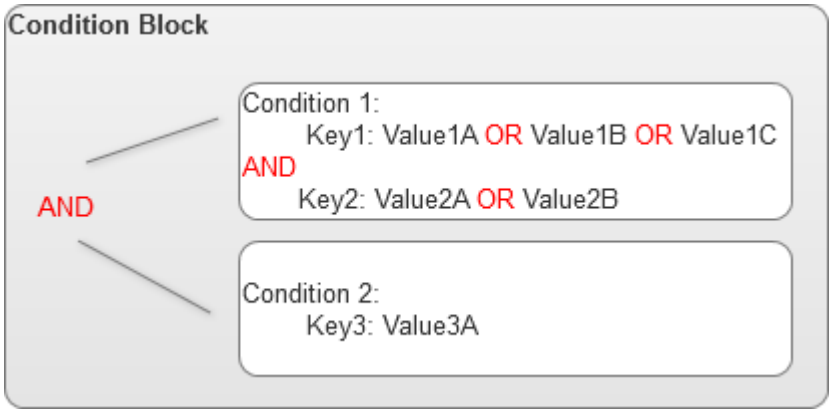
当 Action 中有文件级别权限时（如 oos:GetObject），Resource 需配置到具体 Bucket 下的文件，如"Resource":"arn:ctyun:oos:::example-bucket/*objectname*"、"Resource":"arn:ctyun:oos:::example-bucket/*"。

例如要使 Policy 对于所有以 image 开头的 Object 生效，可以这样配置：

```
"Resource":"arn:ctyun:oos:::example-bucket/image*"
```

6. Condition

Condition 是 Statement 中最复杂的一个元素，我们把它称之为 condition 块，虽然只有一个 condition 元素，但是它可以包含多个 conditions，而且每个 condition 可以包含多个 key-value 对。如下图所示，condition 块中的各个 condition 之间是逻辑与关系，condition key 之间也是逻辑与关系，各个 value 之间是逻辑或关系。



（1） 条件运算符

条件运算符是条件的“动词”形式，可指定 OOS 执行的比较类型。条件运算符可分为以下类别：

1) String Conditions

String conditions 可以允许设置 string 的匹配规则。

Condition	**描述**
StringEquals	精准匹配指定的值，区分大小写。
StringNotEquals	与指定的值不匹配，区分大小写。
StringEqualsIgnoreCase	与指定的值精准匹配，不区分大小写。
StringNotEqualsIgnoreCase	与指定的值不匹配，不区分大小写

StringLike	与指定的值精准匹配。或通过填充通配符，与指定的值相似，可以包括多字符匹配的通配符 (*)或单字符匹配的通配符 (?)。区分大小写。
StringNotLike	与指定的值不匹配，区分大小写的无效匹配。或通过填充通配符，与指定的值也不匹配。

例如，要匹配 HTTP Referer 头是以“http://www.ctyun.cn/”开头的请求，可以这样配置：

```
"StringLike":{
         "ctyun:Referer":[
           "http://www.ctyun.cn/*"
         ]
       }
```

2） 布尔值条件运算符

利用布尔值条件，用户可以通过与“正确”或“错误”的对比，来设置 Condition 元素。

条件运算符	说明
Bool	布尔值匹配

例如，下列声明使用 Bool 条件运算符与 ctyun:SecureTransport 键来指定必须使用 https 发出请求。

```
{
  "Version": "2012-10-17",
  "Statement": [
  {
    "Effect": "Allow",
    "Principal":{ "CTYUN": ["*"] },
   "Action": "oos:GetObject",
   "Resource": "arn:ctyun:oos:::example_bucket/*",
   "Condition": {"Bool": {"ctyun:SecureTransport": "true"}}
  }
 ]
}
```

3） IP Address

IP address conditions 允许设置 IP 地址的匹配规则，与 ctyun:SourceIp key 配合使用。此项的值必须符合标准的 CIDR 格式（例如：10.52.176.0/24），参考 RFC 4632

Condition	描述
IpAddress	IP 地址或范围的白名单
NotIpAddress	IP 地址或范围的黑名单

（2） 条件键

条件键	描述
ctyun:Referer	与字符串运算符结合使用。用于检查请求中的 Referer 请求头
ctyun:SecureTransport	与布尔值运算符结合使用。检查请求是否是使用 SSL 发送的。
ctyun:SourceIp	与 IP 地址运算符结合使用。用于检查请求者的 IP 地址。
ctyun:UserAgent	与字符串运算符结合使用。用于检查请求者的客户端应用程序。

3.5.3 示例

1. 下面是一个定义 Referer Policy 的例子。

   配置 Bucket（名为 example-bucket）的访问策略：允许特定子用户（test1，其根用户 ID 为 32fefj64y54gc）访问此 Bucket 资源，并且要求请求包含特定的 Referer 头（http://www.ctyun.cn/ 或 http://ctyun.cn/）。

```
{
  "Version":"2012-10-17",
  "Id":"*",
  "Statement":[
    {
      "Sid":"*",
      "Effect":"Allow",
      "Principal":{ "CTYUN": "arn:ctyun:iam::32fefj64y54gc:user/test1" },
      "Action":"oos:*",
      "Resource":"arn:ctyun:oos:::example-bucket/*",
      "Condition":{
        "StringLike":{
          "ctyun:Referer":[
            "http://www.ctyun.cn/*",
```

```
          "http://ctyun.cn/*"
        ]
      }
    }
  }
 ]
}
```

2. 下面是一个定义 IP Policy 的例子。

   配置 Bucket（名为 example-bucket）的访问策略：允许特定子用户（test2，其根用户 ID 为 32fefj64y54gc）在特定网段（192.168.143.0/24，但排除 192.168.143.188）内进行访问。

```
{
        "Version": "2012-10-17",
        "Id": "PolicyId1",
        "Statement": [
          {
              "Sid": "IPAllow",
              "Effect": "Allow",
              "Principal": {
                "CTYUN": "arn:ctyun:iam::32fefj64y54gc:user/test2"
           },
           "Action": "oos:GetObject",
           "Resource": "arn:ctyun:oos:::example-bucket/*",
           "Condition" : {
               "IpAddress" : {
                   "ctyun:SourceIp": "192.168.143.0/24"
               },
               "NotIpAddress" : {
                   "ctyun:SourceIp": "192.168.143.188/32"
               }
           }
       }
    ]
}
```

3. 下面的例子可向匿名用户授予公共读权限

下面的示例策略向任何公用匿名用户授予 oos:GetObject 权限。此权限允许任何人读取文件数据，当用户将 Bucket 配置为网站并且希望每个人都能读取存储桶中的文件时，此配置十分有用。可以将 Bucket 设置为私有，然后配置以下 Bucket 策略。

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"AddPerm",
      "Effect":"Allow",
      "Principal":{ "CTYUN": ["*"] },
      "Action":["oos:GetObject"],
      "Resource":["arn:ctyun:oos:::example-bucket/*"]
    }
  ]
}
```

3.6 合规保留

OOS 提供合规保留功能，即开启 Bucket 合规保留功能后，任何用户（包括根用户）都不能对此 Bucket 内处于合规保留期的文件进行修改和删除。

可以根据需求，对 Bucket 级别开启合规保留功能，以天（Days）或者以年（years）为单位设置合规保留时长，1year=365 days。

注意：

- 合规保留一旦开启，不能关闭，不能缩短合规保留时长，但可以延长合规保留时长。
- 合规保留的时间精确到秒，例如对 Bucket A 设置合规保留时长为 10 天，文件 A1 属于 Bucket A，A1 的最后更新时间为 2019-3-1 12:00:00，该文件会在 2019-3-11 12:00:01 过合规保留期。
- 任何用户（包括根用户）都不能修改、覆盖、删除处于合规保留期的文件。
- 处于合规保留期的文件，无法通过调用 API、控制台修改文件的存储类型，只能通过设置的生命周期规则修改存储类型。

- 文件处于合规保留期：如果生命周期规则为修改文件存储类型，则生命周期规则可以生效。如果生命周期规则为到期删除文件，则文件必须过了合规保留期后，生命周期规则才能生效。

4 HTTP REST 接口（OOS API）

说明：本章节举例中的域名均以对象存储网络的 OOS API 域名 oos-cn.ctyunapi.cn 为例。如果是对象存储网络 2，请使用域名 oos-cn2.ctyunapi.cn。如果是香港精品网，请使用域名 oos-cnhk-hqnet.ctyunapi.cn。如果是香港普通网，请使用域名 oos-cnhk-nqnet.ctyunapi.cn。

4.1 OOS API 请求结构

- **接入地址**

  对象存储网络接入地址为：oos-cn.ctyunapi.cn。对象存储网络 2 接入地址为：oos-cn2.ctyunapi.cn。香港精品网接入地址为：oos-cnhk-hqnet.ctyunapi.cn。香港普通网接入地址为：oos-cnhk-nqnet.ctyunapi.cn。

  注意：对于对象存储网络、对象存储网络 2 中的 OOS API，如果您的数据存储在某个资源池，建议您直接使用该资源池的 Endpoint，详见 **Endpoint 列表**。

- **通信协议**

  为了保证通信的安全性，同时支持 HTTP 和 HTTPS。

- **请求方法**

  OOS API 支持 GET、POST、PUT、HEAD、DELETE 和 OPTION 请求方法发送请求。

- **请求参数**

  每个请求都需要指定如下信息：

  - 每个操作接口都需要包含的公共请求参数。
  - 操作接口所特有的请求参数。

  本节主要描述公共请求参数和请求结果。

  说明：在后续提到具体 API 时，举例中都会有**公共请求头**、**公共响应头**，但是不对其进行描述和解释。

4.1.1 公共请求头

以下是 OOS API 的公共请求头。

名称	描述	是否必须
BucketName	Bucket 名字。 除 GET Service（List Buckets）、GET Regions 外，其他接口都需要填写该参数。 Bucket 的命名规范如下： ● Bucket 名称必须全局唯一。 ● Bucket 名称长度介于 3 到 63 字节之间。 ● Bucket 名称只能由小写字母、数字、短横线（-）和点（.）组成。 ● Bucket 名称可以由一个或者多个小节组成，小节之间用点（.）隔开，各个小节需要： ■ 只能包含小写字母、数字和短横线（-）。 ■ 必须以小写字母或者数字开始。 ■ 必须以小写字母或者数字结束。 ● Bucket 名称不能是一组或多组“数字.数字”的组合（如 192.162.0.1）。 ● Bucket 名称中不能包含双点（..）、横线点（-.）和点横线（.-）。 ● 不允许使用非法敏感字符，例如暴恐涉政相关信息等。	是
Authorization	用于身份验证的请求头。	是
Content-Length	请求体的长度。	否
Content-MD5	按照 RFC 1864，使用 base64 编码格式生成信息的 128 位 MD5 值。此请求头可以用作数据完整性检查，以验证数据是否与客户端发送的数据相同。	否
x-amz-content-sha256	当使用 V4 签名对请求进行身份验证时，此请求头提供请求有效负载的哈希。	V4 签名必填。
Content-Type	描述请求内容的标准 MIME 类型。	否

Date	请求的日期和时间，GMT 时间。	V2 签名可以填写 x-Amz-Date 或 Date。
x-amz-date	请求的日期和时间。日期和时间格式必须遵循 ISO 8601 标准，并且必须使用“yyyyMMddT HHmmssZ”格式进行格式化。例如，如果日期和时间是“08/01/2018 15：32：41.982-700”，则必须首先将其转换为 UTC（协调世界时），然后提交为“20180801T083241Z”。	V4 签名必填。
Host	请求的域名。 ● 对象存储网络的请求域名为：oos-cn.ctyunapi.cn。 ● 对象存储网络 2 的请求域名为：oos-cn2.ctyunapi.cn。 ● 香港精品网的请求域名为：oos-cnhk-hqnet.ctyunapi.cn。 ● 香港普通网的请求域名为：oos-cnhk-nqnet.ctyunapi.cn。	是
Connection	客户端与 OOS 服务器之间的连接状态。 取值： ● keep-alive：长连接，请求结束后继续保持连接。 ● close：短连接，请求结束后关闭连接。 默认值为：keep-alive。	否

4.1.2 公共响应头

每个 OOS API 响应结果中都包含公共响应头。

名称	描述

Content-Length	响应体的长度。
Content-Type	响应内容的 MIME 类型。
Date	OOS 返回响应的日期和时间。
ETag	文件的哈希值。ETag 只反映了文件的内容，而不是其元数据。此响应头可以用作数据完整性检查，以验证数据是否与客户端发送的数据相同。
Server	服务端名称，默认值是 CTYUN。
x-amz-request-id	用于唯一标示请求的 ID。
Connection	客户端与 OOS 服务器之间的连接状态。 ● 如果请求时 Connection 值为 keep-alive，请求结束后继续保持连接，不返回此响应头。 ● 如果请求时 Connection 值为 close，请求结束后关闭连接，返回此响应头 Connection: close。

4.2 关于 Service 的操作

说明：以下举例中的域名均以对象存储网络的 OOS API 域名 oos-cn.ctyunapi.cn 为例。如果是对象存储网络 2，请使用域名 oos-cn2.ctyunapi.cn。如果是香港精品网，请使用域名 oos-cnhk-hqnet.ctyunapi.cn。如果是香港普通网，请使用域名 oos-cnhk-nqnet.ctyunapi.cn。

4.2.1 GET Service（List Buckets）

此操作用来返回请求者拥有的读权限的所有 Bucket，其中“/”表示根目录。该 API 只对验证用户有效，匿名用户不能执行该操作

- **请求语法**

```
GET / HTTP/1.1
Host: oos-cn.ctyunapi.cn
Date: Date
Authorization: SignatureValue
```

- **请求参数**

该操作不使用请求参数。

- **响应结果**

名称	描述
Owner	Bucket 所属账户的信息。
ID	Bucket 所属账户的 ID。
Buckets	存储一个或多个 Bucket 的容器。
Bucket	存储 Bucket 信息的容器。
Name	Bucket 的名称。
CreationDate	Bucket 的创建日期。

- **请求示例**

```
GET / HTTP/1.1
Host: oos-cn.ctyunapi.cn
Date: Thu, 28 Oct 2021 06:06:32 GMT
Content-Type: application/octet-stream
```

```
Connection: Keep-Alive
Authorization: SignatureValue
```

- 响应示例

```
HTTP/1.1 200 OK
Content-Type: application/xml;charset=UTF-8
Date: Thu, 28 Oct 2021 06:06:32 GMT
x-amz-request-id: 8ec6af80431f4f0281edf1f800f8fa00bfc1c7b5b7b9bbbdbf
Server: CTYUN
Content-Length: 1999

<?xml version="1.0" encoding="UTF-8"?>
<ListAllMyBucketsResult xmlns="http://doc.s3.amazonaws.com/2006-03-01">
    <Owner>
        <ID>xxxxxxxx@ctyun.cn</ID>
        <DisplayName></DisplayName>
    </Owner>
    <Buckets>
        <Bucket>
            <Name>test690-2</Name>
            <CreationDate>2021-07-10T17:52:34.949Z</CreationDate>
        </Bucket>
        <Bucket>
            <Name>yxxxytest-14363</Name>
            <CreationDate>2021-10-28T06:06:19.309Z</CreationDate>
        </Bucket>
    </Buckets>
</ListAllMyBucketsResult>
```

4.2.2 GET Regions

获取资源池中的索引位置和数据位置列表。

注意：香港节点不支持此接口。

- **请求语法**

```
GET /?regions HTTP/1.1
HOST:oos-cn.ctyunapi.cn
Date:date
Authorization: SignatureValue
```

- **请求参数**

名称	描述	是否必须
regions	表示要获取索引位置和数据位置的列表。	是

- **响应结果**

名称	描述
BucketRegions	Bucket 的索引位置和数据位置。 类型：容器
MetadataRegions	Bucket 的索引位置。 类型：容器
DataRegions	Bucket 的数据位置。 类型：容器
Region	索引位置或数据位置的值。 类型：字符串

- **请求示例**

```
GET /?regions HTTP/1.1
Host: oos-cn.ctyunapi.cn
Date: Fri, 29 Oct 2021 06:49:13 GMT
Content-Type: application/octet-stream
Connection: Keep-Alive
Authorization: SignatureValue
```

- **响应示例**

```
HTTP/1.1 200 OK
Content-Type: application/xml;charset=UTF-8
Date: Fri, 29 Oct 2021 06:49:12 GMT
x-amz-request-id: 79a75e2d5bc54d4ffc686c737b73757b3a3c4030323436383a
Server: CTYUN
Content-Length: 281

<?xml version="1.0" encoding="UTF-8"?>
<BucketRegions>
    <MetadataRegions>
        <Region>ShenYang</Region>
        <Region>ChengDu</Region>
        <Region>ZhengZhou</Region>
    </MetadataRegions>
    <DataRegions>
        <Region>ShenYang</Region>
        <Region>ChengDu</Region>
        <Region>ZhengZhou</Region>
    </DataRegions>
</BucketRegions>
```

4.3 关于 Bucket 的操作

说明：以下举例中的域名均以对象存储网络的 OOS API 域名 oos-cn.ctyunapi.cn 为例。如果是对象存储网络 2，请使用域名 oos-cn2.ctyunapi.cn。如果是香港精品网，请使用域名 oos-cnhk-hqnet.ctyunapi.cn。如果是香港普通网，请使用域名 oos-cnhk-nqnet.ctyunapi.cn。

4.3.1 PUT Bucket

PUT Bucket 操作可以用来：

- 创建一个新的 Bucket。
- 对已有 Bucket 的 Bucket ACL 进行修改。
- 对数据位置进行修改，但是不能修改索引位置（香港节点不支持）。

只有根用户和具有 PUT Bucket 权限的子用户才能创建 Bucket。

Bucket 的命名规范具体参见 **Bucket 的命名规范**。

注意：香港节点只有请求头，没有请求体。

- **请求语法**

```
PUT / HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Content-Type: application/xml; charset=utf-8
Content-Length: Length
Date: date
Authorization: SignatureValue

<CreateBucketConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <MetadataLocationConstraint>
        <Location>MetadataRegion</Location>
    </MetadataLocationConstraint >
    <DataLocationConstraint>
        <Type>RegionType</Type>
        <LocationList>
            <Location>DataRegion</Location>
            <Location>DataRegion</Location>
        </LocationList>
        <ScheduleStrategy>Strategy</ScheduleStrategy>
    </DataLocationConstraint >
</CreateBucketConfiguration>
```

- **请求参数**

名称	描述	是否必须
x-amz-acl	设置 Bucket 的 ACL（Access Control List） 类型：字符串 有效值： ● private：私有。只有根用户和具有相应权限的子用户可以对该 Bucket 内的文件进行读写操作（包括 Put、Delete 和 Get Object）。其他人（包括匿名访问）只有通过 Bucket Policy 授权或分享链接才可访问该 Bucket 内的文件。 ● public-read：公共读。只有根用户和具有相应权限的子用户可以对该 Bucket 内的 Object 进行写操作（包括 Put 和 Delete Object）。任何人（包括匿名访问）都可以对该 Bucket 内的 Object 进行读操作，这有可能造成您数据的外泄以及费用激增，请慎用该权限。 ● public-read-write：公共读写。任何人（包括匿名访问）都可以对该 Bucket 内的 Object 进行读/写/删除操作。这有可能造成您数据的外泄以及费用激增，若被人恶意写入违法信息还可能会侵害您的合法权益，除特殊场景外，不建议您配置公共读写权限。 **注意**：如果想使用访问权限为公共读写的存储桶，请联系天翼云客服评估审核后开通此功能。 默认值为 private。	否

- **请求元素（香港节点不支持）**

名称	描述	是否必须
CreateBucketConfiguration	设置 Bucket 索引位置和数据位置的容器。 类型：容器	否

MetadataLocationConstraint	设置 Bucket 的索引位置。 类型：容器 父节点：CreateBucketConfiguration	创建 Bucket 的时候可非必填。修改 Bucket ACL 或数据位置的时候不能填写。
DataLocationConstraint	设置 Bucket 的数据位置。 类型：容器 父节点：CreateBucketConfiguration	否
Type	数据位置的类型 类型：枚举 有效值：Local(本地) |Specified（指定位置） 默认：Local 父节点：DataLocationConstraint	否
LocationList	指定的数据位置 类型：容器 父节点：DataLocationConstraint	否
Location	索引位置或数据位置。 类型：字符串 有效值： ● 父节点为 MetadataLocationConstraint，表示索引位置。对于对象存储网络，取值为：ChengDu、FuZhou、GuiYang、HangZhou、LaSa、LanZhou、QingDao、ShenYang、ShenZhen、WuHan、WuHu、WuLuMuQi、ZhengZhou、SH2、SuZhou；对于对象存储网络 2，取值为：NeiMeng1、HangZhou1。	否

	● 父节点为 LocationList，表示数据位置。对于对象存储网络，取值为：ChengDu、GuiYang、LaSa、LanZhou、QingDao、SH2、ShenYang、ShenZhen、SuZhou、WuHan、WuHu、WuLuMuQi、ZhengZhou；对于对象存储网络 2，取值为：NeiMeng1、HangZhou1。 默认值：无 父节点：MetadataLocationConstraint 或 Locationlist	
ScheduleStrategy	指定数据时的调度策略 类型：枚举 有效值：Allowed(允许 OOS 自动调度)|NotAllowed（不允许 OOS 自动调度） 默认：Allowed 父节点：DataLocationConstraint	否

- **请求示例 1**

请求创建一个名叫 docs 的 Bucket，索引位置设置为 LaSa，数据位置设置为优先本地，Bucket 权限设置为私有。

```
PUT / HTTP/1.1
Host: docs.oos-cn.ctyunapi.cn
Content-Length:200
Content-Type: application/xml;charset=utf-8
Date: Mon, 03 Sep 2012 12:00:00 GMT
x-amz-acl: private
Authorization: SignatureValue

<CreateBucketConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <MetadataLocationConstraint>
        <Location>LaSa</Location>
    </MetadataLocationConstraint>
```

```
    <DataLocationConstraint>
        <Type>Local</Type>
    </DataLocationConstraint>
</CreateBucketConfiguration >
```

- **响应示例 1**

```
HTTP/1.1 200 OK
x-amz-request-id: c962c64b9f294f7b14877a897e8b8d9352545d484a4c4e5052
Date: Mon, 03 Sep 2012 12:00:00 GMT
Location: /docs
Content-Length: 0
Server: CTYUN
```

- **请求示例 2**

请求创建一个名叫 docs 的 Bucket，索引位置设置为 LaSa，数据位置设置为拉萨、成都，调度策略是允许 OOS 自动调度。

```
PUT / HTTP/1.1
Host: docs.oos-cn.ctyunapi.cn
Content-Length:200
Content-Type: application/xml; charset=utf-8
Date: Mon, 03 Sep 2012 12:00:00 GMT
Authorization: SignatureValue

<CreateBucketConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <MetadataLocationConstraint>
        <Location>LaSa</Location>
    </MetadataLocationConstraint>
    <DataLocationConstraint>
         <Type>Specified</Type>
          <LocationList>
            <Location>LaSa</Location>
            <Location>ChengDu</Location>
          </LocationList>
          <ScheduleStrategy>Allowed</ScheduleStrategy>
    </DataLocationConstraint>
</CreateBucketConfiguration>
```

- **响应示例 2**

```
HTTP/1.1 200 OK
x-amz-request-id: c962c64b9f294f7b14877a897e7b8d9352545d484a4c4e6879
Date: Mon, 03 Sep 2012 12:00:00 GMT
Location: /docs
Content-Length: 0
Server: CTYUN
```

- **请求示例 3**

修改数据位置为**就近**。

```
PUT / HTTP/1.1
Host: docs.oos-cn.ctyunapi.cn
Date: Sun, 29 Sep 2019 07:16:11 GMT
Content-Type: application/xml;charset=utf-8
Content-Length: 170
Authorization: SignatureValue

<CreateBucketConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <DataLocationConstraint>
    <Type>Local</Type>
  </DataLocationConstraint>
</CreateBucketConfiguration>
```

- **响应示例 3**

```
HTTP/1.1 200 OK
Location: /docs
Date: Sun, 29 Sep 2019 07:16:32 GMT
x-amz-request-id: c08665066c4e4b4186f9ecfbf0fdff05c4c6cdbabcbec0c2c4
Content-Length: 0
Server: CTYUN
```

4.3.2 GET Bucket Location

此操作用来获取 Bucket 的索引位置和数据位置，只有根用户和拥有 GET Bucket Location 权限的子用户才能执行此操作。

注意：香港节点不支持此接口。

- **请求语法**

```
GET /?location HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Authorization: SignatureValue
```

- **响应结果**

名称	描述
BucketConfiguration	设置 Bucket 索引位置和数据位置的容器。 类型：容器
MetadataLocationConstraint	设置 Bucket 的索引位置。 类型：容器 父节点：CreateBucketConfiguration
DataLocationConstraint	设置 Bucket 的数据位置。 类型：容器 父节点：CreateBucketConfiguration
Type	数据位置的类型 类型：枚举 有效值：Local(本地) |Specified（指定位置），默认 Local。 父节点：DataLocationConstraint
LocationList	指定的数据位置。 类型：容器 父节点：DataLocationConstraint
Location	索引位置或数据位置。

	类型：字符串 有效值： ● 父节点为 MetadataLocationConstraint，表示索引位置，对于对象存储网络，取值为：ChengDu、FuZhou、GuiYang、HangZhou、LaSa、LanZhou、QingDao、ShenYang、ShenZhen、WuHan、WuHu、WuLuMuQi、ZhengZhou、SH2、SuZhou；对于对象存储网络 2，取值为：NeiMeng1、HangZhou1。 ● 父节点为 LocationList，表示数据位置。对于对象存储网络，取值为：ChengDu、GuiYang、LaSa、LanZhou、QingDao、SH2、ShenYang、ShenZhen、SuZhou、WuHan、WuHu、WuLuMuQi、ZhengZhou；对于对象存储网络 2，取值为：NeiMeng1、HangZhou1。 父节点：MetadataLocationConstraint 或 LocationList
ScheduleStrategy	指定数据时的调度策略 类型：枚举 取值：Allowed(允许 OOS 自动调度)|NotAllowed（不允许 OOS 自动调度）。 父节点：DataLocationConstraint

- **请求示例**

获取一个名叫 docs 的 Bucket 的数据位置和索引位置：

```
GET /?location HTTP/1.1
Host: docs.oos-cn.ctyunapi.cn
Content-Type: application/xml;charset=utf-8
Date: Mon, 03 Sep 2012 12:00:00 GMT
Authorization: SignatureValue
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id: b9e8da1354774e89d14437463b484a500f111b0507090b0d0f
Date: Mon, 03 Sep 2012 12:00:00 GMT
Content-Type: application/xml;charset=utf-8
Content-Length: 200
Server: CTYUN

<BucketConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <MetadataLocationConstraint>
        <Location>LaSa</Location>
    </MetadataLocationConstraint>
    <DataLocationConstraint>
        <Type>Local</Type>
    </DataLocationConstraint>
</BucketConfiguration>
```

4.3.3 GET Bucket ACL

此操作用来获取 Bucket ACL 信息，只有拥有 GET Bucket ACL 权限的用户才可以执行此操作。

- **请求语法**

```
GET /?acl HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Authorization: SignatureValue
```

- **响应结果**

名称	描述
AccessControlPolicy	响应的容器。 类型：容器 子节点：Owner、AccessControlList
Owner	存储 Bucket 所属账户信息的容器。 类型：容器 父节点：AccessControlPolicy 子节点：ID、DisplayName
ID	Bucket 所属账户的 ID 信息。 类型：字符串 父节点：Owner
AccessControlList	存储 ACL 信息的容器。 类型：容器 父节点：AccessControlPolicy 子节点：Grant
Grant	存储 Permission 和 Grantee 的容器。 类型：容器 父节点：AccessControlList 子节点：Grantee、URI

Grantee	用来存储 Display 和拥有 ID 的用户被承认的许可的容器。
Permission	对一个 Bucket 认可的许可信息。 类型：字符串 取值： ● READ（公共读） ● FULL_CONTROL（公共读写） ● 空（私有） 父节点：Grant

- **请求示例**

请求返回 docs 的 ACL 信息

```
GET /?acl HTTP/1.1
Host: docs.oos-cn.ctyunapi.cn
Date: Mon, 01 Nov 2021 06:31:14 GMT
Content-Type: application/octet-stream
Connection: Keep-Alive
Authorization: SignatureValue
```

- **响应示例**

```
HTTP/1.1 200 OK
Last-Modified: Mon, 13 Sep 2021 07:51:46 GMT
Content-Type: application/xml;charset=UTF-8
Date: Mon, 01 Nov 2021 06:31:14 GMT
x-amz-request-id: 23ee087970af44915ecaced5ddd5d7dd9c9ea4929496989a9c
Content-Length: 392
Server: CTYUN

<?xml version="1.0" encoding="UTF-8"?>
<AccessControlPolicy>
    <Owner>
        <ID> ctyuntest@ctyun.cn</ID>
        <DisplayName>.........</DisplayName>
    </Owner>
    <AccessControlList>
        <Grant>
            <Grantee xsi:type="Group"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

```
                <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
            </Grantee>
            <Permission></Permission>
        </Grant>
    </AccessControlList>
</AccessControlPolicy>
```

4.3.4 GET Bucket (List Objects)

此操作用来返回 Bucket 中部分或者全部（最多 1000）的 Object 信息。用户可以在请求元素中设置选择条件来获取 Bucket 中的 Object 的子集。

- **请求语法**

```
GET / HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Authorization: signatureValue
```

- **请求参数**

名称	描述	是否必须
delimiter	分隔符，一个用来对关键字们进行分组的字符。所有的关键字都包含 delimiter 和 prefix 间的相同子串，prefix 之后第一个遇到 delimiter 的字符串都会加到一个叫 CommonPrefix 的组中。如果没有特别定义 prefix，所有的关键字都会被返回，但不会有 CommonPrefix。delimiter 只支持”/”，不支持其他分隔符。 类型：字符串	否
marker	分页标识。还有需要返回的用户时，上条响应结果中会返回该参数。查看未显示项时，请求参数中需要携带此参数。 类型：字符串 取值：与上条响应中返回的结果值相同。	否
max-keys	设置响应中最多返回的条数。如果存在超出您指定的返回项，则响应元 IsTruncated 为 true，表示需要再次发送请求查看未显示的项。查看未显示的项时，需要携带响应参数 Marker 的值。 类型：整型 取值：1~1000，默认值为 1000。	否

prefix	前缀用来限制返回的结果必须以这个前缀开始，可以通过前缀将 Bucket 分为若干个组。	否
encoding-type	指定响应中 Delimiter、Marker、Prefix、NextMarker、Key、CommonPrefixes.Prefix 的编码类型。如果 Delimiter、Marker、Prefix、NextMarker、Key、CommonPrefixes.Prefix 包含 xml 1.0 标准不支持的控制字符，可通过设置该参数对响应中的 Delimiter、Marker、Prefix、NextMarker、Key、CommonPrefixes.Prefix 进行编码。 类型：字符串 取值：url，字母不区分大小写。	否

- **响应结果**

名称	描述
ListBucketResult	包含响应的容器。 类型：容器 子节点：Name、Prefix、Marker、MaxKeys、EncodingType、Delimiter、IsTruncated、NextMarker、Contents、CommonPrefixes
Name	Bucket 的名称。 类型：字符串 父节点：ListBucketResult
Prefix	关键字以特定的前缀开始。 类型：字符串 父节点：ListBucketResult
Marker	标记从哪个位置开始罗列出 Bucket 中的 Object。 类型：字符串 父节点：ListBucketResult
MaxKeys	响应中最多返回的条数。 类型：字符串

	父节点：ListBucketResult
EncodingType	Delimiter、Marker、Prefix、NextMarker、Key 的编码类型。 类型：字符串 父节点：ListBucketResult
Delimiter	分隔符。 类型：字符串 父节点：ListBucketResult
IsTruncated	通过是（true）或否（false）来表示返回的结果是否为所有要求的结果。 类型：Boolean 父节点：ListBucketResult
NextMarker	若 IsTruncated 返回 true，用户可以将 NextMarker 的值设置为下次请求中的 marker 参数，以便获取后续文件。 类型：字符串 父节点：ListBucketResult
Contents	每个 Object 返回的元数据。 类型：容器 父节点：ListBucketResult 子节点：Key，LastModified，Etag，Size，StorageClass
Key	文件的关键字。 类型：字符串 父节点：Contents
LastModified	记录的文件最后一次被修改的日期和时间。 类型：字符串 父节点：Contents
ETag	通过 MD5 的方式计算出标签，ETag 主要用来反映文件内容的信息发生了改变，并不反映元数据的变化。 类型：字符串

	父节点：Contents
Size	记录文件 object 的大小。 类型：字符串 父节点：Contents
StorageClass	存储类型： ● STANDARD：标准存储。 ● STANDARD_IA：低频访问存储。 类型：字符串 父节点：Contents
CommonPrefixes	当定义 delimiter 之后，返回结果中会包含 CommonPrefixes，CommonPrefix 中包含以 prefix 开头，delimiter 结束的字符串组合，比如 prefix 是 note/,同时 delimiter 是斜杠（/），结果中的 note/summer/july/lotus.jpg 将返回 note/summer/，其余结果将按照 Maxkeys 要求返回。 类型：容器 父节点：ListBucketResult 子节点：Prefix
CommonPrefixes.Prefix	如果请求中不包含 Prefix 参数，那么这个元素只显示那些在 delimiter 字符第一次出现之前的 key 的子字符串，且这些 key 不在响应的其它位置出现。 如果请求中包含 Prefix 参数，那么这个元素显示在 prefix 之后，到第一次出现 delimiter 之间的子串。 类型：字符串 父节点：CommonPrefixes

- **请求示例**

```
GET / HTTP/1.1
Host: docs.oos-cn.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
X-Amz-Date: 20200304T070632Z
Content-Type: application/xml; charset=utf-8
```

```
Authorization: SignatureValue
```

- **响应示例**

```
HTTP/1.1 200 OK
Date: Wed, 04 Mar 2020 07:06:40 GMT
x-amz-request-id: b13be3abf4db4af33eaaaeb5bdb5b7bd7c7e82727476787a7c
Content-Length: 538
Content-Type: application/xml; charset=utf-8
Server: CTYUN

<?xml version="1.0" encoding="UTF-8"?>
  <ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <Name>docs</Name>
   <Prefix></Prefix>
   <Marker></Marker>
   <MaxKeys>1000</MaxKeys>
   <EncodingType></EncodingType>
   <IsTruncated>false</IsTruncated>
   <Contents>
    <Key>1.png</Key>
    <LastModified>2020-03-04T07:04:48.135Z</LastModified>
    <ETag>&amp;quot;01fa001a4c65b834696c5e0049fd5004&amp;quot;</ETag>
    <Size>25348</Size>
    <StorageClass>STANDARD</StorageClass>
    <Owner>
       <ID></ID>
       <DisplayName></DisplayName>
    </Owner>
  </Contents>
</ListBucketResult>
```

- **使用请求变量的示例：**

假设数据库中存储数据如下所示：

docs/sample.pdf

docs/Ant/sample.doc

docs/Feb/sample2.doc

docs/Feb/sample3.doc

docs/Feb/sample4.doc

要查询 prefix 为 docs/,delimiter 为/，同时 marker 为 docs/F，max-keys 为 40 的结果，请求头为

```
GET /?prefix=docs/&marker=docs/F&max-keys=40&delimiter=/ HTTP/1.1
Host: docs.oos-cn.ctyunapi.cn
Date: Mon, 03 Sep 2012 12:00:00 GMT
Authorization: AWS 81ebc16ddc8d2af82c90:thdUi9VAkzhkniLj96JIrOPGi0g
```

返回的结果集为：

```
<ListBucketResult xmlns="http://oos-cn.ctyunapi.cn/docs/2012-09-03/">
  <Name>docs</Name>
  <Prefix>docs/</Prefix>
  <Marker>docs/F</Marker>
  <MaxKeys>40</MaxKeys>
  <EncodingType></EncodingType>
  <Delimiter>/</Delimiter>
  <IsTruncated>false</IsTruncated>
  <Contents>
    <Key>sample.pdf</Key>
    <LastModified>2012-09-01T01:56:20.000Z</LastModified>
    <ETag>&amp;quot;bf1d737a4d46a19f3bced6905cc8b902&amp;quot;</ETag>
    <Size>142863</Size>
    <Owner>
      <ID> </ID>
      <DisplayName> </DisplayName>
    </Owner>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
  <CommonPrefixes>
    <Prefix>Feb/</Prefix>
    <Prefix>Ant/</Prefix>
  </CommonPrefixes>
</ListBucketResult>
```

4.3.5 DELETE Bucket

此操作用来删除 Bucket，但要求被删除 Bucket 中无 Object，即该 Bucket 中的所有 Object 都已被删除。

- **请求语法**

```
DELETE / HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Authorization: SignatureValue
```

- **请求示例**

删除名叫 docs 的 Bucket：

```
DELETE / HTTP/1.1
Host: docs.oos-cn.ctyunapi.cn
Date: Mon, 03 Sep 2012 12:00:00 GMT
Content-Type: application/xml;charset=utf-8
Authorization: SignatureValue
```

- **响应示例**

```
HTTP/1.1 204 No Content
x-amz-request-id: 4f549138d7bc46c4ce4134433845474d0c0e18020406080a0c
Date: Mon, 03 Sep 2012 12:00:00 GMT
Server: CTYUN
```

4.3.6 PUT Bucket Policy

在 PUT 操作的 url 中加上 Policy，可以进行添加或修改 Policy 的操作。如果 Bucket 已经存在了 Policy，此操作会替换原有 Policy。只有根用户和拥有 PUT Bucket Policy 权限的用户才能执行此操作，否则会返回 403 AccessDenied 错误。

注意：如果 Bucket 的属性为私有或者公共读，使用该接口配置允许任何用户可以向该 Bucket 写文件的策略时，请联系天翼云客服评估审核后开通。

- **请求语法**

```
PUT /?policy HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Authorization: SignatureValue

Policy written in JSON
```

请求的内容是一个包含 Policy 语句的 JSON 串，详见 **Bucket Policy 安全策略**。

- **请求示例**

```
PUT /?policy HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
Date: Sun, 28 Apr 2024 02:02:21 GMT
Content-MD5: +nl0RJvKLaXlRhwFXiBLVw==
Content-Type: application/octet-stream
X-Amz-Content-Sha256: c71bbdea6d26f41a56ce0312bfeadffa06718956c925dbc82af5df7bbad6a58f
Authorization: SignatureValue
Connection: keep-alive
Content-Length: 465

{
  "Version":"2012-10-17",
  "Id":"*",
  "Statement":[
    {
      "Sid":"*",
      "Effect":"Allow",
      "Principal":{ "CTYUN": "arn:ctyun:iam::32fefj64y54gc:user/test1" },
      "Action":"oos:*",
```

```
      "Resource":"arn:ctyun:oos:::example-bucket/*",
      "Condition":{
        "StringLike":{
          "ctyun:Referer":[
            "https://www.ctyun.cn/*",
            "https://ctyun.cn/*"
          ]
        }
      }
    }
  ]
}
```

- 响应示例

```
HTTP/1.1 200 OK
x-amz-request-id: 0f512a6284004715f16457665b686a702f313a2527292b2d2f
Date: Sun, 28 Apr 2024 02:02:22 GMT
Content-Length: 0
Server: CTYUN
```

4.3.7 GET Bucket Policy

此操作用来返回指定 Bucket 的 Policy。只有根用户和拥有 GET Bucket Policy 权限的子用户才能执行此操作，否则会返回 403 AccessDenied 错误。如果 Bucket 没有 Policy，返回 404，NoSuchPolicy 错误。

- **请求语法**

```
GET /?policy HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Authorization: SignatureValue
```

- **请求示例**

```
GET /?policy HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
Content-MD5: 1B2M2Y8AsgTpgAmY7PhCfg==
Content-Type: application/octet-stream
X-Amz-Content-Sha256: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
Date: Sun, 28 Apr 2024 06:13:38 GMT
Authorization: SignatureValue
```

- **响应示例**

```
HTTP/1.1 200 OK
Content-Type: application/xml;charset=UTF-8
x-amz-request-id: 2d3abf4e95fa4a092b9e91a095a2a4aa696b725f6163656769
Date: Sun, 28 Apr 2024 06:13:38 GMT
Content-Length: 465
Server: CTYUN

{
  "Version":"2012-10-17",
  "Id":"*",
  "Statement":[
    {
      "Sid":"*",
      "Effect":"Allow",
      "Principal":{ "CTYUN": "arn:ctyun:iam::32fefj64y54gc:user/test1" },
      "Action":"oos:*",
      "Resource":"arn:ctyun:oos:::example-bucket/*",
```

```
      "Condition":{
        "StringLike":{
          "ctyun:Referer":[
            "https://www.ctyun.cn/*",
            "https://ctyun.cn/*"
          ]
        }
      }
    }
  ]
}
```

4.3.8 DELETE Bucket Policy

在 DELETE 操作的 url 中加上 Policy，可以删除指定 Bucket 的 Policy。只有根用户和拥有 DELETE Bucket Policy 权限的子用户才能执行此操作，否则会返回 403 AccessDenied 错误。如果 Bucket 配置了 Policy，删除成功，返回 200 OK。如果 Bucket 没有配置 Policy，返回 204 NoContent。

- **请求语法**

```
DELETE /?policy HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Authorization: SignatureValue
```

- **请求示例**

```
DELETE /?policy HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
Content-MD5: 1B2M2Y8AsgTpgAmY7PhCfg==
Content-Type: application/octet-stream
X-Amz-Content-Sha256: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
Date: Sun, 28 Apr 2024 06:37:14 GMT
Authorization: SignatureValue
Connection: keep-alive
```

- **响应示例**

```
HTTP/1.1 204 No Content
x-amz-request-id: b10f8d28916049988bfef100f502040ac9cbd5bfc1c3c5c7c9
Date: Sun, 28 Apr 2024 06:37:14 GMT
Server: CTYUN
```

4.3.9 PUT Bucket Website

此操作用来配置网站托管属性。如果 Bucket 已经存在了 website，此操作会替换原有 website。只有根用户和拥有 PUT Bucket Website 权限的子用户才能执行此操作。

注意：

- 如果配置静态网站托管后，当匿名用户直接访问存储桶的域名，会将静态网站文件下载到本地。如果要实现访问静态网站时是预览网站内容，而非下载静态网站文件，静态网站域名须是存储桶绑定的已备案自定义域名，为存储桶绑定自定义域名请联系天翼云客服申请。
- OOS 自有网站托管域名不支持 HTTPS 访问，用户自定义域名支持 HTTPS 访问。如果需要支持 HTTPS 访问，请联系天翼云客服，提供域名证书，证书支持格式：crt+key 或者 PEM，请确保提供的证书在有效期内，建议证书有效期至少 1 年及以上，避免使用免费证书。
- 尽量避免目标存储桶名中带有“.”，否则通过 HTTPS 访问时可能出现客户端校验证书出错。
- 设置 Bucket 的网络配置请求消息体的上限是 10KiB。

网站托管配置步骤如下：

1）创建一个公共读属性的存储桶（Bucket）。

2）向天翼云客服提交工单，申请客户自定义域名添加白名单。

3）在域名管理中添加别名。

- 如果不使用 CDN 加速，将 Bucket 的 CNAME Record Value（*BucketName*.oos-website-cn.oos-*xx*.ctyunapi.cn）作为别名添加到域名管理系统中。
- 如果使用 CDN 加速，将 CDN 厂商提供的别名添加到域名管理系统中，然后在 CDN 回源地址中配置 OOS 侧的 CNAME Record Value，并将回源 host 配置为您的自定义域名（如 your***domain.com）。

说明：创建 Bucket 时显示的 Endpoint 为 oos-cn.ctyunapi.cn，该 Endpoint 是针对整个对象存储网络的域名，该域名在解析时，会根据用户地理位置的不同解析到不同的资源池地址。如果创建 Bucket 时有多个数据位置，系统默认选取创建时第一个有效数据位置作为 CNAME Record Value（*BucketName*.oos-website-cn.oos-

xx.ctyunapi.cn）。CNAME Record Value 可以通过控制台 Bucket 属性中的**网站**查看。如果创建 Bucket 时，只有一个数据位置可用，则在 Bucket 区域中展示的 CNAME Record Value 为 *BucketName*.oos-website-cn.oos-cn.ctyunapi.cn。所以如果使用静态网站托管，建议您根据 Bucket 区域属性中的数据位置，选择您想使用的数据位置的 CNAME Record Value 作为域名管理系统中的别名。例如您创建 Bucket 时有效数据位置为沈阳、兰州、成都、贵阳，则 Bucket 中展示的 CNAME Record Value 为 *BucketName*.oos-website-cn.oos-lnsy.ctyunapi.cn，您可以将 *BucketName*.oos-website-cn.oos-lnsy.ctyunapi.cn 作为别名，也可以将兰州、成都或者贵阳为域名的 CNAME Record Value 作为您的别名。

4) 上传文件

将网站的所有文件（html、CSS、js、图片等）上传到之前创建的 Bucket 中， 注意要保持文件之间的相对路径。

5) 配置 Bucket 网站属性：可以通过控制台或者调用本接口配置。

- **请求语法**

托管模式为当前存储桶

```
PUT /?website HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Content-Length: ContentLength
Authorization: SignatureValue

<WebsiteConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
     <IndexDocument>
            <Suffix>index.html</Suffix>
      </IndexDocument>
      <ErrorDocument>
            <Key>errorDocument.html</Key>
       </ErrorDocument>
       <RoutingRules>
            <RoutingRule>
                 <Condition>
                   <HttpErrorCodeReturnedEquals>string</HttpErrorCodeReturnedEquals>
                   <KeyPrefixEquals>string</KeyPrefixEquals>
```

```
                </Condition>
                <Redirect>
                    <HostName>string</HostName>
                    <Protocol>string</Protocol>
                    <ReplaceKeyPrefixWith>string</ReplaceKeyPrefixWith>
                    <ReplaceKeyWith>string</ReplaceKeyWith>
                </Redirect>
            </RoutingRule>
    </RoutingRules>
</WebsiteConfiguration>
```

托管模式为重定向请求

```
PUT /?website HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Content-Length: ContentLength
Authorization: SignatureValue

<WebsiteConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
      <RedirectAllRequestsTo>
            <HostName>string</HostName>
            <Protocol>string</Protocol>
      </RedirectAllRequestsTo>
</WebsiteConfiguration>
```

- **请求元素**

名称	描述	是否必须
WebsiteConfiguration	请求的容器。 类型：容器 子节点：IndexDocument、ErrorDocument、RoutingRules 或 RedirectAllRequestsTo **注意：**IndexDocument 和 RedirectAllRequestsTo 互斥。如果填写了 RedirectAllRequestsTo，则	是

	IndexDocument、ErrorDocument、RoutingRules 都不能填写。	
IndexDocument	Suffix 元素的容器。 **注意**：如果未填写 RedirectAllRequestsTo，则 IndexDocument 必填。 类型：容器 父节点：WebsiteConfiguration 子节点：Suffix	否
Suffix	在请求 website endpoint 上的路径时，Suffix 会被加在请求的后面。例如，如果 suffix 是 Index.html，而你请求的是 bucket/images/，那么返回的响应是名为 images/index.html 的 Object。 类型：字符串 父节点：IndexDocument	是
ErrorDocument	Key 的容器。 类型：容器 父节点：WebsiteConfiguration 子节点：Key	否
Key	如果出现 4XX 错误，会返回指定的 Object。 类型：字符串 有效值：长度为 1~1024 的字符串。 父节点：ErrorDocument	是
RoutingRules	托管模式配置到当前 Bucket 的重定向规则容器。 **注意**：如果 RoutingRules 下有多个 RoutingRule，各 RoutingRule 之间无影响，按照配置的先后顺序向下执行，当有一个满足条	否

	件时，就不再继续向后匹配。如果都没有匹配，就不使用重定向规则。 类型：容器 父节点：WebsiteConfiguration 子节点：RoutingRule	
RoutingRule	重定向规则的容器。一条重定向规则包含一个 Condition 和一个 Redirect，当 Condition 匹配时，Redirect 生效。容器中至少要有一个重定向规则。 **注意**：一个 RoutingRule 下，出现多个 Condition 和 Redirect 时，以最后一个为准。 类型：容器 父节点：RoutingRules 子节点：Condition、Redirect	是
Condition	描述重定向规则匹配的条件的容器。如果重定向规则匹配的条件未配置，则重定向规则将应用于所有请求。 **注意**：该容器可以不配置，如果配置，则至少应该包含 HttpErrorCodeReturnedEquals、KeyPrefixEquals 中的一个。 类型：容器 父节点：RoutingRule 子节点：HttpErrorCodeReturnedEquals、KeyPrefixEquals	否
HttpErrorCodeReturnedEquals	指定 Redirect 生效时的 HTTP 错误码。当发生错误时，如果错误码等于这个值，那么 Redirect 生效。	否

	注意：HttpErrorCodeReturnedEquals 和 KeyPrefixEquals 同时存在时，只有都匹配时，Redirect 才生效。 类型：字符串 有效值：[400, 417]，[500, 505]。 例如：当返回的 http 错误码为 404 时重定向到 NotFound.html，可以将 Condition 中的 HttpErrorCodeReturnedEquals 设置为 404，Redirect 中的 ReplaceKeyWith 设置为 `NotFound.html`。 父节点：Condition	
KeyPrefixEquals	重定向规则生效时的文件名的前缀。 **注意**：HttpErrorCodeReturnedEquals 和 KeyPrefixEquals 同时存在时，只有都匹配时，Redirect 才生效。 类型：字符串 取值：长度为 0-1024 的字符串。 父节点：Condition	否
Redirect	重定信息容器。 **注意**：Redirect 配置包含的元素可以为空，也可以包含以下元素:Protocol、HostName、ReplaceKeyPrefixWith、ReplaceKeyWith。当某一元素存在多条值时以最后一条为准。 类型：容器 父节点：RoutingRule 子节点：Protocol、HostName、ReplaceKeyPrefixWith、ReplaceKeyWith	是
Protocol	重定向请求时使用的协议。	否

	类型：字符串 有效值：http、https，默认值为 http。 父节点：Redirect 或者 RedirectAllRequestsTo	
HostName	重定向请求时使用的站点名。 如果父节点为 RedirectAllRequestsTo，此项必须填写。 类型：字符串 有效值：1~1024 个字符，不能包含空格。父节点为 Redirect，也不能包含斜杠(/)。 父节点：Redirect 或者 RedirectAllRequestsTo	否
ReplaceKeyPrefixWith	重定向请求时使用的文件名前缀。 **注意**：ReplaceKeyPrefixWith 与 ReplaceKeyWith 不能同时存在。 类型：字符串 有效值：0~1024 个字符。 父节点：Redirect	否
ReplaceKeyWith	指定重定向请求时使用的文件名。 **注意**：ReplaceKeyPrefixWith 与 ReplaceKeyWith 不能同时存在。 类型：字符串 有效值：0~1024 个字符。 父节点：Redirect	否
RedirectAllRequestsTo	托管模式为重定向请求的容器。 **注意**：如果未填写 IndexDocument，则 RedirectAllRequestsTo 必填。如果填写了 RedirectAllRequestsTo，则 IndexDocument、ErrorDocument、RoutingRules 都不能填写。 父节点：WebsiteConfiguration	否

	子节点：HostName、Protocol	

- **请求示例 1**

指定 index.html 为首页，error.html 为错误页。

```
PUT /?website HTTP/1.1
x-amz-content-sha256: UNSIGNED-PAYLOAD
x-amz-date: 20220718T084033Z
Host: example-bucket.oos-cn.ctyunapi.cn
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
Content-Length: 268
Authorization: SignatureValue

<WebsiteConfiguration xmlns='http://s3.amazonaws.com/doc/2006-03-01/'>
      <IndexDocument>
            <Suffix>index.html</Suffix>
      </IndexDocument>
      <ErrorDocument>
            <Key>error.html</Key>
      </ErrorDocument>
</WebsiteConfiguration>
```

- **响应示例 1**

```
HTTP/1.1 200 OK
Date: Mon, 18 Jul 2022 08:40:33 GMT
x-amz-request-id: 91dd8b753eba4237aefaf102080508e70a14f6f8fbf6ff0bec
Content-Length: 0
Server: CTYUN
```

- **请求示例 2**

将所有的请求重定向到主机 docs.oos-cn.ctyunapi.cn，且使用 https 协议。

```
PUT /?website HTTP/1.1
x-amz-content-sha256: UNSIGNED-PAYLOAD
x-amz-date: 20220718T084539Z
Host: example-bucket.oos-cn.ctyunapi.cn
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
Content-Length: 252
```

```
Authorization: SignatureValue

<WebsiteConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
       <RedirectAllRequestsTo>
            <HostName>docs.oos-cn.ctyunapi.cn</HostName>
            <Protocol>https</Protocol>
      </RedirectAllRequestsTo>
</WebsiteConfiguration>
```

- **响应示例 2**

```
HTTP/1.1 200 OK
Date: Mon, 18 Jul 2022 08:45:39 GMT
x-amz-request-id: 98449be04a8e4abb93dfd6e7edeaedcceff9dbdde0dbe4f0d1
Content-Length: 0
Server: CTYUN
```

- **请求示例 3**

  指定 index.html 为首页，error.html 为错误页。将所有前缀为“docs”的请求，全部重定向至本 Bucket 中前缀为“documents/”的文件上。

```
PUT /?website HTTP/1.1
x-amz-content-sha256: UNSIGNED-PAYLOAD
x-amz-date: 20220718T085146Z
Host: example-bucket.oos-cn.ctyunapi.cn
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
Content-Length: 471
Authorization: SignatureValue

<WebsiteConfiguration xmlns='http://s3.amazonaws.com/doc/2006-03-01/'>
  <IndexDocument>
    <Suffix>index.html</Suffix>
  </IndexDocument>
  <ErrorDocument>
    <Key>error.html</Key>
  </ErrorDocument>
  <RoutingRules>
    <RoutingRule>
    <Condition>
      <KeyPrefixEquals>docs</KeyPrefixEquals>
```

```
    </Condition>
    <Redirect>
      <ReplaceKeyPrefixWith>documents/</ReplaceKeyPrefixWith>
    </Redirect>
    </RoutingRule>
  </RoutingRules>
</WebsiteConfiguration>
```

- **响应示例 3**

```
HTTP/1.1 200 OK
Date: Mon, 18 Jul 2022 08:51:46 GMT
x-amz-request-id: 01413e0eca2e4cbf66b2a9bac0bdc09fc2ccaeb0b3aeb7c3a4
Content-Length: 0
Server: CTYUN
```

- **请求示例 4**

  指定 index.html 为首页，error.html 为错误页。如果发生 404 错误，重定向到网址“docs.oos-cn.ctyunapi.cn”上前缀为“report-404/”的文件。

```
PUT /?website HTTP/1.1
x-amz-content-sha256: UNSIGNED-PAYLOAD
x-amz-date: 20220718T090739Z
User-Agent: Java/1.8.0_221
Host: example-bucket.oos-cn.ctyunapi.cn
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
Content-Length: 538
Authorization: SignatureValue

<WebsiteConfiguration xmlns='http://s3.amazonaws.com/doc/2006-03-01/'>
  <IndexDocument>
    <Suffix>index.html</Suffix>
  </IndexDocument>
  <ErrorDocument>
    <Key>error.html</Key>
  </ErrorDocument>
  <RoutingRules>
    <RoutingRule>
    <Condition>
      <HttpErrorCodeReturnedEquals>404</HttpErrorCodeReturnedEquals>
```

```
    </Condition>
    <Redirect>
       <HostName>docs.oos-cn.ctyunapi.cn</HostName>
      <ReplaceKeyPrefixWith>report-404/</ReplaceKeyPrefixWith>
    </Redirect>
    </RoutingRule>
  </RoutingRules>
</WebsiteConfiguration>
```

- **响应示例 4**

```
HTTP/1.1 200 OK
Date: Mon, 18 Jul 2022 09:07:39 GMT
x-amz-request-id: 2814d402a9f64041024e45565c595c3b5e684a4c4f4a535f40
Content-Length: 0
Server: CTYUN
```

- **请求示例 5**

  指定 index.html 为首页，error.html 为错误页。将前缀为“images/”的请求，重定向至本 Bucket 内的“errorpage.html”文件。

```
PUT /?website HTTP/1.1
x-amz-content-sha256: UNSIGNED-PAYLOAD
x-amz-date: 20220718T091138Z
Host: example-bucket.oos-cn.ctyunapi.cn
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
Content-Length: 465
Authorization: SignatureValue

<WebsiteConfiguration xmlns='http://s3.amazonaws.com/doc/2006-03-01/'>
  <IndexDocument>
    <Suffix>index.html</Suffix>
  </IndexDocument>
  <ErrorDocument>
    <Key>error.html</Key>
  </ErrorDocument>
  <RoutingRules>
    <RoutingRule>
    <Condition>
      <KeyPrefixEquals>images/</KeyPrefixEquals>
```

```
    </Condition>
    <Redirect>
      <ReplaceKeyWith>errorpage.html</ReplaceKeyWith>
    </Redirect>
    </RoutingRule>
  </RoutingRules>
</WebsiteConfiguration>
```

- **响应示例 5**

```
HTTP/1.1 200 OK
Date: Mon, 18 Jul 2022 09:11:38 GMT
x-amz-request-id: 2ef16ade344d4c111f6b6273797679587b8567696c67707c5d
Content-Length: 0
Server: CTYUN
```

4.3.10 GET Bucket Website

此操作用来获得指定 Bucket 的 website。只有根用户和拥有 GET Bucket Website 权限的子用户才能执行此操作，否则会返回 403 AccessDenied 错误。

- **请求语法**

```
GET /?website HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Authorization: SignatureValue
```

- **响应结果**

名称	描述
WebsiteConfiguration	响应的容器。 类型：容器 子节点：IndexDocument、ErrorDocument、RoutingRules 或 RedirectAllRequestsTo
IndexDocument	Suffix 元素的容器。 类型：容器 父节点：WebsiteConfiguration 子节点：Suffix
Suffix	在请求 website endpoint 时，Suffix 会被加在请求的后面。 类型：字符串 父节点：IndexDocument
ErrorDocument	Key 的容器。 类型：容器 父节点：WebsiteConfiguration 子节点： Key
Key	如果出现 4XX 错误，返回的 Object。 类型：字符串

	父节点：ErrorDocument
RoutingRules	托管模式配置到当前 Bucket 的重定向规则容器。 类型：容器 父节点：WebsiteConfiguration 子节点：RoutingRule
RoutingRule	具体重定向规则的容器。 父节点：RoutingRules 子节点：Condition、Redirect
Condition	描述重定向规则匹配的条件的容器。 类型：容器 父节点：RoutingRule 子节点：HttpErrorCodeReturnedEquals、KeyPrefixEquals
HttpErrorCodeReturnedEquals	Redirect 生效时的 HTTP 错误码。 类型：字符串 父节点：Condition
KeyPrefixEquals	重定向规则生效时的文件名的前缀。 类型：字符串 父节点：Condition
Redirect	重定信息容器。 类型：容器 父节点：RoutingRule 子节点：Protocol、HostName、ReplaceKeyPrefixWith、ReplaceKeyWith
Protocol	描述重定向请求时使用的协议。 类型：字符串 父节点：Redirect 或者 RedirectAllRequestsTo
HostName	重定向请求时使用的站点名。 类型：字符串

	父节点：Redirect 或者 RedirectAllRequestsTo
ReplaceKeyPrefixWith	重定向请求时使用的文件名前缀。 类型：字符串 父节点：Redirect
ReplaceKeyWith	重定向请求时使用的文件名。 类型：字符串 父节点：Redirect
RedirectAllRequestsTo	托管模式为重定向请求的容器。 父节点：WebsiteConfiguration 子节点：HostName、Protocol

- **请求示例**

```
GET /?website HTTP/1.1
x-amz-content-sha256: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date: 20220718T091530Z
Host: example-bucket.oos-cn.ctyunapi.cn
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
Authorization: SignatureValue
```

- **响应示例**

```
HTTP/1.1 200 OK
Content-Type: application/xml;charset=UTF-8
Date: Mon, 18 Jul 2022 09:15:30 GMT
x-amz-request-id: 88d52b33bef4440b3a867d8e9491947396a0828487828b9778
Content-Length: 430
Server: CTYUN

<?xml version="1.0" encoding="UTF-8"?>
<WebsiteConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <IndexDocument>
       <Suffix>index.html</Suffix>
```

```
    </IndexDocument>
    <ErrorDocument>
        <Key>error.html</Key>
    </ErrorDocument>
    <RoutingRules>
        <RoutingRule>
            <Condition>
                <KeyPrefixEquals>images/</KeyPrefixEquals>
            </Condition>
            <Redirect>
                <ReplaceKeyWith>errorpage.html</ReplaceKeyWith>
            </Redirect>
        </RoutingRule>
    </RoutingRules>
</WebsiteConfiguration>
```

4.3.11 DELETE Bucket Website

此操作用来删除指定 Bucket 的 website。只有根用户和拥有 DELETE Bucket Website 权限的子用户才能执行此操作，否则会返回 403 AccessDenied 错误。如果 Bucket 没有设置 Website 或者 Website 删除成功，返回 200 OK。

- **请求语法**

```
DELETE /?website HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Authorization: SignatureValue
```

- **请求示例**

```
DELETE /?website HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
Date: Mon, 03 Sep 2012 12:00:00 GMT
Authorization: SignatureValue
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id: ebca116ac80a43008bfef100f502040ac9cbcfbfc1c3c5c7c9
Date: Mon, 03 Sep 2012 12:00:00 GMT
Server: CTYUN
```

4.3.12 List Multipart Uploads

该接口用于列出所有已经通过 Initiate Multipart Upload 请求初始化，但未完成或未终止的分片上传过程。

响应中最多返回 1000 个分片上传过程的信息，它既是响应能返回的最大分片上传过程数目，也是请求的默认值。用户也可以通过设置 max-uploads 参数来限制响应中的分片上传过程数目。如果当前的分片上传过程数超出了这个值，则响应中会包含一个值为 true 的 IsTruncated 元素。如果用户要列出多于这个值的分片上传过程信息，则需要继续调用 List Multipart Uploads 请求，并在请求中设置 *key-marker* 和 *upload-id-marker* 参数。

在响应体中，分片上传过程的信息通过 key 来排序。如果用户的应用程序中启动了多个使用同一 key 文件开头的分片上传过程，那么响应体中分片上传过程首先是通过 key 来排序，在相同 key 的分片上传内部则是按上传启动的起始时间的升序来进行排列。

- **请求语法**

```
GET /?uploads HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: Date
Authorization: SignatureValue
```

- **请求参数**

名称	描述	是否必须
delimiter	分隔符，一个用来对关键字们进行分组的字符。所有的关键字都包含 delimiter 和 prefix 间的相同子串，prefix 之后第一个遇到 delimiter 的字符串都会加到一个叫 CommonPrefix 的组中。如果没有特别定义 prefix，所有的关键字都会被返回，但不会有 CommonPrefix。delimiter 只支持”/”，不支持其他分隔符。 类型：字符串	否
max-uploads	设置返回的分片上传过程的最大数目。 类型：整型 取值：[1, 1000]	否

	默认值：1000	
key-marker	与 *upload-id-marker* 参数一起，该参数指定列表操作从什么位置后面开始。如果 *upload-id-marker* 参数没有被指定，那么只有比 *key-marker* 参数指定的 key 更大的 key 会被列出。如果 *upload-id-marker* 参数被指定，任何包含与 *key-marker* 指定值相等或大于 key 的分片上传过程都会被包括，假设这些分片上传过程包含了比 *upload-id-marker* 指定值更大的 ID。 类型：字符串	否
prefix	该参数用于列出那些以 prefix 为前缀的正在进行的上传过程，用户可以使用多个 prefix 来把一个 Bucket 分成不同的组（可以考虑采用类似文件系统中的文件夹的作用那样，使用 prefix 来对 key 进行分组）。 类型：字符串	否
encoding-type	指定响应中 KeyMarker、NextKeyMarker、Key、Prefix、Delimiter 的编码类型。如果 KeyMarker、NextKeyMarker、Key、Prefix、Delimiter 包含 xml 1.0 标准不支持的控制字符，可通过设置该参数对响应中的 KeyMarker、NextKeyMarker、Key、Prefix、Delimiter 进行编码。 类型：字符串 取值：url，字母不区分大小写。	否
upload-id-marker	与 *key-marker* 参数一起，指定列表操作从什么位置后面开始。如果 *key-marker* 没有被指定，*upload-id-marker* 参数也会被忽略。否则，任何 key 值等于或大于 *key-marker* 的上传过程都会被包含在列表中，只要 ID 大于 *upload-id-marker* 指定的值。	否

- **响应结果**

名称	描述
ListMultipartUploadsResult	包含整个响应的容器。 类型：容器 子节点：Bucket、KeyMarker、UploadIdMarker、NextKeyMarker、NextUploadIdMarker、Maxuploads、Delimiter、Prefix、CommonPrefixes、IsTruncated、EncodingType、Upload
Bucket	分片上传对应的存储桶名称。 类型：字符串 父节点：ListMultipartUploadsResult
KeyMarker	指定 key 值，在这个 key 当前位置或它之后开始列表操作。 类型：字符串 父节点：ListMultipartUploadsResult
UploadIdMarker	指定分片上传 ID，在这个 ID 所在位置之后开始列表操作。 类型：字符串 父节点：ListMultipartUploadsResult
NextKeyMarker	当此次列表不能将所有正在执行的分片上传过程列举完成时，NextKeyMarker 作为下一次列表请求的 key-marker 参数的值。 类型：字符串 父节点：ListMultipartUploadsResult
NextUploadIdMarker	当此次列表不能将所有正在执行的分片上传过程列举完成时，NextKeyMarker 作为下一次列表请求的 upload-id-marker 参数的值。 类型：字符串 父节点：ListMultipartUploadsResult
MaxUploads	响应中包含的上传过程的最大数目。 类型：字符串 父节点：ListMultipartUploadsResult

IsTruncated	标识此次分片上传过程中的所有片段是否全部被列出，如果为 true 则表示没有全部列出。如果分片上传过程的片段数超过了 MaxParts 元素指定的最大数，则会导致一次列表请求无法将所有片段数列出。 类型：Boolean 父节点：ListMultipartUploadsResult
EncodingType	KeyMarker、NextKeyMarker、Key、Prefix、Delimiter 的编码类型。 类型：字符串 父节点：ListMultipartUploadsResult
Upload	某个分片上传过程的容器，响应体中可能包含 0 个或多个 Upload 元素。 类型: 容器 父节点: ListMultipartUploadsResult 子节点: Key、UploadId、Initiator、Owner、StorageClass、Initiated
Key	分片上传过程起始位置文件的 Key 值。 类型: String 父节点: Upload
UploadId	分片上传过程的 ID 号。 类型: 整型 父节点: Upload
Initiator	指定执行此次分片上传过程的用户账户。 类型：容器 父节点：Upload 子节点：ID，DisplayName
ID	OOS 账户的 ID。 类型：字符串

	父节点：Initiator
StorageClass	文件的存储类型： ● STANDARD ● STANDARD_IA 类型：字符串 父节点：Upload
Initiated	分片上传过程启动的时间。 类型：Date 父节点：Upload
Prefix	如果请求中包含了 Prefix 参数，则这个字段会包含 Prefix 的值。返回的结果只包含那些 key 值以 Prefix 开头的文件。 类型：字符串 父节点：ListMultipartUploadsResult
Delimiter	分割符，用来对关键字们进行分组的字符。所有的关键字都包含 delimiter 和 prefix 间的相同子串，prefix 之后第一个遇到 delimiter 的字符串都会加到一个叫 CommonPrefix 的组中。如果没有特别定义 prefix，所有的关键字都会被返回，但不会有 CommonPrefix。 类型：字符串 父节点：ListMultipartUploadsResult
CommonPrefixes	当定义 delimiter 之后，返回结果中会包含 CommonPrefixes，CommonPrefix 中包含以 prefix 开头，delimiter 结束的字符串组合，比如 prefix 是 note/，同时 delimiter 是斜杠（/），结果中的 note/summer/july/lotus.jpg 将返回 note/summer/，其余结果将按照 max-uploads 要求返回。 类型：容器 父节点：ListMultipartUploadsResult 子节点：Prefix

CommonPrefixes.Prefix	如果请求中不包含 Prefix 参数，那么这个元素只显示那些在 delimiter 字符第一次出现之前的 key 的子字符串，且这些 key 不在响应的其它位置出现。 如果请求中包含 Prefix 参数，那么这个元素显示在 prefix 之后，到第一次出现 delimiter 之间的子串。 类型：字符串 父节点：CommonPrefixes

- **请求示例**

下面的请求列出正在进行的三个分片上传过程。请求指定了 *max-uploads* 参数来设置响应中返回的分片上传过程的最大数目。

```
GET /?uploads&max-uploads=3 HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
Date: Mon, 1 Nov 2010 20:34:56 GMT
Authorization: SignatureValue
```

- **响应示例**

下面的示例表示这次 List 操作无法将分片上传过程列举完，需要指定 NextKeyMarker 和 NextUploadIdMarker 元素。用户需要在下一次 List 操作中指定这两个值。也就是说，在下一次 List 操作中指定 *key-marker*=my-image.my（NextKeyMarker 的值）和 *upload-id-marker*=AW56IGBkZQEgd3h6ILVsdmluUydzIWVwbP9hTCAmQZlsQZW（NextUploadIdMarker 的值）。

响应示例中也显示了一个两个分片上传过程拥有同一个 key（my-image.my）的例子。响应包含了两个通过 key 来排序的上传过程，在每个 key 内部上传过程是根据上传启动的起始时间来排序的。

```
HTTP/1.1 200 OK
x-amz-request-id: eec6eee8e3ff42e305786b7a6f7c7e8443454c393b3d3f4143
Date: Mon, 1 Nov 2010 20:34:56 GMT
Content-Length: 1359
Server: CTYUN
<?xml version="1.0" encoding="UTF-8"?>
<ListMultipartUploadsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
```

```
<Bucket>bucket</Bucket>
<KeyMarker></KeyMarker>
<UploadIdMarker></UploadIdMarker>
<NextKeyMarker> my-image.my</NextKeyMarker>
<NextUploadIdMarker>1638428231343309201</NextUploadIdMarker>
<MaxUploads>3</MaxUploads>
<IsTruncated>true</IsTruncated>
<EncodingType></EncodingType>
<Upload>
   <Key>my-divisor</Key>
   <UploadId>1638428231343309198</UploadId>
   <Initiator>
       <ID>mailaddress@ctyun.cn</ID>
       <DisplayName> </DisplayName>
   </Initiator>
   <Owner>
       <ID></ID>
       <DisplayName></DisplayName>
   </Owner>
   <StorageClass>STANDARD</StorageClass>
   <Initiated>2010-11-10T20:48:33.000Z</Initiated>
</Upload>
<Upload>
   <Key>my-movie.m2ts</Key>
   <UploadId>1638428231343309199</UploadId>
   <Initiator>
       <ID>mailaddress@ctyun.cn</ID>
       <DisplayName> </DisplayName>
   </Initiator>
   <Owner>
       <ID> </ID>
       <DisplayName> </DisplayName>
   </Owner>
   <StorageClass>STANDARD</StorageClass>
   <Initiated>2010-11-10T20:48:33.000Z</Initiated>
 </Upload>
 <Upload>
     <Key> my-image.my</Key>
     <UploadId>1638428231343309200</UploadId>
     <Initiator>
         <ID>mailaddress@ctyun.cn</ID>
         <DisplayName> </DisplayName>
```

```
        </Initiator>
        <Owner>
            <ID> </ID>
            <DisplayName> </DisplayName>
        </Owner>
        <StorageClass>STANDARD</StorageClass>
        <Initiated>2010-11-10T20:49:33.000Z</Initiated>
    </Upload>
</ListMultipartUploadsResult>
```

4.3.13 PUT Bucket Logging

此操作用来添加/修改/删除 logging 的操作。如果 Bucket 已经存在了 logging，此操作会替换原有 logging。只有根用户和拥有 PUT Bucket Logging 权限的子用户才能执行此操作，否则会返回 403 AccessDenied 错误。

- **请求语法**

```
PUT /?logging HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Authorization: SignatureValue

Request elements vary depending on what you're setting.
```

- **请求元素**

名称	描述	是否必须
BucketLoggingStatus	请求的容器。 类型：容器 子节点：LoggingEnabled	是
LoggingEnabled	日志信息的容器，当启动日志时，需要包含这个元素。 类型：容器。 父节点：BucketLoggingStatus 子节点：TargetBucket、TargetPrefix	否
TargetBucket	指定要保存 log 的 Bucket，OOS 会向此 Bucket 存储日志。可以设置任意一个你拥有的 Bucket 作为 TargetBucket，包括启动日志的 bucket 本身。你也可以设置将多个 Bucket 的日志存放到一个 TargetBucket 中，在这种情况下，你需要为每个源 Bucket 设置不同的 TargetPrefix，以便不同 Bucket 的 log 可以被区分出来。 类型：字符串 父节点：LoggingEnabled	否
TargetPrefix	生成的 log 文件将以此为前缀命名。	否

	类型：字符串 父节点：LoggingEnabled	

- **请求示例 1**

  启动日志。

```
PUT /?logging HTTP/1.1
Host: docs.oos-cn.ctyunapi.cn
Content-Type: application/xml; charset=utf-8
Date: Mon, 03 Sep 2012 12:00:00 GMT
Authorization: SignatureValue

<?xml version="1.0" encoding="UTF-8"?>
<BucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01">
    <LoggingEnabled>
        <TargetBucket>mybucketlogs</TargetBucket>
        <TargetPrefix>mybucket-access_log-/</TargetPrefix>
    </LoggingEnabled>
</BucketLoggingStatus>
```

- **响应示例 1**

```
HTTP/1.1 200 OK
x-amz-request-id: 0275747fbe8742a179ecdfeee3f0f2f8b7b9c3adafb1b3b5b7
Date: Mon, 03 Sep 2012 12:00:00 GMT
Server: CTYUN
```

- **请求示例 2**

  取消日志。

```
PUT /?logging HTTP/1.1
Host: docs.oos-cn.ctyunapi.cn
Date: Mon, 03 Sep 2012 12:00:00 GMT
Authorization: SignatureValue

<?xml version="1.0" encoding="UTF-8"?>
<BucketLoggingStatus xmlns=http://doc.s3.amazonaws.com/2006-03-01/>
</BucketLoggingStatus>
```

- **响应示例 2**

```
HTTP/1.1 200 OK
x-amz-request-id: 63b0c5fe79b7418d9407fa09fe0b0d13d2d4d8c8caccced0d2
```

```
Date: Mon, 03 Sep 2012 12:00:00 GMT
Server: CTYUN
```

- **日志名称**

```
TargetPrefix<RegionName>/UTCYYYY-MM-DD-hh-mm-ss-唯一字符串_id
```

其中 YYYY，MM，DD，hh，mm，ss 分别代表日志发送时的年，月，日，小时，分钟，秒。TargetPrefix 是用户在启动 Bucket 日志时配置的，如果未配置，则该项不存在。唯一字符串是服务端生成的，某些场景下不会生成。id 表示日志服务器序号。

系统不会删除旧的日志文件。用户可以自己删除以前的日志文件，可以在 List Objects 时指定 prefix 参数，挑选出旧的日志文件，然后删除。

当客户端的请求到达后，日志记录不会被立刻推送到 TargetBucket 中，会延迟一段时间。

- **日志格式**

字段名称	示例	备注
BucketOwner	mailaddress@ctyun.cn	源 Bucket 的 Owner。
Bucket 名称	mybucket	请求的 Bucket，如果 OOS 收到一个错误的 Bucket 名称，那么这个请求不会出现在任何 log 中。
Time	04/08/2012 22:34:02 zz	请求到达的时间，格式是 dd/MM/yyyy HH:mm:ss zz。
Remote IP	72.21.206.5	请求者的 IP 地址。中间的代理和防火墙可能会隐藏实际发出请求的机器的地址。
Requester	● 根用户邮箱：mailaddress@ctyun.cn ● 子用户：arn:ctyun:iam::10rc2arpn6306:user/test_user ● 匿名用户：Anonymous	● 如果是根用户，显示根用户的邮箱，如：mailaddress@ctyun.cn。 ● 如果子是用户，显示子用户的 ARN，如：arn:ctyun:iam::10rc2arpn6306:user/test_user。

		● 如果是匿名访问，显示“Anonymous”。
Request ID	e4dd82b2f0994896	Request ID 是 OOS 生成的一个字符串，用于唯一标示每个请求。
Operation	REST.PUT.OBJECT	REST.HTTP_method.resource_type。
Key	/photos/2012/08/puppy.jpg	请求的“Key”部分，URL 编码。如果请求中没有指定文件，显示“-”。
Request-URI	"GET /mybucket/photos/2012/08/puppy.jpg HTTP/1.1"	HTTP 请求中的 Request-URI 部分。
HTTP status	200	响应的 HTTP 状态码。
Error Code	NoSuchBucket	OOS 错误码，如果没有错误，显示“-”。
Bytes Sent	2662992	写请求或读响应发送的字节数，不包括 HTTP 协议的头。如果是 0，显示“-”。
Object Size	3462992	Object 的总大小。
Time	70	从收到请求到发出响应的时间，单位是 ms。
Referer	"http://oos.ctyun.cn"	HTTP Referer 请求头的值。
User-Agent	"curl/7.15.1"	HTTP User-Agent 请求头的值。
Version Id	3HL4kqtJvjVBH40Nrjfkd	请求中的 versionId 参数，如果没有，显示“-”。
Turn-Around Time	1024	OOS 处理请求的时间，单位毫秒。即 OOS 接到客户端请求最后一个字节的时间，至 OOS 向客户端发送第一个响应字节的时间。

		说明：响应码为 200 和 206 的 GET Object 请求显示处理请求时间，其他操作显示“-”。

4.3.14 GET Bucket Logging

此操作用来获得指定 Bucket 的 logging。只有根用户和拥有 GET Bucket Logging 权限的子用户才能执行此操作，否则会返回 403 AccessDenied 错误。

- **请求语法**

```
GET /?logging HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Authorization: SignatureValue
```

- **响应结果**

名称	描述
BucketLoggingStatus	响应的容器。 类型：容器 子节点：LoggingEnabled
LoggingEnabled	日志信息的容器，当启动日志时，包含这个元素。否则此元素及其子元素都不显示。 类型：容器 父节点：BucketLoggingStatus 子节点：TargetBucket、TargetPrefix
TargetBucket	保存 log 的 Bucket，OOS 会向此 Bucket 存储日志。 类型：字符串 父节点：LoggingEnabled
TargetPrefix	生成的 log 文件将以此为前缀命名。 类型：字符串 父节点：LoggingEnabled

- **请求示例**

```
GET /?logging HTTP/1.1
Host: docs.oos-cn.ctyunapi.cn
Date: Mon, 03 Sep 2012 12:00:00 GMT
Authorization: SignatureValue
```

- **响应示例**

以下是设置了日志的响应示例

```
HTTP/1.1 200 OK
x-amz-request-id: 7b6bfbc182504e46f26558675c696b7130323c26282a2c2e30
Date: Mon, 03 Sep 2012 12:00:00 GMT
Server: CTYUN

<?xml version="1.0" encoding="UTF-8"?>
<BucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01">
    <LoggingEnabled>
        <TargetBucket>example-bucket</TargetBucket>
        <TargetPrefix>mybucket-access_log-/</TargetPrefix>
</LoggingEnabled>
```

以下是没有设置日志时的响应示例

```
HTTP/1.1 200 OK
x-amz-request-id: ef00a70d4f2b43416ee1d4e3d8e5e7edacaeb7a2a4a6a8aaac
Date: Mon, 03 Sep 2012 12:00:00 GMT
Server: CTYUN

<?xml version="1.0" encoding="UTF-8"?>
<BucketLoggingStatus xmlns="http://s3.amazonaws.com/doc/2006-03-01/"></BucketLoggingStatus>
```

4.3.15 HEAD Bucket

此操作用于判断 Bucket 是否存在，而且用户是否有权限访问。如果 Bucket 存在，而且用户有权限访问时，此操作返回 200 OK。否则，返回 404 不存在，或者 403 没有权限。

- **请求语法**

```
HEAD / HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Authorization: SignatureValue
```

- **请求示例**

```
HEAD / HTTP/1.1
Host: docs.oos-cn.ctyunapi.cn
Date: Mon, 03 Sep 2012 12:00:00 GMT
Authorization: SignatureValue
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id: d7a3025538f74a8fa4170a190e1b1d23e2e4e8d8dadcdee0e2
Date: Mon, 03 Sep 2012 12:00:00 GMT
Server: CTYUN
```

4.3.16 PUT Bucket Lifecycle

此操作用来设置 Bucket 生命周期规则。只有根用户和具有 PUT Bucket Lifecycle 权限的子用户才能执行此操作。

通过设置存储桶的生命周期规则，可以：

- 删除与生命周期规则匹配的文件：当文件的生命周期到期时，OOS 会异步删除它们。生命周期中配置的到期时间和实际删除时间之间可能会有一段延迟。但文件到期被删除后，用户将不需要为到期的文件付费。OOS 删除到期文件后，会在 Bucket log 中记录一条日志，操作项是"OOS.EXPIRE.OBJECT"。
- 将与生命周期规则匹配的文件由标准存储转换为低频访问存储：可以根据需要设置生命周期规则从文件最后一次修改时间生效，还是从文件最后一次访问时间生效。OOS 转换存储类型为低频访问存储后，会在 Bucket log 中记录一条日志，操作项是"OOS.TRANSITION_SIA.OBJECT"。

注意：

- 如果文件的生命周期规则设置的是到期后删除，文件到期后将被永久删除，无法恢复。
- 如果 Bucket 内的生命周期规则正在执行时被修改配置，则修改后的配置并不立即生效，需等原生命周期规则执行完成后才能生效。
- 如果 Bucket 没有配置过生命周期规则，执行该操作将创建新的生命周期规则。如果 Bucket 已存在生命周期规则，则执行此操作将覆盖原有规则。
- 每个 Bucket 最多创建 1000 条生命周期规则。
- 同一 Bucket，同一类型（到期删除或者到期转成低频访问存储）的生命周期规则不能存在叠加前缀，例如已创建到期删除文件的生命周期规则的前缀是 ABC，则无法再创建前缀为 ABCD 或 AB 或 A 的到期删除文件的生命周期规则。
- 当用户为 Bucket 设置了生命周期规则，这些规则将同时应用于已有文件和后续新创建的文件。例如，用户今天增加了一个生命周期，指定某些前缀的文件 30 天后过期，那么 OOS 将会将满足条件的 30 天前创建的文件都加入到待删除队列中。

OOS 通过将文件的最后一次修改时间或最后一次访问时间，加上生命周期时间来计算到期时间，并且将时间近似到下一天的 GMT 零点时间。例如基于创建时间设置生命周期：一个文

件于 GMT 2016 年 1 月 15 日 10:30 创建，生命周期为 3 天，那么文件的到期时间是 GMT 2016 年 1 月 19 日 00:00。当重写一个文件时，OOS 将以最后更新时间为准，来重新计算该文件的到期时间。

可以通过 GET Object、HEAD Object 查询文件的到期时间。

- **请求语法**

```
PUT /?lifecycle HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Authorization: SignatureValue
Content-MD5: MD5

<LifecycleConfiguration>
    <Rule>
        <ID>UniqueIdentifier</ID>
        <Prefix>Prefix</Prefix>
        <Status>LifecycleStatus</Status>
        <Expiration>
            <Days>NumberOfDays</Days>
        </Expiration>
        <Transition>
            <Days>NumberOfDays</Days>
            <IsAccessTime>IsAccessTime</IsAccessTime>
            <StorageClass>StorageClass</StorageClass>
        </Transition>
    </Rule>
</LifecycleConfiguration>
```

- **请求头**

名称	描述	是否必须
Content-MD5	数据的 base64 编码的 128 位 MD5。此请求头必填，以便校验数据的完整性。	是

- **请求参数**

名称	描述	是否必须

LifecycleConfiguration	生命周期规则容器，最多包含 1000 个规则。 类型：容器 子节点：Rule	是
Rule	配置一条生命周期规则的容器。 类型：容器 子节点：ID、Prefix、Status、Expiration、Transition 父节点：LifecycleConfiguration	是
ID	规则的唯一标识。 类型：字符串 取值：1~255 个字符。说明：如果 ID 不填写，系统会随机生成一个 48 字节长度的 ID。 父节点：Rule	否
Prefix	指定使用生命周期规则的文件前缀。只有匹配文件前缀的文件才能被该规则影响，且对于同一 Bucket，同一类型的生命周期规则的前缀不可重叠。如果 prefix 为空，则默认生命周期规则匹配整个存储桶。 类型：字符串 取值：0~1024 个字符。 父节点：Rule	是
Status	指定生命周期规则的状态。 类型：字符串 取值： ● Enabled：生命周期规则生效。 ● Disabled：生命周期规则不生效，OOS 执行时会忽略该条规则。 父节点：Rule	是
Expiration	指定生命周期规则的过期时间容器。 类型：容器	Expiration 和

	父节点：Rule 子节点：Days 或 Date	Transition 至少包含 1 个
Transition	指定生命周期规则的转换存储类型。 类型：容器 父节点：Rule 子节点：Days 或 Date、StorageClass	Expiration 和 Transition 至少包含一个
Days	指定生命周期规则在匹配文件最后一次修改时间或最后一次访问时间多少天后生效。 **说明**：当请求中存在 IsAccessTime 参数，且 IsAccessTime 取值为 true 时，此参数表示生命周期规则在距离 Object 最后一次访问多少天后生效。 **注意**：如果过期时间和转换时间同时配置，过期时间需要晚于转换时间。 类型：整型 取值：大于 0 的正整数。 父节点：Expiration 或 Transition。	Days 和 Date 二选一
IsAccessTime	指定是否基于最后一次访问时间匹配规则。 **注意**：只有配置 Days 时才生效。 类型：布尔 取值： ● true：生命周期规则匹配文件的最后一次访问时间。当基于最后一次访问时间匹配规则时，文件只支持从标准存储转储为低频访问存储。 ● false：生命周期规则匹配文件的最后一次修改时间。默认值为 false。 父节点：Transition。	否

Date	指定生命周期规则生效日期，OOS 对最后一次修改时间在此日期之前的文件执行生命周期规则。 取值：日期格式必须为 ISO8601 格式，并且为 UTC 的零点。例如：2002-10-11T00:00:00.000Z。 类型：字符串 父节点：Expiration 或 Transition	Days 和 Date 二选一
StorageClass	指定文件转换的存储类型。 类型：字符串 取值：STANDARD_IA：低频访问存储。 父节点：Transition	Transition 已设置时，必填。

- **请求示例 1**

下面的生命周期只包含一个规则，这个规则指定所有以 logs/为前缀的文件将在创建后的 30 天过期。因为生命周期规则的状态是 Enabled，OOS 会每隔一段时间来评估这个规则，删除过期的文件。

```
PUT /?lifecycle HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
Date: Tue, 13 Dec 2011 17:54:50 GMT
Content-MD5: 8dYiLewFWZyGgV2Q5FNI4W==
Authorization: SignatureValue
Content-Length: 207

<LifecycleConfiguration>
    <Rule>
        <ID>delete-logs-in-30-days-rule</ID>
        <Prefix>logs</Prefix>
        <Status>Enabled</Status>
        <Expiration>
            <Days>30</Days>
        </Expiration>
    </Rule>
</LifecycleConfiguration>
```

- **响应示例 1**

```
HTTP/1.1 200 OK
x-amz-request-id: 8911108cf1b04a9234a79aa99eabadb3727478686a6c6e7072
Date: Tue, 13 Dec 2011 19:14:41 GMT
Server: CTYUN
```

- **请求示例 2**

指定存储文件在创建 10 天后转换为低频访问存储，创建 20 天后删除。

```
PUT /?lifecycle HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
Date: Mon, 29 Jul 2019 05:08:31 GMT
Content-Type: application/xml
Content-MD5: ouPJihB832ZHNLZNyl2Ccw==
Content-Length: 175
Authorization: SignatureValue

<LifecycleConfiguration>
    <Rule>
        <Prefix>logs</Prefix>
        <Status>Enabled</Status>
        <Transition>
            <Days>10</Days>
            <StorageClass>STANDARD_IA</StorageClass>
        </Transition>
        <Expiration>
            <Days>20</Days>
        </Expiration>
    </Rule>
</LifecycleConfiguration>
```

- **响应示例 2**

```
HTTP/1.1 200 OK
x-amz-request-id: 08979294ca514b044ec1b4c3b8c5c7cd8c8e91828486888a8c
Date: Mon, 29 Jul 2019 05:10:33 GMT
Server: CTYUN
```

- **请求示例 3：多规则**

  该示例生命周期包含 2 个规则，第一个规则指定所有以 logs/为前缀的文件将在创建后的 30 天过期。第二个规则指定所有以 documents/为前缀的文件将在创建后的 365 天过期，但第二个规则的状态是 Disabled，即不生效。

```
PUT /?lifecycle HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
Date: Tue, 13 Dec 2011 17:54:50 GMT
Content-MD5: 8dYiLewFWZyGgV2Q5FNI4W==
Content-Length: 413
Authorization: SignatureValue

<LifecycleConfiguration>
    <Rule>
        <ID>delete-logs-rule</ID>
        <Prefix>logs/</Prefix>
        <Status>Enabled</Status>
        <Expiration>
            <Days>30</Days>
        </Expiration>
    </Rule>
    <Rule>
        <ID>delete-documents-rule</ID>
        <Prefix>documents/</Prefix>
        <Status>Disabled</Status>
        <Expiration>
            <Days>365</Days>
        </Expiration>
    </Rule>
</LifecycleConfiguration>
```

- **响应示例 3**

```
HTTP/1.1 200 OK
x-amz-request-id: 8b08216804aa4f5569e3e5dcdee4a3a5ad999b9d9fa1a3a5a7
Date: Tue, 13 Dec 2011 19:14:41 GMT
```

```
Server: CTYUN
```

- **请求示例 4**

  下面规则指定所有文件将在创建后的 3650 天被删除。

  注意：当指定空的前缀时，规则将对 Bucket 内的所有文件生效。OOS 将会删除满足此规则的所有文件。

```
PUT /?lifecycle HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
Date: Tue, 13 Dec 2011 17:54:50 GMT
Content-MD5: 8dYiLewFWZyGgV2Q5FNI4W==
Content-Length: 226
Authorization: SignatureValue

<LifecycleConfiguration>
    <Rule>
        <ID>10 years all objects expire rule </ID>
        <Prefix></Prefix>
        <Status>Enabled</Status>
        <Expiration>
            <Days>3650</Days>
        </Expiration>
    </Rule>
</LifecycleConfiguration>
```

- **响应示例 4**

```
HTTP/1.1 200 OK
x-amz-request-id: bf925f41e4094d0cfc76786f71773638402c2e30323436383a
Date:Tue, 13 Dec 2011 19:14:41 GMT
Server: CTYUN
```

- **请求示例 5**

  为前缀是 test 的文件设置生命周期规则：文件最后一次访问 3 天后转为低频访问存储。

```
PUT /?lifecycle HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
```

```
x-amz-content-sha256: UNSIGNED-PAYLOAD
x-amz-date: 20230821T092414Z
Content-MD5: NH5mGrd4Be/Q0EWSuXpfYA==
Connection: keep-alive
Content-Length: 324
Authorization: SignatureValue

<LifecycleConfiguration>
    <Rule>
        <ID>test_lifecycle_1</ID>
        <Prefix>test</Prefix>
        <Status>Enabled</Status>
        <Transition>
            <Days>3</Days>
            <IsAccessTime>true</IsAccessTime>
            <StorageClass>STANDARD_IA</StorageClass>
        </Transition>
    </Rule>
</LifecycleConfiguration>
```

- **响应示例 5**

```
HTTP/1.1 200 OK
Date: Mon, 21 Aug 2023 09:24:14 GMT
x-amz-request-id: cbc195a82f344b0b0b574e5f6562654467715355585 35c6849
Content-Length: 0
Server: CTYUN
```

- **请求示例 6**

  在设置多个规则时，请注意各规则之间不要互相冲突。比如，下面的生命周期中配置了一个规则，指定所有以 documents 为前缀的文件在 30 天后过期。而另一个规则指定所有以 documents/2011 为前缀的文件在 365 天过期。在这种情况下，OOS 将返回错误信息。

```
PUT /?lifecycle HTTP/1.1
Host: docs.oos-cn.ctyunapi.cn
Date: Tue, 13 Dec 2011 17:54:50 GMT
```

```
Content-MD5: 8dYiLewFWZyGgV2Q5FNI4W==
Content-Length: 226
Authorization: SignatureValue

<LifecycleConfiguration>
    <Rule>
        <ID>111</ID>
        <Prefix>documents</Prefix>
        <Status>Enabled</Status>
        <Expiration>
            <Days>30</Days>
        </Expiration>
    </Rule>
    <Rule>
        <ID>222</ID>
        <Prefix>documents/2011</Prefix>
        <Status>Enabled</Status>
        <Expiration>
            <Days>365</Days>
        </Expiration>
    </Rule>
</LifecycleConfiguration>
```

4.3.17 GET Bucket Lifecycle

此接口用于返回配置的 Bucket 生命周期。

- **请求语法**

```
GET /?lifecycle HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Authorization: SignatureValue
```

- **响应结果**

名称	描述
LifecycleConfiguration	生命周期规则容器。 类型：容器 子节点：Rule
Rule	生命周期规则的容器。 类型：容器 父节点：LifecycleConfiguration
ID	规则的唯一标识。 类型：字符串 父节点：Rule
Prefix	使用规则的文件前缀。 类型：字符串 父节点：Rule
Status	生命周期规则的状态： ● Enabled：生命周期规则生效。 ● Disabled：生命周期规则不生效。 类型：字符串 父节点：Rule
Expiration	描述过期动作的容器。

	类型：容器 父节点：Rule 子节点：Days 或 Date
Transition	生命周期规则的转换存储类型。 类型：容器 父节点：Rule 子节点：StorageClass、Days 或 Date
StorageClass	文件转换的存储类型：STANDARD_IA：低频访问存储。 类型：字符串 父节点：Transition
Days	● 当 IsAccessTime 为 false 时，表示生命周期规则在匹配文件最后一次修改多少天后生效。 ● 当 IsAccessTime 为 true 时，表示生命周期规则在匹配文件最后一次访问时间多少天后生效。 类型：整型 父节点：Expiration 或 Transition
IsAccessTime	是否基于最后一次访问时间匹配规则： ● true：生命周期规则匹配文件的最后一次访问时间。 ● false：生命周期规则匹配文件的最后一次修改时间。 类型：布尔型 父节点：Transition
Date	生命周期规则生效日期，OOS 对在此日期之前创建的文件执行生命周期规则。 类型：字符串 父节点：Expiration 或 Transition
AtimeBase	当生命周期规则基于最后一次访问时间匹配时，返回示例中包含 AtimeBase 元素，表示默认最后一次访问的时间戳（从 1970-01-01

	00:00:00 UTC 计算起的秒数），即为 Bucket 开启访问跟踪时间点的时间戳。 类型：时间戳 父节点：Rule

- **请求示例 1**

下面的例子显示所有以 logs 为前缀的文件将在最后一次修改时间的 30 天后到期删除。

```
GET /?lifecycle HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
Date: Tue, 13 Dec 2011 17:54:50 GMT
Authorization: SignatureValue
```

- **响应示例 1**

```
HTTP/1.1 200 OK
x-amz-request-id: b13502b728c8468a2da7a9a0a2a867696f5d5f61636567696b
Date:Tue, 13 Dec 2011 19:14:41 GMT
Content-Length: 267
Server: CTYUN

 <?xml version="1.0" encoding="UTF-8"?>
 <LifecycleConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <Rule>
       <ID>30-day-log-deletion-rule</ID>
       <Prefix>logs</Prefix>
       <Status>Enabled</Status>
       <Expiration>
           <Days>30</Days>
       </Expiration>
   </Rule>
</LifecycleConfiguration>
```

- **请求示例 2**

下面的例子显示所有以 test 为前缀的文件将在最后一次访问时间的 3 天后转为低频访问存储。

```
GET /testbucket1?lifecycle HTTP/1.1
```

```
x-amz-content-sha256: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date: 20230821T092414Z
Host: example-bucket.oos-cn.ctyunapi.cn
Connection: keep-alive
Authorization: SignatureValue
```

- **响应示例 2**

```
HTTP/1.1 200 OK
x-amz-request-id: 92bc7f34cf824ffb057f81787a803f41493537393b3d3f4143
Date:Tue, 13 Dec 2011 19:14:41 GMT
Content-Length: 267
Server: CTYUN

<?xml version="1.0" encoding="UTF-8"?>
<LifecycleConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Rule>
    <ID>test_lifecycle_1</ID>
    <Prefix>test</Prefix>
    <Status>Enabled</Status>
    <Transition>
        <Days>3</Days>
        <IsAccessTime>true</IsAccessTime>
        <StorageClass>STANDARD_IA</StorageClass>
    </Transition>
    <AtimeBase>1692608940364</AtimeBase>
   </Rule>
</LifecycleConfiguration>
```

4.3.18 DELETE Bucket Lifecycle

此操作用于删除配置的 Bucket 生命周期，OOS 将会删除指定 Bucket 的所有生命周期配置规则。用户的文件将永远不会到期，OOS 也不会再自动删除文件。只有根用户和拥有 DELETE Bucket Lifecycle 权限的子用户才能执行此操作。

- **请求语法**

```
DELETE /?lifecycle HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Authorization: SignatureValue
```

- **请求示例**

```
DELETE /?lifecycle HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
Date: Wed, 14 Dec 2011 05:37:16 GMT
Authorization: SignatureValue
```

- **响应示例**

```
HTTP/1.1 204 No Content
x-amz-request-id: 50976829c9e84b400d87898082884749503d3f41434547494b
Date: Wed, 14 Dec 2011 05:37:16 GMT
Server: CTYUN
```

4.3.19 PUT Bucket CORS

此操作用来设置 Bucket 的跨域资源共享(Cross-Origin Resource Sharing，CORS)。浏览器限制脚本内发起跨源 HTTP 请求，即同源策略。例如，当来自于 A 网站的页面中的 JavaScript 代码希望访问 B 网站的时候，浏览器会拒绝该访问，因为 A、B 两个网站是属于不同的域。通过配置 CORS，可以解决不同域相互访问的问题，CORS 定义了客户端 Web 应用程序在一个域中与另一个域中的资源进行交互的方式。

以下是有关使用 CORS 的示例场景：

- 场景 1：比如用户的网站 www.exam***ple.com，后端使用了 OOS。在 web 应用中提供了使用 JavaScript 实现的上传文件功能，但是在该 web 应用中，只能向 www.exam***ple.com 发送请求，向其他网站发送的请求都会被浏览器拒绝。这样就导致用户上传的数据必须从 www.exam***ple.com 中转。如果设置了跨域访问的话，用户就可以直接上传到 OOS，而无需从 www.exam***ple.com 中转。
- 场景 2：假设用户在名为 example-bucket 的 Bucket 中托管网站，网站的 Endpoint 是 https://example-bucket.oos-website-cn.oos-xx.ctyunapi.cn。现在，用户想要使用网页上的 JavaScript (存储在此 Bucket 中)，通过 OOS API endpoint oos-xx.ctyunapi.cn 向 bucket 发送 GET 和 PUT 请求。浏览器通常会阻止 JavaScript 发送这些请求，但借助 CORS，用户可以配置 Bucket 支持来自 example-bucket.oos-website-cn.oos-xx.ctyunapi.cn 的跨域请求。

设置 Bucket 的跨域请求。如果配置已经存在，OOS 会覆盖它。只有根用户和拥有 PUT Bucket CORS 权限的子用户才能执行此操作，否则会返回 403 AccessDenied 错误。在配置跨域请求时，用户可以通过 XML 来配置允许跨域的源和 HTTP 方法。XML 请求体不能超过 64KiB。

OOS 收到来自浏览器的预检请求后，它将为 Bucket 评估 CORS 配置，并使用第一个与浏览器请求相匹配的 CORSRule 规则来实现跨域请求。要使规则实现匹配，必须满足以下条件：

- 请求的 Origin 标头必须匹配一个 AllowedOrigin 元素。
- 请求方法(例如，GET 或 PUT)，或者预检 OPTIONS 请求中的 Access-Control-Request-Method 请求头，必须是某个 AllowedMethod 元素。
- 在预检请求中， Access-Control-Request-Headers 请求头中列出的每个请求头，必须匹配一个 AllowedHeader 元素。

- **请求语法**

```
PUT /?cors HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Content-Length: length
Date: date
Content-MD5: MD5
Authorization: SignatureValue

<CORSConfiguration>
    <CORSRule>
        <AllowedOrigin>Origin you want to allow cross-domain requests
from</AllowedOrigin>
        <AllowedOrigin>...</AllowedOrigin>
        ...
        <AllowedMethod>HTTP method</AllowedMethod>
        <AllowedMethod>...</AllowedMethod>
        ...
        <MaxAgeSeconds>Time in seconds your browser to cache the pre-flight OPTIONS
response for a resource</MaxAgeSeconds>
        <AllowedHeader>Headers that you want the browser to be allowed to
send</AllowedHeader>
        <AllowedHeader>...</AllowedHeader>
        ...
        <ExposeHeader>Headers in the response that you want accessible from client
application</ExposeHeader>
        <ExposeHeader>...</ExposeHeader>
        ...
    </CORSRule>
    <CORSRule>
        ...
    </CORSRule>
</CORSConfiguration>
```

- **请求参数**

名称	描述	是否必须
CORSConfiguration	最多包含100个CORSRule元素的容器。 类型：容器 子节点：CORSRule	是

CORSRule	用户允许跨域的源和方法。 类型：容器 子节点：AllowedOrigin、AllowedMethod、AllowedHeader MaxAgeSeconds、ExposeHeader、ID 父节点：CORSConfiguration	是
ID	规则的唯一标识。最长255个字符。 类型：字符串 父节点：CORSRule	否
AllowedMethod	允许跨域的HTTP方法。每个CORSRule应至少包含一个源和一个方法。 类型: 枚举 (GET, PUT, HEAD, POST, DELETE) 父节点：CORSRule	是
AllowedOrigin	允许跨域的源。每个CORSRule应至少包含一个源和一个方法。 可以包含通配符*，但最多只能包含一个，比如：http://*.ctyun.cn用户也可以只指定 * 表示允许所有源跨域访问。 类型：字符串 父节点：CORSRule	是
AllowedHeader	控制在预检 OPTIONS 请求中 Access-Control-Request-Headers 头中指定的 header 是否允许。Access-Control-Request-Headers 中的每个请求头名称，必须在规则中有匹配的条目。 规则中的每个 AllowedHeader 最多可以包含一个 * 通配符字符。例如，<AllowedHeader>x-amz-*</AllowedHeader> 类型：字符串	否

	父节点：CORSRule	
MaxAgeSeconds	指定浏览器对特定资源的预检（OPTIONS）请求返回结果的缓存时间，单位为秒。通过缓存响应，在需要重复原始请求时，浏览器无需向 OOS 发送预检请求。 一个CORSRule最多包含一个MaxAgeSeconds元素。 类型：整型 取值：大于等于-1的整数。-1表示禁用缓存。 父节点：CORSRule	否
ExposeHeader	指定客户应用程序(例如，JavaScript XMLHttpRequest文件)能够访问的响应头。 类型：字符串 父节点: CORSRule	否

- **请求示例**

  第一个规则允许来自 https://docs.oos-cn.ctyunapi.cn 源的跨源 PUT、POST 和 DELETE 请求。该规则还通过 Access-Control-Request-Headers 标头允许预检 OPTIONS 请求中的所有标头。作为对任何预检 OPTIONS 请求的响应，OOS 将返回请求的任意请求头。

  第二个规则允许与第一个规则具有相同的跨源请求，但第二个规则应用于另一个源 https://example.bucket.oos-cn.ctyunapi.cn。

  第三个规则允许来自所有源的跨源 GET 请求。“*”通配符字符是指所有的源。

```
PUT /?cors HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
x-amz-date: Tue, 21 Aug 2012 17:54:50 GMT
Content-MD5: 8dYiLewFWZyGgV2Q5FNI4W==
Authorization: SignatureValue
Content-Length: 445

<CORSConfiguration>
 <CORSRule>
   <AllowedOrigin>https://docs.oos-cn.ctyunapi.cn</AllowedOrigin>

   <AllowedMethod>PUT</AllowedMethod>
```

```
    <AllowedMethod>POST</AllowedMethod>
    <AllowedMethod>DELETE</AllowedMethod>

    <AllowedHeader>*</AllowedHeader>
  </CORSRule>
  <CORSRule>
    <AllowedOrigin>https://example.bucket.oos-cn.ctyunapi.cn</AllowedOrigin>

    <AllowedMethod>PUT</AllowedMethod>
    <AllowedMethod>POST</AllowedMethod>
    <AllowedMethod>DELETE</AllowedMethod>

    <AllowedHeader>*</AllowedHeader>
  </CORSRule>
  <CORSRule>
    <AllowedOrigin>*</AllowedOrigin>
    <AllowedMethod>GET</AllowedMethod>
  </CORSRule>
</CORSConfiguration>
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id: 8859db542a32455738b2b4abadb3727479686a6c6e70727476
Date: Tue, 21 Aug 2012 17:54:50 GMT
Server:CTYUN
```

4.3.20 GET Bucket CORS

返回 Bucket 的跨域配置信息。只有根用户和拥有 GET Bucket CORS 权限的子用户才能执行此操作，否则会返回 403 AccessDenied 错误。

- **请求语法**

```
GET /?cors HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Authorization: SignatureValue
```

- **响应结果**

名称	描述
CORSConfiguration	最多包含100个CORSRule元素的容器 类型：容器
CORSRule	用户允许跨域的源和方法。 类型：容器 子节点：AllowedOrigin、AllowedMethod、MaxAgeSeconds、ExposeHeader、ID 父节点：CORSConfiguration
ID	规则的唯一标示。最长255个字符。 类型：字符串 父节点：CORSRule
AllowedMethod	用户允许跨域的HTTP方法。每个CORSRule应至少包含一个源和一个方法。 类型：枚举 取值：GET、PUT、HEAD、POST、DELETE。 父节点：CORSRule
AllowedOrigin	用户允许跨域的源。最多包含一个 * 通配符。比如：http://*.ctyun.cn。 用户也可以只指定 * 表示允许所有源跨域访问。

	类型：字符串 父节点：CORSRule
AllowedHeader	通过 Access-Control-Request-Headers 请求头，指定预检OPTIONS请求中允许的请求头。Access-Control-Request-Headers 中的每个请求头名称，必须在规则中有匹配的相应条目。OOS将仅发送允许的响应头。规则中的每个 AllowedHeader 字符串可以最多包含一个 * 通配符字符。例如，<AllowedHeader>x-amz-*</AllowedHeader> 类型：字符串 父节点：CORSRule
MaxAgeSeconds	指定浏览器为预检请求缓存响应的时间 (以秒为单位)。 一个CORSRule最多包含一个MaxAgeSeconds元素。 类型：整型 父节点：CORSRule
ExposeHeader	指定客户应用程序 (例如，JavaScript XMLHttpRequest文件) 能够访问的响应头。 类型：字符串 父节点：CORSRule

- **请求示例**

```
GET /?cors HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
Date: Wed, 13 Dec 2017 19:14:42 GMT
Authorization: SignatureValue
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id: 2c3ee2045e7f4839c03a3c33353bfafc04f0f2f4f6f8fafcfe
Date: Wed, 13 Dec 2017 19:14:42 GMT
Server: CTYUN
Content-Length: 280

<CORSConfiguration>
```

```
    <CORSRule>
        <AllowedOrigin>https://docs.oos-cn.ctyunapi.cn</AllowedOrigin>
        <AllowedMethod>GET</AllowedMethod>
        <MaxAgeSeconds>3000</MaxAgeSeconds>
        <ExposeHeader>x-amz-server-side-encryption</ExposeHeader>
    </CORSRule>
</CORSConfiguration>
```

4.3.21 DELETE Bucket CORS

删除 Bucket 的跨域配置信息。只有根用户和拥有 DELETE Bucket CORS 权限的子用户才能执行此操作，否则会返回 403 AccessDenied 错误。

- 如果 Bucket 配置了 CORS，删除成功，返回 200 OK。
- 如果 Bucket 没有配置 CORS，返回 204 NoContent。

- **请求语法**

```
DELETE /?cors HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Authorization: SignatureValue
```

- **请求示例**

```
DELETE /?cors HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
Date: Tue, 13 Dec 2011 19:14:42 GMT
Authorization: SignatureValue
```

- **响应示例**

```
HTTP/1.1 204 No Content
x-amz-request-id: a22e2ec7cee74e8874eef0e7e9efaeb0b3a4a6a8aaacaeb0b2
Date: Tue, 13 Dec 2011 19:14:42 GMT
Server: CTYUN
```

4.3.22 PUT Bucket Object Lock

使用此操作可以开启合规保留功能，开启后将对 Bucket 中所有文件生效。只有根用户和有权限的子用户才可以进行此操作，匿名用户不能进行此操作。

开启 Bucket 合规保留功能后，任何用户（包括根用户）都不能对此 Bucket 内处于合规保留期的文件进行修改和删除。

可以重复调用此接口：

- 如果已经开启合规保留策略：设置合规保留时长大于或等于上次设置的时长，才能生效。如果使用 Years 和 Days 两种方式设置合规保留时长，年与天的换算关系为：1 年等于 365 天。
- 如果未开启合规保留策略：设置合规保留时长可以大于、等于或小于上次设置的时长。

注意：

- 合规保留一旦开启，不能关闭，不能缩短合规保留时长，但可以延长合规保留时长。
- 合规保留的时间精确到秒，例如对 Bucket A 设置合规保留时长为 10 天，文件 A 属于 Bucket A，A1 的最后更新时间为 2019-3-1 12:00:00，该文件会在 2019-3-11 12:00:01 过合规保留期。
- 任何用户（包括根用户）都不能修改、覆盖、删除处于合规保留期的文件。
- 处于合规保留期的文件，无法通过调用 API、控制台修改文件的存储类型，只能通过生命周期修改存储类型。
- 处于合规保留期的文件，如果设置了生命周期规则，则修改存储类型的生命周期规则可以生效，设置删除操作的生命周期规则待文件过了合规保留期后才能生效。

- **请求语法**

```
PUT /?object-lock HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Content-MD5:md5
Content-Length:Length
Authorization: SignatureValue

<ObjectLockConfiguration>
    <ObjectLockEnabled>Enabled</ObjectLockEnabled>
```

```
    <Rule>
        <DefaultRetention>
            <Mode>COMPLIANCE</Mode>
            <Days>days</Days>
            <Years>years</Years>
        </DefaultRetention>
    </Rule>
</ObjectLockConfiguration>
```

- **请求头**

名称	描述	是否必须
Content-MD5	数据的 base64 编码的 128 位 MD5。此请求头必填，以便校验数据的完整性。	是

- **请求参数**

名称	描述	是否必须
ObjectLockConfiguration	合规保留配置信息的容器。 类型：容器 子节点：ObjectLockEnabled	是
ObjectLockEnabled	Bucket 是否开启合规保留功能。 类型：枚举 取值： ● Enabled：开启合规保留。 ● Disabled：不开启合规保留。 父节点：ObjectLockConfiguration	是
Rule	设置合规保留规则。 类型：容器 父节点：ObjectLockConfiguration 子节点：DefaultRetention	否
DefaultRetention	默认的合规保留配置。	是

	类型：容器 父节点：Rule 子节点：Mode、Days 或 Years 二选一	
Mode	合规保留模式。 类型：枚举 取值：COMPLIANCE：合规保留。 父节点：DefaultRetention	是
Days	合规保留的天数。 类型：整型 取值：整数形式，1~36500。 **说明**：年与天的换算关系：1 年等于 365 天。 父节点：DefaultRetention	条件 Days 和 Years 必须 二选一
Years	合规保留的年数。 类型：整型 取值：整数形式，1~100。 **说明**：年与天的换算关系：1 年等于 365 天。 父节点：DefaultRetention	条件 Days 和 Years 必须 二选一

- **请求示例**

```
PUT /?object-lock HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
Date: Tue, 12 May 2020 06:18:52 GMT
Content-Type: application/xml; charset=utf-8
Content-Length: 232
Content-MD5: m08Xh8w1oh6bZrLA6Hseaw==
Authorization: SignatureValue

<ObjectLockConfiguration>
    <ObjectLockEnabled>Enabled</ObjectLockEnabled>
    <Rule>
        <DefaultRetention>
            <Mode>COMPLIANCE</Mode>
            <Days>1</Days>
```

```
        </DefaultRetention>
    </Rule>
</ObjectLockConfiguration>
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id: a7ed9cfd3fc743c035afb1a8aab06f71776567696b6d6f7173
Date: Tue, 12 May 2020 03:24:38 GMT
Server: CTYUN
```

4.3.23 GET Bucket Object Lock

使用此操作可以获取 Bucket 合规保留的配置信息。只有根用户和有权限的子用户才可以进行此操作。

- **请求语法**

```
GET /?object-lock HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Authorization: SignatureValue
```

- **响应结果**

名称	描述
ObjectLockConfiguration	合规保留配置信息的容器。 类型：容器 子节点：ObjectLockEnabled、Rule
ObjectLockEnabled	Bucket 是否开启合规保留功能： ● Enabled：开启合规保留。 ● Disabled：不开启合规保留。 类型：枚举 父节点：ObjectLockConfiguration
Rule	合规保留的规则。 类型：容器 父节点：ObjectLockConfiguration 子节点：DefaultRetention
DefaultRetention	默认的合规保留配置。 类型：容器 父节点：Rule 子节点：Mode、Days 或 Years 二选一
Mode	合规保留模式。COMPLIANCE：合规保留。 类型：枚举

	父节点：DefaultRetention
Days	合规保留的天数。 类型：整型 父节点：DefaultRetention
Years	合规保留的年数。 类型：整型 父节点：DefaultRetention

- **请求示例**

```
GET /?object-lock HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
Date: 20200514T020802Z
Authorization: SignatureValue
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id: bda73fc9ada1433b54ced0c7c9cf8e90948486888a8c8e9092
Date: Thu, 14 May 2020 02:08:15 GMT
Content-Length:50
Server: CTYUN

<?xml version="1.0" encoding="UTF-8"?>
<ObjectLockConfiguration>
    <ObjectLockEnabled>Enabled</ObjectLockEnabled>
    <Rule>
        <DefaultRetention>
            <Mode>COMPLIANCE</Mode>
            <Days>1</Days>
        </DefaultRetention>
    </Rule>
</ObjectLockConfiguration>
```

4.3.24 DELETE Bucket Object Lock

使用此操作可以删除未启用的合规保留配置信息。只有根用户和有权限的子用户才可以进行此操作。

- **请求语法**

```
DELETE /?object-lock HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Authorization: SignatureValue
```

- **请求示例**

```
DELETE /?object-lock HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
X-Amz-Date: 20200514T020802Z
Authorization: SignatureValue
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id: ebfd507075c243b8057f81787a803f41483537393b3d3f4143
Date: Thu, 14 May 2020 02:08:15 GMT
Server: CTYUN
```

4.3.25 PUT Bucket Inventory Configuration

此操作用来为 Bucket 配置 Bucket 清单规则。

通过 OOS Bucket 清单功能可以获取 Bucket 中指定文件（Object）的大小、存储类型等信息。相对于 GET Bucket (List Objects)接口，Bucket 清单结果文件可以按天或者按周以 CSV 的形式输出指定文件的相关信息，且不会影响 Bucket 的请求速率。在需要列举海量 Object 的场景中，推荐使用 Bucket 清单功能。

说明：每个 Bucket 最多可以配置 10 条 Bucket 清单规则。

- **请求语法**

```
PUT /?inventory&id=id HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Content-MD5:md5
Content-Length:length
Authorization: SignatureValue

<?xml version="1.0" encoding="UTF-8"?>
<InventoryConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <Destination>
      <OOSBucketDestination>
         <Bucket>BucketName</Bucket>
         <Format>format</Format>
         <Prefix>prefix</Prefix>
      </OOSBucketDestination>
   </Destination>
   <IsEnabled>IsEnabled</IsEnabled>
   <Filter>
      <Prefix>prefix</Prefix>
   </Filter>
   <Id>Id</Id>
   <OptionalFields>
      <Field>field</Field>
      ……
      <Field>field</Field>
   </OptionalFields>
   <Schedule>
      <Frequency>frequency</Frequency>
   </Schedule>
```

```
</InventoryConfiguration>
```

- **请求头**

名称	描述	是否必须
id	指定的清单名称，清单名称在当前 Bucket 下必须唯一。 类型：字符串 取值：只允许使用小写字母、数字、短横线（-）和下划线（_），且不能以短横线（-）或下划线（_）开头或结尾，1~64 个字符。	是

- **请求参数**

名称	描述	是否必须
InventoryConfiguration	清单配置的容器。 类型：容器 子节点：Destination、IsEnabled、Filter、Id、OptionalFields、Schedule	是
Destination	存放清单结果的容器。 类型：容器 父节点：InventoryConfiguration 子节点：OOSBucketDestination	是
OOSBucketDestination	存放清单结果的 Bucket 信息。 类型：容器 父节点：Destination 子节点：Bucket、Format、Prefix	是
Bucket	指定存放清单结果文件的 Bucket。 类型：字符串	是

	取值：格式为 arn:ctyun:oos:::*bucketname*， *bucketname* 为已经创建的 Bucket 名字。 父节点：OOSBucketDestination	
Format	指定清单结果文件的类型。 类型：字符串 取值：CSV 父节点：OOSBucketDestination	是
Prefix	指定清单结果文件的存储路径前缀。 类型：字符串 取值：0~512 个字符。 父节点：OOSBucketDestination	否
IsEnabled	指定清单功能是否启用。 类型：布尔型 取值： ● true：启用清单功能。 ● false：不启用清单功能。 父节点：InventoryConfiguration	是
Filter	清单筛选的前缀。 **说明**：指定前缀后，清单将筛选出符合前缀设置的文件。 类型：容器 父节点：InventoryConfiguration 子节点：Prefix	否
Prefix	指定筛选规则的匹配前缀。 类型：字符串 取值：0~1024 个字符。 **说明**：如果未指定筛选规则的匹配前缀，则匹配源 Bucket 中的所有文件。如果源 Bucket 没有任何文	否

	件，或清单设置的前缀没有匹配到任何文件，则会生成相应文件和文件夹，但清单结果文件中无内容。 父节点：Filter	
Id	指定的清单名称，清单名称在当前 Bucket 下必须全局唯一。 **注意**：必须与请求头中的 Id 相同。 类型：字符串 取值：只允许使用小写字母、数字、短横线（-）和下划线（_），且不能以短横线（-）或下划线（_）开头或结尾，1~64 个字符。 父节点：InventoryConfiguration	是
OptionalFields	清单结果配置项的容器。 类型：容器 父节点：InventoryConfiguration 子节点：Field	否
Field	设置清单结果中包含配置项，可以设置多个配置项。 类型：字符串 取值： ● Size：Object 的大小。 ● LastModifiedDate：Object 最后一次修改时间。 ● ETag：Object 的 ETag 值，用于标识 Object 的内容。 ● StorageClass：Object 的存储类型。 ● IsMultipartUploaded：是否为通过分片上传方式上传的 Object。	否

	说明：如果未设置配置项，清单默认输出源 Bucket 和 Key（文件名称）。 父节点：OptionalFields	
Schedule	存放清单结果导出周期的容器。 类型：容器 父节点：InventoryConfiguration 子节点：Frequency	是
Frequency	指定清单结果文件导出的周期。 类型：字符串 取值： ● Daily：按天导出清单结果文件。 ● Weekly：按周导出清单文件。清单规则开启当天会根据清单规则启动一次清单导出任务，第二天启动周期性清单导出任务。例如周一开启清单规则，周一当天会启动清单导出任务，后期会按每周二启动清单导出任务。 **说明**：当前的清单结果文件导出完成后，才会创建新的清单任务。如果 Object 较多时（数量大于 10 亿），建议按周导出清单结果文件。 父节点：Schedule	是

- **请求示例**

```
PUT /?inventory&id=1 HTTP/1.1
Date: Wed, 23 Aug 2023 07:14:20 GMT
Host: example-bucket.oos-cn.ctyunapi.cn
Connection: keep-alive
Content-Length: 466
Authorization: SignatureValue

<?xml version="1.0" encoding="UTF-8"?>
```

```
<InventoryConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Destination>
    <OOSBucketDestination>
      <Bucket>arn:ctyun:oos:::docs</Bucket>
      <Format>CSV</Format>
      <Prefix>Inventory-</Prefix>
    </OOSBucketDestination>
  </Destination>
  <Filter>
    <Prefix></Prefix>
  </Filter>
  <Id>1</Id>
  <IsEnabled>true</IsEnabled>
  <OptionalFields>
    <Field>Size</Field>
  </OptionalFields>
  <Schedule>
    <Frequency>Daily</Frequency>
  </Schedule>
</InventoryConfiguration>
```

- **响应示例**

```
HTTP/1.1 200 OK
Date: Wed, 23 Aug 2023 07:14:20 GMT
x-amz-request-id: 16b538e3df744c04e02c23343a373a19353125392e2f48341e
Content-Length: 0
Server: CTYUN
```

4.3.26 GET Bucket Inventory Configuration

此操作用来查看某 Bucket 中指定清单配置。

- **请求语法**

```
GET /?inventory&id=id HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Authorization: SignatureValue
```

- **请求头**

名称	描述	是否必须
id	清单名称。 类型：字符串 取值：只允许使用小写字母、数字、短横线（-）和下划线（_），且不能以短横线（-）或下划线（_）开头或结尾，1~64 个字符。	是

- **响应结果**

名称	描述
InventoryConfiguration	清单配置的容器。 类型：容器 子节点：Destination、IsEnabled、Filter、Id、OptionalFields、Schedule
Destination	存放清单结果的容器。 类型：容器 父节点：InventoryConfiguration 子节点：OOSBucketDestination
OOSBucketDestination	存放清单结果的 Bucket 信息。 类型：容器

	父节点：Destination 子节点：Bucket、Format、Prefix
Bucket	存放清单结果文件的 Bucket。 类型：字符串 父节点：OOSBucketDestination
Format	清单结果文件的类型。 类型：字符串 父节点：OOSBucketDestination
Prefix	清单结果的存储路径前缀。 类型：字符串 父节点：OOSBucketDestination
IsEnabled	清单功能是否启用： ● true：启用清单功能。 ● false：不启用清单功能。 类型：布尔型 父节点：InventoryConfiguration
Filter	清单筛选的前缀。 类型：容器 父节点：InventoryConfiguration 子节点：Prefix
Prefix	筛选规则的匹配前缀。 类型：字符串 父节点：Filter
Id	清单名称。 类型：字符串 父节点：InventoryConfiguration
OptionalFields	清单结果配置项的容器。 类型：容器

	父节点：InventoryConfiguration 子节点：Field
Field	清单结果中包含配置项： ● Size：Object 的大小。 ● LastModifiedDate：Object 最后一次修改时间。 ● ETag：Object 的 ETag 值，用于标识 Object 的内容。 ● StorageClass：Object 的存储类型。 ● IsMultipartUploaded：是否为通过分片上传方式上传的 Object。 **说明**：如果未设置配置项，清单默认输出 Bucket 和 Key（文件名称）。 类型：字符串 父节点：OptionalFields
Schedule	存放清单结果导出周期的容器。 类型：容器 父节点：InventoryConfiguration 子节点：Frequency
Frequency	清单结果文件导出的周期： ● Daily：按天导出清单结果文件。 ● Weekly：按周导出清单结果文件。 类型：字符串 父节点：Schedule

- **请求示例**

```
GET /?inventory&id=1 HTTP/1.1
Date: Wed, 23 Aug 2023 07:09:10 GMT
Host: example-bucket.oos-cn.ctyunapi.cn
Connection: keep-alive
Authorization: SignatureValue
```

- **响应示例**

```
HTTP/1.1 200 OK
Content-Type: application/xml;charset=UTF-8
Date: Wed, 23 Aug 2023 07:09:10 GMT
x-amz-request-id: 684a29525fcc40123ebab1c2c8c5c8a7c3bfb3c7bcbdd6c1ac
Content-Length: 466
Server: CTYUN

<?xml version="1.0" encoding="UTF-8"?>
<InventoryConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Destination>
    <OOSBucketDestination>
      <Bucket>arn:ctyun:oos:::docs</Bucket>
      <Format>CSV</Format>
      <Prefix>Inventory-</Prefix>
    </OOSBucketDestination>
  </Destination>
  <Filter>
    <Prefix></Prefix>
  </Filter>
  <Id>1</Id>
  <IsEnabled>true</IsEnabled>
  <OptionalFields>
    <Field>Size</Field>
  </OptionalFields>
  <Schedule>
    <Frequency>Daily</Frequency>
  </Schedule>
</InventoryConfiguration>
```

4.3.27 List Bucket Inventory Configuration

此操作用来查看某 Bucket 的所有 Bucket 清单的配置。

- **请求语法**

不带 continuation-token 参数

```
GET /?inventory HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Authorization: SignatureValue
```

带 continuation-token 参数

```
GET /?inventory&continuation-token=continuation-token HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Authorization: SignatureValue
```

- **请求头**

名称	描述	是否必须
continuation-token	指定 List 操作需要从此 token 开始，前一次响应中返回的 NextContinuationToken 值。 若上一次请求无法返回全部清单配置项，则会返回 NextContinuationToken，后一次请求需要输入返回的 NextContinuationToken 参数作为 continuation-token 的值。	否

- **响应结果**

名称	描述
ListInventoryConfigurationsResult	整个响应的容器。 类型：容器 子节点：ContinuationToken、InventoryConfiguration、IsTruncated、NextContinuationToken

ContinuationToken	本次请求的 continuation-token 参数。 类型：字符串 父节点：ListInventoryConfigurationsResult
InventoryConfiguration	清单配置的容器。 类型：容器 父节点：ListInventoryConfigurationsResult 子节点：Destination、IsEnabled、Filter、Id、OptionalFields、Schedule
Destination	存放清单结果的容器。 类型：容器 父节点：InventoryConfiguration 子节点：OOSBucketDestination
OOSBucketDestination	存放清单结果的 Bucket 信息。 类型：容器 父节点：Destination 子节点：Bucket、Format、Prefix
Bucket	存放清单结果文件的 Bucket。 类型：字符串 父节点：OOSBucketDestination
Format	清单结果文件的类型。 类型：字符串 父节点：OOSBucketDestination
Prefix	清单结果的存储路径前缀。 类型：字符串 父节点：OOSBucketDestination
IsEnabled	清单功能是否启用： ● true：启用清单功能。 ● false：不启用清单功能。

	类型：布尔型 父节点：InventoryConfiguration
Filter	清单筛选的前缀。 类型：容器 父节点：InventoryConfiguration 子节点：Prefix
Prefix	筛选规则的匹配前缀。 类型：字符串 父节点：Filter
Id	清单名称。 类型：字符串 父节点：InventoryConfiguration
OptionalFields	清单结果配置项的容器。 类型：容器 父节点：InventoryConfiguration 子节点：Field
Field	清单结果中包含配置项： ● Size：Object 的大小。 ● LastModifiedDate：Object 最后一次修改时间。 ● ETag：Object 的 ETag 值，用于标识 Object 的内容。 ● StorageClass：Object 的存储类型。 ● IsMultipartUploaded：是否为通过分片上传方式上传的 Object。 **说明**：如果未设置配置项，清单默认输出源 Bucket 和 Key（文件名称）。 类型：字符串 父节点：OptionalFields

Schedule	存放清单结果导出周期的容器。 类型：容器 父节点：InventoryConfiguration 子节点：Frequency
Frequency	清单结果文件导出的周期： ● Daily：按天导出清单结果文件。 ● Weekly：按周导出清单结果文件。 类型：字符串 父节点：Schedule
IsTruncated	是否还有未列举的清单： ● true：表示还有未列举的清单。您可以将 NextContinuationToken 字段的值作为下一次 list 请求的 continuation-token 参数，以获取下一页的清单配置列表。 ● false：表示本次已列举完成所有清单。 类型：布尔型 父节点：ListInventoryConfigurationsResult
NextContinuationToken	当响应中的 IsTruncated 为 true，且 NextContinuationToken 非空时，使用该字段作为下一次 list 请求的 continuation-token 参数。 类型：字符串 父节点：ListInventoryConfigurationsResult

- **请求示例**

```
GET /?inventory HTTP/1.1
Date: Thu, 24 Aug 2023 02:16:10 GMT
Host: example-bucket.oos-cn.ctyunapi.cn
Connection: keep-alive
Authorization: SignatureValue
```

- **响应示例**

```
HTTP/1.1 200 OK
Content-Type: application/xml;charset=UTF-8
Date: Thu, 24 Aug 2023 02:16:10 GMT
x-amz-request-id: 3332f358e09a430ce53128393f3c3f1e3a362a3e33344d1587
Content-Length: 3938
Server: CTYUN

<?xml version="1.0" encoding="UTF-8"?>
<ListInventoryConfigurationsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <ContinuationToken></ContinuationToken>
  <InventoryConfiguration>
    <Destination>
      <OOSBucketDestination>
        <Bucket>arn:ctyun:oos:::docs</Bucket>
        <Format>CSV</Format>
        <Prefix>0</Prefix>
      </OOSBucketDestination>
    </Destination>
    <Filter>
      <Prefix></Prefix>
    </Filter>
    <Id>0</Id>
    <IsEnabled>true</IsEnabled>
    <OptionalFields>
      <Field>Size</Field>
    </OptionalFields>
    <Schedule>
      <Frequency>Daily</Frequency>
    </Schedule>
  </InventoryConfiguration>
  <InventoryConfiguration>
    <Destination>
      <OOSBucketDestination>
        <Bucket>arn:ctyun:oos:::docs</Bucket>
        <Format>CSV</Format>
        <Prefix>1</Prefix>
      </OOSBucketDestination>
    </Destination>
    <Filter>
      <Prefix></Prefix>
    </Filter>
```

```
    <Id>1</Id>
    <IsEnabled>true</IsEnabled>
    <OptionalFields>
      <Field>Size</Field>
    </OptionalFields>
    <Schedule>
      <Frequency>Daily</Frequency>
    </Schedule>
  </InventoryConfiguration>

......

  <InventoryConfiguration>
    <Destination>
      <OOSBucketDestination>
        <Bucket>arn:ctyun:oos:::docs</Bucket>
        <Format>CSV</Format>
        <Prefix>9</Prefix>
      </OOSBucketDestination>
    </Destination>
    <Filter>
      <Prefix></Prefix>
    </Filter>
    <Id>9</Id>
    <IsEnabled>true</IsEnabled>
    <OptionalFields>
      <Field>Size</Field>
    </OptionalFields>
    <Schedule>
      <Frequency>Daily</Frequency>
    </Schedule>
  </InventoryConfiguration>
  <IsTruncated>false</IsTruncated>
</ListInventoryConfigurationsResult>
```

4.3.28 DELETE Bucket Inventory Configuration

此操作用来删除某 Bucket 的指定 Bucket 清单配置。

- **请求语法**

```
DELETE /?inventory&id=Id HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Authorization: SignatureValue
```

- **请求头**

名称	描述	是否必须
id	清单名称。 类型：字符串 取值：只允许使用小写字母、数字、短横线（-）和下划线（_），且不能以短横线（-）或下划线（_）开头或结尾，1~64 个字符。	是

- **请求示例**

```
DELETE /?inventory&id=1 HTTP/1.1
Date: Wed, 23 Aug 2023 07:09:10 GMT
Host: example-bucket.oos-cn.ctyunapi.cn
Connection: keep-alive
Authorization: SignatureValue
```

- **响应示例**

```
HTTP/1.1 204 No Content
Date: Wed, 23 Aug 2023 07:09:10 GMT
x-amz-request-id: 2d286939dfb64203257168797f7c7f5e36766a7e73748d7963
Server: CTYUN
```

4.4 关于 Object 的操作

说明：以下举例中的域名均以对象存储网络的 OOS API 域名 oos-cn.ctyunapi.cn 为例。如果是对象存储网络 2，请使用域名 oos-cn2.ctyunapi.cn。如果是香港精品网，请使用域名 oos-cnhk-hqnet.ctyunapi.cn。如果是香港普通网，请使用域名 oos-cnhk-nqnet.ctyunapi.cn。

4.4.1 PUT Object

此操作用来向指定 Bucket 中添加一个文件，要求发送请求者对该 Bucket 有写权限，用户必须添加完整的文件。

说明：文件名称不能包含 ASCII 码为 0 的字符（NUL）。

- **请求语法**

```
PUT /ObjectName HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Authorization: SignatureValue
```

- **请求头格式**

名称	描述	是否必须
Cache-Control	按照请求/回应的方式用来定义缓存行为。 类型：字符串。	否
Content-Disposition	指出文件的描述性的信息。 类型：字符串。	否
Content-Encoding	指出文件所使用的编码格式。 类型：字符串。	否
Content-Length	指定文件的大小，单位为字节。 类型：字符串。	是
Content-MD5	按照 RFC 1864，使用 base64 编码格式生成信息的 128 位 MD5 值。此请求头可以用作数据完整性检查，以验证数据是否与客户端发送的数据相同。	否

	类型：字符串。	
Content-Type	标准的 MIME 类型用来描述内容格式。 类型：字符串。 取值：MIME 类型。默认值为 application/octet-stream。	否
Expires	文件不再被缓存的时间。 类型：字符串。	否
x-amz-meta-	任何头以这个前缀开始都会被认为是用户的元数据，当用户检索时，它将会和文件一起被存储并返回。PUT 请求头大小限制为 8KiB。在 PUT 请求头中，用户定义的元数据大小限制为 2KiB。 类型：字符串。	否
x-amz-limit	文件上传限制的速率。格式为：x-amz-limit:rate=xxx。 类型：字符串。 取值：大于 0 的正整数，单位是 KiB/s。当取值是大于 0 小于 128 的整数时，按速率等于 128KiB/s 处理。	否
x-amz-storage-class	数据的存储类型。 类型：字符串。 取值: ● STANDARD：标准存储。 ● STANDARD_IA：低频访问存储。 默认值为STANDARD。	否
x-ctyun-data-location	设置数据存储的位置。 **注意**：香港节点不支持此参数。 类型：字符串。 取值： 格式为：type=Local,scheduleStrategy=*scheduleStrategy*或者type=Specified,location=*location*,scheduleStrategy=*scheduleStrategy*	否

	● type：指定数据存储位置的类型，取值为 Local 或者 Specified。local 表示就近写入，Specified 表示指定位置。如果 type 取值为 Specified，则需要指定具体的数据位置 location，location 可以填写多个，以逗号分隔。对于对象存储网络，可取值为：ChengDu、GuiYang、LaSa、LanZhou、QingDao、SH2、ShenYang、ShenZhen、SuZhou、WuHan、WuHu、WuLuMuQi、ZhengZhou；对于对象存储网络 2，可取值为：NeiMeng1、HangZhou1。 ● scheduleStrategy：调度策略，取值为： ➢ Allowed：允许 OOS 自动调度数据存储位置。 ➢ NotAllowed：不允许 OOS 自动调度数据存储位置。	

- **请求示例**

在名叫 example-bucket 的 Bucket 中，存储一张叫 my-image.jpg 的图片。

```
PUT /my-image.jpg HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
Date: Mon, 03 Sep 2012 17:50:00 GMT
Content-Type: image/jpeg
Content-Length: 11434
Authorization: SignatureValue
[11434 bytes of object data]
```

- **响应示例**

```
HTTP/1.1 100 Continue

HTTP/1.1 200 OK
x-amz-request-id: 767bc103031946fa81fbfdf4f6fcbbbdc0b1b3b5b7b9bbbdbf
Date: Mon, 03 Sep 2012 17:50:00 GMT
ETag: "1b2cf535f27731c974343645a3985328"
Content-Length: 0
```

```
Server: CTYUN
```

4.4.2 GET Object

此操作用来检索在 OOS 中的文件信息，执行此操作，用户必须对 Object 所在的 Bucket 有读权限。如果 Bucket 是 public-read 的权限，匿名用户也可以通过非授权的方式进行读操作。

- **请求语法**

```
GET /ObjectName HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Authorization: SignatureValue
```

- **请求变量**

变量	描述	是否必须
response-content-type	设置返回头中的 Content-Type。 类型：字符串	否
response-content-language	设置返回头中的 Content-Language。 类型：字符串	否
response-cache-control	设置返回头中的 Cache-Control。 类型：字符串	否
response-content-disposition	设置返回头中的 Content-Disposition。 注：OOS 会把 response-content-disposition 中的值设置到响应头 Content-Disposition 中。对于不同的浏览器，此值的编码方式可能不同，此工作由客户端来完成。例如对于 IE 浏览器，要设置下载的文件名为”文件.txt”，那么 response-content-disposition 要设置为 attachment;filename=URLEncoder.encod	否

	e(URLEncoder.encode("文件.txt",”UTF-8”), ”UTF-8”)。 类型：字符串	
response-content-encoding	设置返回头中的 Content-Encoding。 类型：字符串	否
response-expires	设置返回头中的 Expires。 类型：字符串	否
x-amz-limitrate	文件下载限制的速率。 **注意**：x-amz-limitrate 和 x-amz-limit 只能二选一。 类型：字符串 取值：取值为大于 0 的正整数，单位是 KiB/s。	否
x-amz-limit	文件下载限制的速率。 **注意**：x-amz-limit 和 x-amz-limitrate 只能二选一。 类型：字符串 取值：格式为： ● x-amz-limit:rate=xxx ● x-amz-limit:concurrency=xxx ● x-amz-limit:rate=xxx, concurrency=xxx 其中 ● rate 为速率，取值为大于 0 的正整数，单位是 KiB/s。当取值是大于 0 小于 128 的整数时，按速率等于 128KiB/s 处理。 ● concurrency 为并发连接数。取值为大于 0 的正整数。	否

- **请求头格式**

名称	描述	是否必须
Range	指定下载文件的字节范围。 类型：字符串。	否
If-Modified-Since	只返回一个在指定时间点后被修改的文件，否则返回 304 错误 类型：字符串	否
If-Unmodified-Since	返回一个在指定时间点后未被修改的文件，否则返回 412 错误。 类型：字符串	否
If-Match	当文件的 ETag 与指定值一致时，返回此文件。否则返回 412 错误。 类型：字符串	否
If-None-Match	当文件的 ETag 与指定值不一致时，返回此文件。否则返回 304 错误。 类型：字符串	否
x-amz-limitrate	文件下载限制的速率。 **注意**：x-amz-limitrate 和 x-amz-limit 只能二选一。 类型：字符串 取值：大于 0 的正整数，单位是 KiB/s。	否
x-amz-limit	文件下载限制的速率。 **注意**：**x-amz-limit** 和 **x-amz-limitrate** 只能二选一。 类型：字符串 取值：格式为： ● x-amz-limit:rate=xxx ● x-amz-limit:concurrency=xxx ● x-amz-limit:rate=xxx, concurrency=xxx 其中	否

	• rate 为速率，取值为大于 0 的正整数，单位是 KiB/s。当取值是大于 0 小于 128 的整数时，按速率等于 128KiB/s 处理。 • concurrency 为并发连接数。取值为大于 0 的正整数。	

- **响应头**

变量	描述
x-amz-expiration	如果文件被配置了到期时间，那么 OOS 返回此响应头。这个响应头包含键值对 *expiry-date* 和 *rule-id*。*rule-id* 的值是 URL 编码的。
x-ctyun-metadata-location	文件的索引位置。 **注意**：香港节点不会返回此项。 类型：枚举 取值：对于对象存储网络，取值：ChengDu、FuZhou、GuiYang、HangZhou、LaSa、LanZhou、QingDao、ShenYang、ShenZhen、WuHan、WuHu、WuLuMuQi、ZhengZhou、SH2、SuZhou；对于对象存储网络 2，取值为：NeiMeng1、HangZhou1。
x-ctyun-data-location	获取文件的数据位置。 **注意**：香港节点不会返回此项。 类型：枚举 取值：对于对象存储网络，取值：ChengDu、GuiYang、LaSa、LanZhou、QingDao、SH2、ShenYang、ShenZhen、SuZhou、WuHan、WuHu、WuLuMuQi、ZhengZhou；对于对象存储网络 2，取值为：NeiMeng1、HangZhou1。
x-amz-meta-*	以该前缀开头的用户定义的元数据响应头。每一个都作为一组键值对存储和返回。OOS 不验证或解释用户定义的元数据。 类型：字符串。

x-amz-storage-class	文件的存储类型，如果存储类型为 STANDARD，则不返回此参数。 类型：字符串 取值：STANDARD_IA：低频访问存储。

- **请求示例**

```
GET /test.txt HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
Authorization: SignatureValue
Date: Mon, 15 Nov 2021 06:39:40 GMT
Content-Type: application/octet-stream
Connection: Keep-Alive
```

- **响应示例**

```
HTTP/1.1 200 OK
Content-Length: 1467326
Date: Mon, 15 Nov 2021 06:39:40 GMT
x-amz-request-id: 75d41c3c69fa4ad968d4d8dfe7dfe1e7a6a8ab9c9ea0a2a4a6
ETag: "5db44ee68a1e577907c2699c8f582107"
Last-Modified: Mon, 15 Nov 2021 06:39:39 GMT
Content-Type: application/x-java-archive
Content-MD5: XbRO5ooeV3kHwmmcj1ghBw==
x-ctyun-metadata-location: ChengDu
x-ctyun-data-location: ZhengZhou
Server: CTYUN
```

4.4.3 DELETE Object

此操作用来删除指定的文件，用户要对文件所在的 Bucket 拥有写权限才可以执行此操作。

- **请求语法**

```
DELETE /ObjectName HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Content-Length: length
Authorization: SignatureValue
```

- **请求示例**

删除图片文件 my-image.jpg。

```
DELETE /my-image.jpg HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
Date: Mon, 03 Sep 2012 17:50:00 GMT
Content-Type: application/xml; charset=utf-8
Authorization: SignatureValue
```

- **响应示例**

```
HTTP/1.1 204 NoContent
x-amz-request-id: 5d5becc10cdd4f25047e8077797f3e40443436383a3c3e4042
Date: Mon, 03 Sep 2012 17:50:00 GMT
Server: CTYUN
```

4.4.4 PUT Object - Copy

此操作用来创建一个存储在 OOS 里的文件拷贝。类似于执行一个 GET，然后再执行一次 PUT。要执行拷贝请求，用户需要对源文件有读权限，对目标 Bucket 有写权限。

注意：当 OOS 接收到请求或者正在执行拷贝操作时，拷贝操作可能会返回失败。如果在拷贝操作开始之前出现异常，OOS 返回标准的错误信息。如果在拷贝操作过程中出现异常，由于 200 OK 状态码是先返回的，这意味着 200 OK 响应体可能包含成功或错误。请在客户端应用程序中解析响应体的内容并进行适当处理。

- **请求语法**

```
PUT /destinationObject HTTP/1.1
Host: destinationBucket.oos-cn.ctyunapi.cn
x-amz-copy-source: /source_bucket/sourceObject
x-amz-metadata-directive: metadata_directive
x-amz-copy-source-if-match: etag
x-amz-copy-source-if-none-match: etag
x-amz-copy-source-if-unmodified-since: time_stamp
x-amz-copy-source-if-modified-since: time_stamp
x-amz-meta-paramater: value
Authorization: SignatureValue
Date: date
```

- **请求头格式**

名称	描述	是否必须
x-amz-copy-source	源 Bucket 和文件的名称，用斜杠(/)分割。 类型：字符串	是
x-amz-metadata-directive	指明元数据是源文件的拷贝或者元数据被请求头提供的元数据覆盖。 类型：字符串 取值： ● COPY：除存储类型（x-amz-storage-class）、数据位置（x-ctyun-data-location）外的其他元	否

	数据保持不变，拷贝源文件的元数据。 ● REPLACE：所有原始元数据都被指定的元数据覆盖。 默认值为 COPY。 **注意**：如果取值为 COPY，源文件和目的文件相同，则必须携带 x-amz-storage-class，否则不能拷贝，返回 400 错误码。	
x-amz-copy-source-if-match	只有当源文件的 Etag 与给定 Etag 匹配时，才能执行文件拷贝操作，否则返回 412 HTTP 状态码错误。 类型：字符串	否
x-amz-copy-source-if-none-match	只有当源文件的 Etag 与给定 Etag 不匹配时，才能执行文件拷贝操作，否则返回 412 HTTP 状态码错误。 类型：字符串	否
x-amz-copy-source-if-unmodified-since	只有源文件在指定时间点之后没有修改，才执行文件拷贝操作，否则返回 412 错误。 类型：字符串 取值：格式为 EEE, d MMM yyyy HH:mm:ss 'GMT'。	否
x-amz-copy-source-if-modified-since	只有源文件在指定时间点之后修改过，才执行文件拷贝操作，否则返回 412 错误。 类型：字符串。	否

	取值：格式为 EEE, d MMM yyyy HH:mm:ss 'GMT'。	
x-amz-storage-class	目标文件的存储类型。 类型：字符串。 取值: ● **STANDARD**：标准存储。 ● **STANDARD_IA**：低频访问存储 默认值为 STANDARD。	否
x-ctyun-data-location	设置数据存储的位置。 **注意**：香港节点不支持此参数。 类型：字符串。 取值： 格式为： type=Local,scheduleStrategy=*scheduleStrategy*或者 type=Specified,location=*location*,scheduleStrategy=*scheduleStrategy* ● type：指定数据存储位置的类型，取值为 Local 或者 Specified。Local 表示就近写入，Specified 表示指定位置。如果 type 取值为 Specified，则需要指定具体的数据位置 location，location 可以填写多个，以逗号分隔，对于对象存储网络，可取值为：ChengDu、GuiYang、LaSa、LanZhou、QingDao、SH2、ShenYang、ShenZhen、SuZhou、WuHan、WuHu、WuLuMuQi、	否

	ZhengZhou；对于对象存储网络 2，可取值为：NeiMeng1、HangZhou1。 ● scheduleStrategy：调度策略，取值为： ➢ **Allowed**：允许 OOS 自动调度数据存储位置。 ➢ **NotAllowed**：不允许 OOS 自动调度数据存储位置。	
x-amz-meta-*paramater*	用户自定义的元数据，用户可以根据需要，自定义一些元数据的参数。 类型：字符串。	否

- **响应结果**

名称	描述
Content-Length	响应体的长度。只有客户端携带 Expect: 102-processing 请求头，才会返回该响应头。
CopyObjectResult	包含所有响应结果的容器。 类型：容器 子节点：LastModified、Etag
LastModified	返回文件最后一次修改的日期。 类型：字符串 父节点：CopyObjectResult
ETag	返回新文件的 ETag。ETag 只反映文件内容发生了改变，元数据未改变。 类型：字符串 父节点：CopyObjectResult

- **请求示例**

将存储桶 example-bucket 中的文件 1.txt 复制一份到本存储桶，新文件命名为 2.txt。

```
PUT /2.txt HTTP/1.1
Host: example-bucket.oos-dhv6.ctyunapi.cn
User-Agent: curl/7.68.0
Accept: */*
x-amz-date: 20210625T033636Z
x-amz-content-sha256: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
Content-Type: application/x-www-form-urlencoded; charset=utf-8
x-amz-copy-source: /testbucket001/1.txt
Content-Length: 0
Authorization: SignatureValue
```

- **响应示例**

```
HTTP/1.1 200 OK
Date: Fri, 25 Jun 2021 03:37:22 GMT
x-amz-request-id: 28fcb1549c45469228746b7c827f82616880808d8876918e66
Content-Type: application/xml
Transfer-Encoding: chunked
Server: CTYUN

<CopyObjectResult>
    <LastModified>2021-06-25T03:37:23.727Z</LastModified>
    <ETag>eb733a00c0c9d336e65691a37ab54293</ETag>
</CopyObjectResult>
```

4.4.5 Initial Multipart Upload

本接口初始化一个分片上传（Multipart Upload）操作，并返回一个上传 ID。此 ID 用来将此次分片上传操作中上传的所有片段合并成一个文件。用户在执行每一次子上传请求（见 Upload Part）时都必须指定该 ID。用户也可以在表示整个分片上传完成的合并分片的请求中指定该 ID。或者在用户放弃该分片上传操作时指定该 ID。

- **请求语法**

```
POST /ObjectName?uploads HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Authorization: SignatureValue
```

- **请求头格式**

名称	描述	是否必须
Cache-Control	可以用来指定请求或响应中的缓存操作。 类型：字符串	否
Content-Disposition	指定文件的描述性信息。 类型：字符串	否
Content-Encoding	指定文件的描述性信息采用何种编码方式以及在获取被 Content-Type 头字段引用的 media-type 时采用何种解码方式。 类型：字符串	否
Content-Type	用来描述文件数据格式的标准 MIME 类型。 类型：字符串 取值：MIME 类型，默认值为 application/octet-stream。	否
Expires	文件不再被缓存的时间。GMT 时间格式。 类型：字符串	否

x-amz-meta-	任何以 x-amz-meta-为前缀的头都被当作用户元数据，它和文件一起存储，当用户获取该文件的时候作为响应的一部分被返回。 类型：字符串	否
x-amz-storage-class	文件的存储类型，针对那些在成功完成分片上传后被创建的文件。 类型：字符串 取值： ● STANDARD：标准存储。 ● STANDARD_IA：低频访问存储。 默认值为 STANDARD。	否
x-ctyun-data-location	设置Bucket的数据位置。 **注意**：香港节点不支持此参数。 类型：key-value形式 取值： 格式为：type=Local,scheduleStrategy=*scheduleStrategy*或者type=Specified,location=*location*,scheduleStrategy=*scheduleStrategy* ● type：指定数据存储位置的类型，取值为 Local 或者 Specified。local 表示就近写入，Specified 表示指定位置。如果 type 取值为 Specified，则需要指定具体的数据位置 location，location 可以填写多个，以逗号分隔，对于对象存储网络，可取值为：ChengDu、GuiYang、LaSa、LanZhou、QingDao、SH2、ShenYang、ShenZhen、SuZhou、WuHan、WuHu、WuLuMuQi、ZhengZhou；对于对象存储网络 2，可取值为：NeiMeng1、HangZhou1。 ● scheduleStrategy：调度策略，取值为：	否

	➢ Allowed：允许 OOS 自动调度数据存储位置 ➢ NotAllowed：不允许 OOS 自动调度数据存储位置。	

- **响应结果**

名称	描述
InitiateMultipartUploadResult	包含所有响应的容器。 类型：容器 子节点：Bucket，Key，UploadId
Bucket	分片上传对应的 Bucket 的名称。 类型：Sring 父节点：InitiateMultipartUploadResult
Key	分片上传对应的文件名称。 类型：字符串 父节点：InitiateMultipartUploadResult
UploadId	分片上传 ID。 类型：字符串 父节点：InitiateMultipartUploadResult

- **请求示例**

初始化文件“example-object”的分片上传操作。

```
POST /example-object?uploads HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
Date: Mon, 1 Nov 2010 20:34:56 GMT
Authorization: SignatureValue
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id: 656c76696e6727732072657175657415ecaced5ddd5d7dda
Date: Mon, 1 Nov 2010 20:34:56 GMT
Content-Length: 197
Server: CTYUN
```

```
<?xml version="1.0" encoding="UTF-8"?>
<InitiateMultipartUploadResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Bucket>example-bucket</Bucket>
    <Key>example-object</Key>
   <UploadId>1638428231343309398</UploadId>
</InitiateMultipartUploadResult>
```

4.4.6 Upload Part

该接口用于实现分片上传操作中片段的上传。

在上传任何一个分片之前，必须执行 Initial Multipart Upload 操作来初始化分片上传操作，初始化成功后，OOS 会返回一个上传 ID，这是一个唯一的标识，用户必须在调用 Upload Part 接口时加入该 ID。

分片号 PartNumber 可以唯一标识一个片段并且定义该分片在文件中的位置，范围从 1 到 10000。如果用户用之前上传过的片段的分片号来上传新的分片，之前的分片将会被覆盖。

除了最后一个分片外，所有分片的都不小于 5M，最后一个分片的大小不受限制。

为了确保数据不会由于网络传输而毁坏，需要在每个分片上传请求中指定 Content-MD5 头，OOS 通过提供的 Content-MD5 值来检查数据的完整性，如果不匹配，则会返回一个错误信息。

- **请求语法**

```
PUT /ObjectName?partNumber=PartNumber&uploadId=UploadId HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Content-Length: Size
Authorization: SignatureValue
```

- **请求头格式**

名称	描述	是否必须
partNumber	标识片段的分片号。 类型：整型	是
uploadId	分片上传 ID。 类型：字符串	是
Content-Length	该分片的大小，单位为字节。 类型：字符串	是
Content-MD5	该分片数据的 128 位采用 base64 编码的 MD5 值。这个头可以用来验证该分片数据是否与原始数据值保持一致。尽管这个值	否

	是可选的，我们仍然推荐使用 Content-MD5 机制来执行端到端的一致性校验。 类型：字符串	
x-amz-limit	文件上传限制的速率。 类型：字符串 取值：格式为： ● x-amz-limit:rate=xxx ● x-amz-limit:concurrency=xxx ● x-amz-limit:rate=xxx, concurrency=xxx 其中 ● rate 为速率，取值为大于 0 的正整数，单位是 KiB/s。当取值是大于 0 小于 128 的整数时，按速率等于 128KiB/s 处理。 ● concurrency 为并发连接数。取值为大于 0 的正整数。	否
Expect	用户的应用中设置请求头为 100-continue，应用在接收到请求回应之前不会发送请求实体。如果基于请求头的消息被拒绝，消息的实体不会被发送。 类型：字符串 有效值：100-continue。	否

- **请求示例**

示例中的 PUT 请求执行一次分片上传过程中的片段（片段 1）上传操作。该请求要求包含在 Initial Multipart Upload 操作中获取到的上传 ID。

```
PUT /my-movie.m2ts?partNumber=1&uploadId=638428231343369398 HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
Date: Mon, 1 Nov 2010 20:34:56 GMT
Content-Length: 10485760
Content-MD5: pUNXr/BjKK5G2UKvaRRrOA==
Authorization: SignatureValue

***part data omitted***
```

- **响应示例**

响应中包含 Etag 头，用户需要在最后发送完成分片上传过程请求的时候包含该 Etag 值。

```
HTTP/1.1 200 OK
x-amz-request-id: 28fcb1549c45469228746b656c76696e672773207265717565
Date: Mon, 1 Nov 2010 20:34:56 GMT
ETag: "b54357faf0632cce46e942fa68356b38"
Content-Length: 0
Connection: keep-alive
Server: CTYUN
```

4.4.7 Complete Multipart Upload

该接口通过合并之前的上传片段来完成一次分片上传过程。

用户首先初始化分片上传过程，然后通过 Upload Part 接口上传所有分片。在成功将一次分片上传过程的所有相关片段上传之后，调用这个接口来结束分片上传过程。当收到这个请求的时候，OOS 会以分片号升序排列的方式将所有片段依次拼接来创建一个新的文件。在这个 Complete Multipart Upload 请求中，用户需要提供一个片段列表。同时，必须确保这个片段列表中的所有片段必须是已经上传完成的，Complete Multipart Upload 操作会将片段列表中提供的片段拼接起来。对片段列表中的每个片段，需要提供该片段上传完成时返回的 ETag 头的值和对应的分片号。

OOS 提供了不合并片段也可以读取 Object 内容的功能。在没有调用 Complete Multipart Upload 接口合并片段时，也可以通过调用 Get Object 接口来获取文件内容，OOS 会根据最近一次创建的 uploadId，以分片号升序的方式顺序读取片段内容，返回给客户端。

- **请求语法**

```
POST /ObjectName?uploadId=UploadId HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: Date
Content-Length: Size
Authorization: SignatureValue

<CompleteMultipartUpload>
    <Part>
        <PartNumber>PartNumber</PartNumber>
        <ETag>ETag</ETag>
    </Part>
...
</CompleteMultipartUpload>
```

- **请求参数**

名称	描述	是否必须

CompleteMultipartUpload	请求的容器。 类型：容器 子节点：1 个或多个 Part 元素。	是
Part	一个片段的容器。 类型：容器 父节点：CompleteMultipartUpload。 子节点：PartNumber、Etag。	是
PartNumber	标识片段的分片号。 类型：整型 父节点：Part。	是
ETag	片段上传完成时返回的 Etag 内容。 类型：字符串 父节点：Part	是

- **响应结果**

名称	描述
CompleteMultipartUploadResult	包含整个响应的容器。 类型：容器 子节点：Location、Bucket、Key、ETag
Location	新创建文件的 URI 地址。 类型：URI 父节点：CompleteMultipartUploadResult
Bucket	分片上传对应的存储桶。 类型：字符串 父节点：CompleteMultipartUploadResult
Key	新创建的文件的 Key。 类型：字符串 父节点：CompleteMultipartUploadResult

ETag	ETag 用来标识新创建的文件数据。 类型：字符串 父节点：CompleteMultipartUploadResult

- **请求示例**

下面的分片上传请求在 CompleteMultipartUpload 元素中指定了三个片段。

```
POST /example-object?uploadId=638428231343369399 HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
Date: Mon, 1 Nov 2010 20:34:56 GMT
Content-Length: 391
Authorization: SignatureValue

<CompleteMultipartUpload>
    <Part>
        <PartNumber>1</PartNumber>
        <ETag>"a54357aff0632cce46d942af68356b38"</ETag>
    </Part>
    <Part>
        <PartNumber>2</PartNumber>
        <ETag>"0c78aef83f66abc1fa1e8477f296d394"</ETag>
    </Part>
    <Part>
        <PartNumber>3</PartNumber>
        <ETag>"acbd18db4cc2f85cedef654fccc4a4d8"</ETag>
    </Part>
</CompleteMultipartUpload>
```

- **响应示例**

下面的响应中表示一个文件被拼接成功。

```
HTTP/1.1 200 OK
x-amz-request-id: f71d177b56cd4d0bcb3e31403542444a090b0fff0103050709
Date: Mon, 1 Nov 2010 20:34:56 GMT
Server: CTYUN

<?xml version="1.0" encoding="UTF-8"?>
<CompleteMultipartUploadResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<Location>http://example-bucket.oos-cn.ctyunapi.cn/example-Object</Location>
    <Bucket>example-Bucket</Bucket>
```

```
    <Key>example-Object</Key>
</CompleteMultipartUploadResult>
```

4.4.8 Abort Multipart Upload

该接口用于终止一次分片上传操作。分片上传操作被终止后，用户不能再通过上传 ID 上传其它片段，之前已上传完成的片段所占用的存储空间将被释放。如果此时任何片段正在上传，该上传过程可能会也可能不会成功。所以，为了释放所有片段所占用的存储空间，可能需要多次终止分片上传操作。

- **请求语法**

```
DELETE /ObjectName?uploadId=UploadId HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: Date
Authorization: SignatureValue
```

- **请求示例**

下面的请求通过指定上传 ID 来终止一次分片上传操作。

```
DELETE /example-object?uploadId=658428231343369399 HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
Date: Mon, 1 Nov 2010 20:34:56 GMT
Authorization: SignatureValue
```

- **响应示例**

```
HTTP/1.1 204 No Content
Date: Mon, 1 Nov 2010 20:34:56 GMT
x-amz-request-id: cafd93679f174e05428e85969c999c7b9f968d9ba0aca2a080
Server: CTYUN
```

4.4.9 List Parts

该操作用于列出一次分片上传过程中已经上传完成的所有片段。

该操作必须包含一个通过 Initial Multipart Upload 操作获取的上传 ID。该请求最多返回 1000 个上传片段信息，默认返回的片段数是 1000。用户可以通过指定 max-parts 参数来指定一次请求返回的片段数。如果用户的分片上传过程超过 1000 个片段，响应中的 IsTruncated 字段的值则被设置成 true，并且指定一个 NextPartNumberMarker 元素。用户可以在下一个连续的 List Part 请求中加入 *part-number-marker* 参数，并把它的值设置成上一个请求返回的 NextPartNumberMarker 值。

- **请求语法**

```
GET /ObjectName?uploadId=UploadId HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: Date
Authorization: SignatureValue
```

- **请求头**

名称	描述	是否必须
uploadId	该 ID 用来标识一个分片上传过程。 类型：字符串	是
max-parts	设置响应体中返回的片段的最大数目。 类型：字符串 取值：[1, 1000]，默认值为 1000。	否
part-number-marker	指定此次列表的起始片段的分片号，只有比该片段的分片号更高的片段才会被列举出来。 类型：字符串	否
encoding-type	指定响应中 Key 的编码类型。如果 Key 包含 xml 1.0 标准不支持的控制字符，可通过设置该参数对响应中的 Key 进行编码。 类型：字符串	否

	取值：url，字母不区分大小写。	

- **响应结果**

名称	描述
ListPartsResult	包含整个响应的容器。 类型：容器 子节点：Bucket、EncodingType、Key、UploadId、Initiator、Owner、StorageClass、PartNumberMarker、NextPartNumberMarker、MaxParts、IsTruncated、Part
Bucket	分片上传对应的 bucket 名称。 类型：字符串 父节点：ListPartsResult
EncodingType	Key 的编码类型。 类型：字符串 父节点：ListPartsResult
Key	新创建的文件的 Key。 类型：字符串 父节点：ListPartsResult。
UploadId	分片上传 ID。 类型：字符串 父节点：ListPartsResult
Initiator	指定执行此次分片上传过程的用户账户。 类型：容器 父节点：ListPartsResult 子节点：ID，DisplayName
ID	OOS 账户的 ID。 类型：字符串 父节点：Initiator

DisplayName	OOS 账户的账户名。 类型：字符串 父节点：Initiator
StorageClass	文件的存储类型： ● STANDARD：标准存储。 ● STANDARD_IA：低频访问存储。 类型：字符串 父节点：ListPartsResult
PartNumberMarker	列表起始位置的片段的分片号。 类型：整型 父节点：ListPartsResult
NextPartNumberMarker	当此次请求没有将所有片段列举完时，此元素指定列表中的最后一个片段的分片号。此分片号用于作为下一次连续列表请求的 *part-number-marker* 参数的值。 类型：整型 父节点：ListPartsResult
MaxParts	响应中片段的最大数目。 类型：整型 父节点：ListPartsResult
IsTruncated	标识此次分片上传过程中的所有片段是否全部被列出，如果为 true 则表示没有全部列出。如果分片上传过程的片段数超过了 MaxParts 元素指定的最大数，则会导致一次列表请求无法将所有片段数列出。 类型：Boolean 父节点：ListPartsResult
Part	与某个片段对应的容器，响应中可能包含 0 个或多个 Part 元素。 类型：容器 父节点：ListPartsResult

	子节点：PartNumber、LastModified、ETag、Size
PartNumber	标识片段的分片号。 类型：整型 父节点：Part
LastModified	片段上传完成的日期。 类型：Date 父节点：Part
ETag	片段上传完成时返回的 ETag 值。 类型：字符串。 父节点：Part
Size	片段的数据大小。 类型：整型 父节点：Part

- **请求示例**

假设用户上传了一系列以 1 开头的有连续分片号的片段，下面的 List Part 请求指定了 *max-parts* 和 *part-number-marker* 查询参数。此次请求列出了紧随片段 1 的两个片段，从请求响应体中可以获取到片段 2 和片段 3。如果有更多的片段存在，则片段列表操作没有完成，响应体中将会包含值为 true 的 IsTruncated 元素。同时，响应体中还会包含值为 3 的 NextPartNumberMarker 元素，这个值用来指定在下一次连续的列表操作中 part-number-marker 参数的值。

```
GET /example-object?uploadId=6684282313433693 99&max-parts=2&part-number-marker=1
HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
Date: Mon, 1 Nov 2010 20:34:56 GMT
Authorization: SignatureValue
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id: c5af22da52524558f16457665b686a702f313b2527292b2d2f
Date: Mon, 1 Nov 2010 20:34:56 GMT
```

```
Content-Length: 1014
Connection: keep-alive
Server: CTYUN

<?xml version="1.0" encoding="UTF-8"?>
<ListPartsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Bucket>example-bucket</Bucket>
    <EncodingType></EncodingType>
    <Key>example-object</Key>
    <UploadId>16384282231343309399</UploadId>
    <Initiator>
       <ID>mailaddress@ctyun.cn</ID>
       <DisplayName>umat-user-11116a31-17b5-4fb7-9df5-b288870f11xx</DisplayName>
    </Initiator>
    <Owner>
       <ID></ID>
       <DisplayName></DisplayName>
    </Owner>
    <StorageClass>STANDARD</StorageClass>
    <PartNumberMarker>1</PartNumberMarker>
    <NextPartNumberMarker>3</NextPartNumberMarker>
    <MaxParts>2</MaxParts>
    <IsTruncated>true</IsTruncated>
    <Part>
       <PartNumber>2</PartNumber>
       <LastModified>2010-11-10T20:48:34.000Z</LastModified>
       <ETag>"7778aef83f66abc1fa1e8477f296d394"</ETag>
       <Size>10485760</Size>
    </Part>
    <Part>
       <PartNumber>3</PartNumber>
       <LastModified>2010-11-10T20:48:33.000Z</LastModified>
       <ETag>"aaaa18db4cc2f85cedef654fccc4a4x8"</ETag>
       <Size>10485760</Size>
    </Part>
</ListPartsResult>
```

4.4.10 Copy Part

可以将已经存在的 Object 作为分段上传的片段，拷贝生成一个新的片段。需要指定请求头 x-amz-copy-source 来定义拷贝源。如果只拷贝源 Object 中的一部分，需要增加请求头 x-amz-copy-source-range。PartNumber 为新文件分片号，UploadId 为新文件的分片上传 ID。

在上传任何一个分片之前，必须执行 Initial Multipart Upload 操作来初始化分片上传操作，初始化成功后，OOS 会返回一个上传 ID，这是一个唯一的标识，用户必须在调用 Copy Part 接口时加入该 ID。

- **请求语法**

```
PUT /ObjectName?partNumber=PartNumber&uploadId=UploadId HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
x-amz-copy-source: /source_bucket/sourceObject
x-amz-copy-source-range:bytes=first-last
x-amz-copy-source-if-match: etag
x-amz-copy-source-if-none-match: etag
x-amz-copy-source-if-unmodified-since: time_stamp
x-amz-copy-source-if-modified-since: time_stamp
Date: Date
Authorization: SignatureValue
```

- **请求头**

名称	描述	是否必须
x-amz-copy-source	指定源 BucketName 和 ObjectName，用斜杠/分隔。 类型：字符串	是
x-amz-copy-source-range	要从源 object 拷贝的 bytes 范围。Range 值的格式是 bytes=第一个字节-最后一个字节。第一个字节从 0 开始。例如要拷贝前 10 个字节，bytes=0-9。只允许对大于 5G 的源 object 进行部分拷贝的操作。 如果要拷贝整个 object，不需要这个头。	否

	复制整个源文件时不需要此请求标头。 类型：字符串	
x-amz-copy-source-if-match	只有文件的实体 Etag 与给定 Etag 匹配，才执行拷贝文件的操作。否则，返回 412 错误码。 类型：字符串	否
x-amz-copy-source-if-none-match	只有文件实体 Etag 和指定实体 Etag 不同，才执行拷贝操作。否则，返回 412 错误码。 类型：字符串。	否
x-amz-copy-source-if-unmodified-since	只有文件在指定时间点之后没有修改过，才执行拷贝操作。否则，返回 412 错误码。格式为 EEE, d MMM yyyy HH:mm:ss 'GMT'。 类型：字符串	否
x-amz-copy-source-if-modified-since	只有文件在指定时间点之后被修改过，才执行拷贝操作。否则，返回 412 错误码。格式为 EEE, d MMM yyyy HH:mm:ss 'GMT'。 类型：字符串	否

- **响应结果**

名称	描述
Content-Length	响应体的长度。只有客户端携带 Expect: 102-processing 请求头，才会返回该响应头。
CopyPartResult	包含整个响应的容器。 类型：容器 子节点：LastModified、ETag

LastModified	分片的最后修改时间。 类型：字符串 父节点：CopyPartResult
ETag	新分片的 ETag。 类型：字符串 父节点：CopyPartResult

- **请求示例**

通过从源 Object 中指定范围，拷贝生成一个新片段。

```
PUT /newobject?partNumber=2&uploadId=738428231343369398 HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
Date: Mon, 11 Apr 2011 20:34:56 GMT
x-amz-copy-source: /source-bucket/sourceobject
x-amz-copy-source-range:bytes=500-6291456
Authorization: SignatureValue
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id: fd02454b7a5143db239689988d9a9ca261636b57595b5d5f61
Date: Mon, 1 Nov 2010 20:34:56 GMT
Server: CTYUN

<CopyPartResult xmlns=”http://s3.amazonaws.com/doc/2006-03-01/”>
     <LastModified>2009-10-28T22:32:00</LastModified>
     <ETag>9b2cf535f27731c974343645a3985328</ETag>
</CopyPartResult>
```

4.4.11 Delete Multiple Objects

批量删除 Object 功能支持用一个 HTTP 请求删除一个 Bucket 中的多个 Object。如果你知道你想删除的 Object 名字，此功能可以批量删除这些 Object，而不用发送多个单独的删除请求。

批量删除请求包含一个不超过 1000 个 Object 的 XML 列表。在这个 xml 中，需要指定要删除的 object 的名字。对于每个 Object，OOS 都会返回删除的结果，成功或者失败。注意，如果请求中的 Object 不存在，那么 OOS 也会返回删除成功。

批量删除功能支持两种格式的响应，全面信息和简明信息。默认情况下，OOS 在响应中会显示全面信息，即包含每个 Object 的删除结果。在简明信息模式下，OOS 只返回删除出错的 object 的结果。对于成功删除的 Object，在响应中将不返回任何信息。

最后，批量删除功能必须使用 Content-MD5 请求头，OOS 使用此头来保证请求体在传输过程中没有被修改。

- **请求语法**

```
POST /?delete HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Content-Length: Size
Content-MD5: MD5
Authorization: SignatureValue
Date: Date

<?xml version="1.0" encoding="UTF-8"?>
<Delete>
    <Quiet>true</Quiet>
    <Object>
        <Key>Key</Key>
    </Object>
    <Object>
        <Key>Key</Key>
    </Object>
    ...
</Delete>
```

- **请求头**

名称	描述	是否必须

Content-MD5	Base64 编码，128 位的 MD5，这个请求头必须被使用，以保证数据在传输过程中没有被篡改。参考 RFC1864。 类型：字符串	是
Content-Length	请求体的长度。 类型：字符串 取值：大于等于 0，小于等于 2*1024*1024 的整数。	是

- **请求参数**

名称	描述	是否必须
Delete	包含整个请求的容器。 类型：容器 子节点：Quiet、Object	是
Quiet	使用简明信息模式来返回响应。 类型：Boolean 取值： ● true：使用简明信息模式返回响应。 ● false：采用详细信息模式返回响应。 默认值为 false。 父节点：Delete	否
Object	包含被删除 object 的容器。 类型：容器 父节点：Delete 子节点：Key	是
Key	被删除 object 名。 类型：字符串 父节点：Object	是

- **响应结果**

名称	描述
DeleteResult	包含整个响应的容器。 类型：容器 子节点：Deleted、Error
Deleted	成功删除的容器，包含成功删除的 object。 类型：容器 父节点：DeleteResult
Key	尝试删除的 object 名。 类型：字符串 父节点：Deleted，或 Error
Error	删除失败的容器，包含删除失败的 object 信息。 类型：容器 父节点：DeleteResult 子节点：Key、Code、Message
Code	删除失败的状态码：AccessDenied、InternalError。 类型：字符串 父节点：Error
Message	错误的描述。 类型：字符串 父节点：Error

- **请求示例**

下面的请求批量删除一些 object，有些删除成功，有些失败（例如没有权限删除）。

```
POST /?delete HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
Date: Mon, 11 Apr 2011 20:34:56 GMT
Content-MD5: p5/WA/oEr30qrEEl21PAqw==
Authorization: SignatureValue
Content-Length: 125
Connection: Keep-Alive
```

```
<Delete>
  <Object>
    <Key>sample1.txt</Key>
  </Object>
  <Object>
    <Key>sample2.txt</Key>
  </Object>
</Delete>
```

- 响应示例

```
HTTP/1.1 200 OK
x-amz-request-id: 39b231ac529a49594cbfb2c1b6c3c5cb8a8c9580828486888a
Date: Mon, 11 Apr 2011 20:34:56 GMT
Server: CTYUN
Content-Length: 251

<?xml version="1.0" encoding="UTF-8"?>
<DeleteResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Deleted>
        <Key>sample1.txt</Key>
    </Deleted>
    <Error>
        <Key>sample2.txt</Key>
        <Code>AccessDenied</Code>
        <Message>AccessDenied</Message>
    </Error>
</DeleteResult>
```

4.4.12 断点续传

通过 MultipleUpload 类以及文件上传请求 UploadFileRequest 类，实现基于分段上传的断点续传的功能。

- **参数设置**

名称	描述
EnableCheckpoint	是否开启断点续传功能。 取值： ● true：开启。 ● false：关闭。 默认值为 false。
PartSize	每个分段的大小 partSize，若 partSize 小于 5MiB，则会将 partSize 调整至 5MiB。 默认：5MiB。
UploadFile	上传的本地文件路径。
checkpointFile	记录本地分片上传结果的文件。如果未指定 checkpointFile，默认为 uploadFile.ucp，与上传的本地文件同路径。
objectMetadata	文件的元数据。
ProgressListener	上传状态监听器。
TaskNum	分片上传并发线程数，默认为 1，最多为 1000。

用 checkpointFile 来记录所有分片的状态。上传过程中的进度信息会保存在该文件中，如果某一分片上传失败，再次上传时会根据文件中记录的点继续上传。上传完成后，该文件会被删除。checkpointfile 默认与待上传的本地文件同目录，为 uploadFile.ucp。

- **Java 上传代码示例**

```
public static void multiUpload(AmazonS3 ossClient) {
       try {
           String bucketName = "testbucketnamexxx";
           String key = "object_test1.pdf" ;
```

```
            String uploadFile = "D:\\tmp\\2.txt";

            // 通过 UploadFileRequest 设置多个参数
            UploadFileRequest request = new UploadFileRequest(bucketName, key);

            // 上传的本地文件
            request.setUploadFile(uploadFile);

            // 分片上传并发线程数，默认为 1，最多为 1000
            request.setTaskNum(2);

            /*每个分片的大小，默认 5MiB，若 partSize 小于 5MiB，除了最后一个分片以外，会将
partSize 调整至 5MiB */
            request.setPartSize(5 * 1024 * 1024);

            // 开启断点续传功能，默认关闭
            request.setEnableCheckpoint(true);
          /*记录本地分片上传结果的文件。开启断点续传功能时需要设置此参数，上传过程中的进度信息会
保存在该文件中，如果某一分片上传失败，再次上传时会根据文件中记录的点继续上传。*/
     // 上传完成后，该文件会被删除。默认与待上传的本地文件同目录，为 uploadFile.ucp
            // request.setCheckpointFile("<yourCheckpointFile>");
            MultipleUpload multiUpload = new MultipleUpload(request, ossClient);

            // 断点续传上传
            CompleteMultipartUploadResult result = multiUpload.upload();
            //System.out.println("location:  "+result.getLocation());
            System.out.println("Upload complete. Upload object name:" + result.getKey());
        } catch (AmazonServiceException ase) {
            System.out.println("Caught an AmazonServiceException, which means your
request made it " + "to OOS, but was rejected with an error response for some reason.");
            System.out.println("Error Message:    " + ase.getMessage());
            System.out.println("HTTP Status Code: " + ase.getStatusCode());
            System.out.println("OOS Error Code:   " + ase.getErrorCode());
            System.out.println("Request ID:       " + ase.getRequestId());

        } catch (AmazonClientException ace) {
            System.out.println("Caught an AmazonClientException, which means the client
encountered "
                    + "a serious internal problem while trying to communicate with OOS, "
                    + "such as not being able to access the network.");
            System.out.println("Error Message: " + ace.getMessage());
        }
```

```
    }
```

4.4.13 POST Object

POST 操作使用 HTML 表单将文件上传到指定的 Bucket。POST 是另一种形式的 PUT 操作，POST 可以让使用者通过 Browser-based 的方式，将文件上传到指定 Bucket 中。PUT 的参数是通过 HTTP Header 提交的，而 POST 通过使用 multipart/form-data 编码的消息体中的字段进行提交。用户必须对操作的 Bucket 有写权限。OOS 不存储部分文件：如果收到成功的响应，那么文件就是存储成功了。

为了保证数据在网络传输过程中没有损坏，可以使用 Content-MD5 字段进行校验。如果请求参数中有 Content-MD5，OOS 将会计算用户提交的文件的 MD5 值。如果计算出的值与用户提供的值不一致，OOS 将会返回一个错误给用户。或者，用户可以在上传文件到 OOS 时计算文件的 MD5 值，并与 OOS 在响应中返回的 ETag 进行比较。ETag 是文件内容的 MD5 值，不包括 metadata。

- **请求语法**

```
POST /HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
User-Agent: browser_data
Accept: file_types
Accept-Language: Regions
Accept-Encoding: encoding
Accept-Charset: character_set
Keep-Alive: 300
Connection: keep-alive
Content-Type: multipart/form-data; boundary=9431149156168
Content-Length: length

--9431149156168
Content-Disposition: form-data; name="key"

Key
--9431149156168
Content-Disposition: form-data; name="success_action_redirect"

success_redirect
--9431149156168
Content-Disposition: form-data; name="Content-Type"

content_type
```

```
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-uuid"

uuid
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-tag"

metadata
--9431149156168
Content-Disposition: form-data; name="AWSAccessKeyId"

access-key-id
--9431149156168
Content-Disposition: form-data; name="Policy"

encoded_policy
--9431149156168
Content-Disposition: form-data; name="Signature"

signature=
--9431149156168
Content-Disposition: form-data; name="file"; filename="MyFilename.jpg"
Content-Type: image/jpeg

file_content
--9431149156168
Content-Disposition: form-data; name="submit"

Upload to OOS
--9431149156168–
```

- **请求参数**

本实现不使用请求参数。

- **表单字段**

说明：表单中指定的每个表单字段（AWSAccessKeyId、signature、x-amz-signature、file、policy 和带 x-ignore-前缀的字段名称除外）必须包含在 policy 条件列表中,两者需要保持一致。

名称	描述	是否必须
AWSAccessKeyId	根根用户或拥有权限的子用户的访问密钥 ID。如果请求包含 policy，对于 V2 签名，则此字段为必填字段。 类型：字符串	条件
Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires	特定于 REST 的请求头。有关更多信息，请参阅 PUT Object。 类型：字符串	否
file	文件或文本内容。文件或文本内容必须是 Form 表单的最后一个字段。一次只能上传一个文件。 类型：文件或文本内容	是
key	上传文件的名称。 ${*filename*}为用户提供的文件名。例如，如果用户 Betty 上传的文件名为 lolcatz.jpg，字段值指定为 /user/betty/${filename}，那么保存的文件名称将会是 /user/betty/lolcatz.jpg。 类型：字符串	是
policy	描述请求中允许的内容的安全策略。对于非匿名请求，policy 字段是必须的。 在 V2 或者 V4 签名计算中，policy 字段是您签名的字符串。 类型：字符串	条件
signature	使用 V2 签名计算的签名。如果请求包含 policy，对于 V2 签名，则此字段为必填字段。 signature = Base64(HMAC-SHA1(YourSecrectKey,StringToSign))。	条件

X-Amz-Algorithm	V4 签名算法。如果请求包含 policy，对于 V4 签名，则此字段为必填字段。 取值：AWS4-HMAC-SHA256。	条件
X-Amz-Credential	用户的 accessKeyId 和范围信息，范围信息包括请求日期、区域、服务、终止字符串 aws4_request，格式如下： <your-access-key-id>/<date>/<region>/<service>/aws4_request 其中： ● date 格式为 YYYYMMDD。 ● region： ■ 对于 oos api：访问域名为 oos-*xx*.ctyunapi.cn，region 为 *xx*。 ■ 对于统计 api：访问域名为 oos-*xx*-mg.ctyunapi.cn，region 为 *xx*-mg。 ■ 对于操作跟踪 api：访问域名为 oos-*xx*-cloudtrail.ctyunapi.cn，region 为 *xx*。 ■ 对于 iam api：访问域名为 oos-*xx*-iam.ctyunapi.cn，region 为 *xx*。 **说明**：各资源池的详细访问域名详见 **Endpoint 列表**。 ● service： ■ 若使用 OOS API 服务，service 为 s3。 ■ 若使用统计分析服务，service 为 s3。 ■ 若使用操作跟踪服务，service 为 cloudtrail。 ■ 若使用 IAM 服务，service 为 sts。 如果请求包含 policy，对于 V4 签名，则此字段为必填字段。	条件
X-Amz-Date	日期和时间格式必须遵循 ISO 8601 标准，并且必须使用“yyyyMMddT HHmmssZ”格式进行格式化。例如，如	条件

	果日期和时间是“08/01/2018 15：32：41.982-700”，则必须首先将其转换为 UTC（协调世界时），然后提交为“20180801T083241Z”。 如果请求包含 policy，对于 V4 签名，则此字段为必填字段。	
X-Amz-Signature	使用 V4 签名计算的签名。如果请求包含 policy，对于 V4 签名，则此字段为必填字段。 x-amz-signature=hex(HMAC-SHA256(SigningKey, StringToSign))	条件
x-amz-security-token	临时会话使用的安全令牌。使用临时密钥构造 V2 或者 V4 签名请求时，此字段为必填字段。	条件
success_action_redirect,redirect	上传成功后客户端重定向到的 URL。OOS 将 Bucket、文件名和 etag 值作为查询字符串参数附加到 URL。 如果未指定 success_action_redirect，OOS 将返回在 success_action_status 字段中指定的空文档类型。 如果 OOS 无法识别用户提供的 URL，将忽略该字段。 如果上传失败，OOS 将显示错误且不会执行重定向操作。 类型：字符串 **注意**：*redirect* 后续可能会被移除，建议使用 *success_action_redirect*	否
success_action_status	如果没有指定 success_action_redirect，上传成功后状态代码将返回到客户端。 有效值：200、201 或 204（默认）。 ● 如果值被设置为 200 或 204，OOS 将返回一个空文档和一个 200 或 204 状态代码。	否

	● 如果值被设置为 201，OOS 将返回一个 XML 文档和一个 201 状态代码。有关 XML 文档内容的信息，请参阅 POST 文件。 ● 如果没有设置值或者设置了无效的值，OOS 将返回一个空文档和一个 204 状态代码。 **注意：**某些版本的 AdobeFlashplayer 无法正确处理使用空白正文的 HTTP 响应。要通过 AdobeFlash 支持上传，建议您将 success_action_status 设置为 201。	
x-amz-storage-class	数据的存储类型。 类型：字符串 取值: ● STANDARD：标准存储。 ● STANDARD_IA：低频访问存储。 默认值为 STANDARD。	否
x-amz-meta-*	任何头以这个前缀开始都会被认为是用户的元数据，当用户检索时，它将会和文件一起被存储并返回。更多信息请参考 PUT Object 类型：字符串	否
x-amz-website-redirect-location	如果 Bucket 配置为网站，重定向对这个文件的请求到相同 Bucket 的另外一个文件或者到一个其他的 URL。OOS 会保存这个值到文件的 metadata。 下面这个例子表示请求头设置重定向到相同 Bucket 的另一个文件(anotherPage.html)。 x-amz-website-redirect-location: /anotherPage.html 重定向到其他网站的例子： x-amz-website-redirect-location: https://example-bucket.oos-cn.ctyunapi.cn。	否

	注意：这个值必须以”/”、https://或 http://开头，且长度不能超过 2KiB。	
x-ctyun-data-location	设置数据存储的位置。 **注意**：香港节点不支持此参数。 类型：key-value形式。 取值： 格式为：type=Local,scheduleStrategy=*scheduleStrategy*或者 type=Specified,location=*location*,scheduleStrategy=*scheduleStrategy* ● type：指定数据存储位置的类型，取值为 Local 或者 Specified。Local 表示就近写入，Specified 表示指定位置。如果 type 取值为 Specified，则需要指定具体的数据位置 location，location 可以填写多个，以逗号分隔，对于对象存储网络，可取值为：ChengDu、GuiYang、LaSa、LanZhou、QingDao、SH2、ShenYang、ShenZhen、SuZhou、WuHan、WuHu、WuLuMuQi、ZhengZhou；对于对象存储网络 2，可取值为：NeiMeng1、HangZhou1。 ● scheduleStrategy：调度策略，取值为： ➢ Allowed：允许 OOS 自动调度数据存储位置。 ➢ NotAllowed：不允许 OOS 自动调度数据存储位置。	否

- **响应头**

响应头（除了通用的响应头以外，本操作可以包括以下响应头。）

名称	描述

x-amz-expiration	如果文件设置了过期时间（参考 **PUT Bucket Lifecycle** 章节），响应头会增加过期信息。过期信息包括过期日期和 *rule-id* 的 *key* 和 *value*。rule-id 的 value 是经过 URL 编码的。 类型：字符串
success_action_redirect, redirect	上传成功重定向的 URL。 类型：字符串

- **响应体**

名称	描述
Bucket	存储 Object 的 Bucket 名称。 类型：字符串。 父元素：PostResponse
ETag	ETag 是文件内容经过 MD5 哈希后得到的值。用户可以在 GET 请求中使用 If-Match 请求头。ETag 仅反映文件内容的变化，不包括元数据。 类型：字符串 父元素：PostResponse
Key	文件名称。 类型：字符串 父元素：PostResponse
Location	文件的 URL。 类型：字符串 父元素: PostResponse

- **请求示例 1**

使用 V4 签名：

```
POST / HTTP/1.1
Content-Type: multipart/form-data; boundary=jOgLwKAuIgki-MzA2jxfztAlGcp-wO0rfdR;
charset=UTF-8
Host: a--12.oos-cn.ctyunapi.cn
```

```
Connection: Keep-Alive
User-Agent: Apache-HttpClient/4.5.7 (Java/1.8.0_351)
Content-Length: 1387

--jOgLwKAuIgki-MzA2jxfztAlGcp-wO0rfdR
Content-Disposition: form-data; name="x-amz-algorithm"

AWS4-HMAC-SHA256
--jOgLwKAuIgki-MzA2jxfztAlGcp-wO0rfdR
Content-Disposition: form-data; name="X-Amz-Credential"

25e67be754a9b2c85870/20240523/cn/s3/aws4_request
--jOgLwKAuIgki-MzA2jxfztAlGcp-wO0rfdR
Content-Disposition: form-data; name="x-amz-date"

20240523T092508Z
--jOgLwKAuIgki-MzA2jxfztAlGcp-wO0rfdR
Content-Disposition: form-data; name="key"

test01-post
--jOgLwKAuIgki-MzA2jxfztAlGcp-wO0rfdR
Content-Disposition: form-data; name="policy"

eyJleHBpcmF0aW9uIjoiMjAyNC0xMi0wMVQxMjowMDowMC4wMDBaIiwiY29uZGl0aW9ucyI6W3siYnVja2V0Ij
oiYS0tMTIifSx7ImtleSI6InRlc3QwMS1wb3N0In0seyJ4LWFtei1hbGdvcml0aG0iOiJBV1M0LUhNQUMtU0hB
MjU2In0seyJ4LWFtei1jcmVkZW50aWFsIjoiMjVlNjdiZTc1NGE5YjJjODU4NzAvMjAyNDA1MjMvY24vczMvYX
dzNF9yZXF1ZXN0In0seyJ4LWFtei1kYXRlIjoiMjAyNDA1MjNUMDkyNTA4WiJ9XX0=
--jOgLwKAuIgki-MzA2jxfztAlGcp-wO0rfdR
Content-Disposition: form-data; name="x-amz-signature"

91a935c2c409851f810cff5db53edb24415292edd651292de67432bc672169d7
--jOgLwKAuIgki-MzA2jxfztAlGcp-wO0rfdR
Content-Disposition: form-data; name="file"; filename="test01-post"
Content-Type: text/plain; charset=UTF-8

hello world!12345!@#$%^&*()_+":[]\?>,.adsf
--jOgLwKAuIgki-MzA2jxfztAlGcp-wO0rfdR
Content-Disposition: form-data; name="submit"

upload to oos
--jOgLwKAuIgki-MzA2jxfztAlGcp-wO0rfdR--
```

- **响应示例 1**

```
HTTP/1.1 204 No Content
ETag: "85c974a5ac9c67c64f55dba5d7c803a1"
Date: Thu, 23 May 2024 01:25:08 GMT
x-amz-request-id: c0e2170effd74bba0b7e71807582848a494b513f4143454749
Location: http://a--12.oos-cn.ctyunapi.cn/test01-post
Server: CTYUN
```

- **请求示例 2**

使用 V2 签名：

```
POST / HTTP/1.1
Content-Type: multipart/form-data; boundary=Ov8XWNyC3VX8y9lKZ4KPHJvpsN8TfDNqN5l;
charset=UTF-8
Host: a--12.oos-cn.ctyunapi.cn
Connection: Keep-Alive
User-Agent: Apache-HttpClient/4.5.7 (Java/1.8.0_351)
Accept-Encoding: gzip,deflate
Content-Length: 841

--Ov8XWNyC3VX8y9lKZ4KPHJvpsN8TfDNqN5l
Content-Disposition: form-data; name="AWSAccessKeyId"

25e67be754a9b2c85870
--Ov8XWNyC3VX8y9lKZ4KPHJvpsN8TfDNqN5l
Content-Disposition: form-data; name="key"

1.post
--Ov8XWNyC3VX8y9lKZ4KPHJvpsN8TfDNqN5l
Content-Disposition: form-data; name="policy"

eyJleHBpcmF0aW9uIjoiMjAyNS0xMS0zMVQxMjowMDowMC4wMDBaIiwiY29uZGl0aW9ucyI6W3siYnVja2V0Ij
oiYS0tMTIifSx7ImtleSI6IjEucG9zdCJ9XX0=
--Ov8XWNyC3VX8y9lKZ4KPHJvpsN8TfDNqN5l
Content-Disposition: form-data; name="signature"

R0y00Fj0FoIFFSuXQvo0i52Gd0Y=
--Ov8XWNyC3VX8y9lKZ4KPHJvpsN8TfDNqN5l
Content-Disposition: form-data; name="file"; filename="1.post"
Content-Type: image/jpeg; charset=UTF-8
```

```
123
--Ov8XWNyC3VX8y9lKZ4KPHJvpsN8TfDNqN5l
Content-Disposition: form-data; name="submit"

upload to oos
--Ov8XWNyC3VX8y9lKZ4KPHJvpsN8TfDNqN5l--
```

- **响应示例 2**

```
HTTP/1.1 204 No Content
ETag: "202cb962ac59075b964b07152d234b70"
Date: Thu, 23 May 2024 02:27:01 GMT
x-amz-request-id: 9d721961d2b14a5a0a7d707f74818389484a543e4042444648
Location: http://a--12.oos-cn.ctyunapi.cn/1.post
Server: CTYUN
```

- **说明**

表单声明包含三个部分：action、method 和 enctype。如果这些值当中的任意一个设置不正确，请求将失败。action 指定处理请求的 URL，必须将它设置为 Bucket 的 URL。例如：Bucket 的名称是 BucketName，则 URL 为 http://BucketName.oos-cn.ctyunapi.cn/。

method 必须是 POST。enctype 设置为“multipart/form-data”。

```
<form action="http:// BucketName.oos-cn.ctyunapi.cn /" method="post"enctype="multipart/form-data">
</form>
```

- **Post Policy 字段的构造**

Policy 是使用 UTF-8 和 Base64 编码的 JSON 文档，即需要对源码进行转换：先确保源码符合 UTF-8 编码格式，然后再进行 Base-64 编码。它指定了请求必须满足的条件并且用于对内容进行身份验证。根据您设计策略文档的方式，您可以对每次上传、每个用户、所有上传或根据其他能够满足您需要的设计来使用它们。

下面是一个简单的 Post Policy 示例：

1. Policy 源码由过期(expiration)和条件(conditions)两部分构成，且符合 UTF-8 编码格式（如果不符合 UTF-8 编码格式，需要转换成符合 UTF-8 编码格式的编码）：

```
{
    "expiration": "2007-12-01T12:00:00.000Z",
    "conditions": [
```

```
        {
            "bucket": "johnsmith"
        },
        [
            "starts-with",
            "$key",
            "user/eric/"
        ]
    ]
}
```

2. 对源码进行 Base64 编码。

```
ewogICAgImV4cGlyYXRpb24iOiAiMjAwNy0xMi0wMVQxMjowMDowMC4wMDBaIiwKICAgICJjb25kaXRpb25zIjog
WwogICAgICAgIHsKICAgICAgICAgICAgImJ1Y2tldCI6ICJqb2huc21pdGgiCiAgICAgICAgfSwKICAgICAgICBb
CiAgICAgICAgICAgICJzdGFydHMtd2l0aCIsCiAgICAgICAgICAgICIka2V5IiwKICAgICAgICAgICAgInVzZXIv
ZXJpYy8iCiAgICAgICAgXQogICAgXQp9
```

过期

过期元素采用 ISO 8601UTC 日期格式来指定策略的过期日期。例如，“2007-12-01T12:00:00.000Z”指定策略在 2007 年 12 月 1 日午夜 UTC 之后失效。在策略文档中过期字段是必需的。

条件

策略文档中的条件验证上传的文件的内容。表单中指定的每个表单字段（AWSAccessKeyId、signature、X-Amz-signature、file、policy 和带 x-ignore-前缀的字段名称除外）必须包含在 policy 条件列表中,两者需要保持一致。如果您有多个具有相同名称的字段，使用逗号进行分隔。例如，如果您有两个名为 x- amz-meta-tag 的字段，第一个字段的值为 Ninja，第二个字段的值为 Stallman，您可以将策略文档设置为 Ninja,Stallman。

针对扩展的字段执行前缀匹配。例如，如果您将文件名称字段设置为 user/betty/${filename}，您的策略可能是["starts-with", "$key", "user/betty/"]。请勿输入["starts-with", "$key", "user/betty/${filename}"]。

元素名称	说明
X-Amz-Algorithm	V4 签名算法。如果请求包含 policy，对于 V4 签名，则此字段为必填字段。 取值：AWS4-HMAC-SHA256。

X-Amz-Credential	用户的 accessKeyId 和范围信息，范围信息包括请求日期、区域、服务、终止字符串 aws4_request，格式如下： <your-access-key-id>/<date>/<region>/<service>/aws4_request 其中： ● date 格式为 YYYYMMDD。 ● region： ■ 对于 oos api：访问域名为 oos-*xx*.ctyunapi.cn，region 为 *xx*。 ■ 对于统计 api：访问域名为 oos-*xx*-mg.ctyunapi.cn，region 为 *xx*-mg。 ■ 对于操作跟踪 api：访问域名为 oos-*xx*-cloudtrail.ctyunapi.cn，region 为 *xx*。 ■ 对于 iam api：访问域名为 oos-*xx*-iam.ctyunapi.cn，region 为 *xx*。 **说明**：各资源池的详细访问域名详见 **Endpoint 列表**。 ● service： ■ 若使用 OOS API 服务，service 为 s3。 ■ 若使用统计分析服务，service 为 s3。 ■ 若使用操作跟踪服务，service 为 cloudtrail。 ■ 若使用 IAM 服务，service 为 sts。 如果请求包含 policy，对于 V4 签名，则此字段为必填字段。
X-Amz-Date	日期和时间格式必须遵循 ISO 8601 标准，并且必须使用“yyyyMMddT HHmmssZ”格式进行格式化。例如，如果日期和时间是“08/01/2018 15：32：41.982-700”，则必须首先将其转换为 UTC（协调世界时），然后提交为“20180801T083241Z”。 如果请求包含 policy，对于 V4 签名，则此字段为必填字段。

x-amz-security-token	临时会话使用的安全令牌。使用临时密钥构造 V2 或者 V4 签名请求时，此字段为必填字段。
x-amz-*	以 x-amz-开头的系统定义的元数据。（如 x-amz-website-redirect-location、x-amz-storage-class，不含 x-amz-signature）。 支持精确匹配。
bucket	指定上传内容允许的 Bucket。 支持精确匹配和 starts-with。
content-length-range	指定已上传内容允许长度的最小值和最大值。 支持范围匹配。
Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires	特定于 REST 的标头。 支持精确匹配和 starts-with。
Key	已上传文件的名称或文件名称前缀。 支持精确匹配和 starts-with。
success_action_redirect, redirect	上传成功后客户端重定向到的 URL。 支持精确匹配和 starts-with。
success_action_status	如果没有指定 success_action_redirect，上传成功后状态代码将返回到客户端。 支持精确匹配。
x-amz-meta- *	特定于用户的元数据。 支持精确匹配和 starts-with。

注意：如果您的工具包添加了其他字段（例如，Flash 添加了文件名），您必须将它们添加到策略文档。如果您可以控制此功能，将 x-ignore- 添加为字段的前缀以使 OOS 忽略此功能并使其不影响此功能的未来版本。

条件匹配

下表介绍条件匹配类型。尽管您必须为您在表单中指定的每个表单字段指定一个条件，您也可以通过为某个表单字段指定多个条件来创建更复杂的匹配条件。

条件	说明

精确匹配	精确匹配将验证字段是否匹配特定的值。此示例指示 Bucket 必须设置为 BucketName： `{"bucket": "BucketName"}` 也可写为： ["eq", "bucket": "BucketName"]
Starts With	如果值必须从某个特定的值开始，请使用 starts-with。本示例指示密钥必须从 user/betty 开始： `["starts-with", "$key", "user/betty/"]`
匹配任何内容	要配置策略以允许字段中的任何内容，请使用 starts-with 和一个空值。本示例允许任何 success_action_redirect： `["starts-with", "$success_action_redirect", ""]`
指定范围	对于接受范围的字段，请使用逗号来分隔上限和下限值。本示例允许 1 到 10 MiB 的文件大小： `["content-length-range", 1048579, 10485760]`

字符串转义

转义序列	描述
\\	反斜杠。
\$	美元符号。
\b	退格键。
\f	换页。
\n	新建行。
\r	回车。
\t	水平选项卡。
\v	垂直选项卡。
\uxxxx	所有 Unicode 字符。

Signature 字段的构造步骤

步骤	说明
1	使用 UTF-8 对 policy 内容进行编码。
2	使用 Base64 对步骤 1 中的 UTF-8 字节进行编码,得出签名字符串（StringTosign）。
3	使用 HMAC SHA-1 算法，对步骤 2 中的签名字符串和您的秘密访问密钥进行计算得出签名：HMAC-SHA1(YourSecretAccessKey,StringToSign)。
4	使用 Base64 对 SHA-1 签名进行编码： Signature=Base64(HMAC-SHA1(YourSecretAccessKey,StringToSign))

X-Amz-Signature 字段的构造步骤

步骤	说明
1	使用 UTF-8 对 policy 内容进行编码。
2	使用 Base64 对步骤 1 中的 UTF-8 字节进行编码,得出签名字符串（StringTosign）。

3	推算出签名用的密钥（SigningKey）。推算过程：
	`DateKey = HMAC-SHA256("AWS4"+"<SecretAccessKey>", "<yyyymmdd>")`
	`DateRegionKey = HMAC-SHA256(<DateKey>, "<oos-region>")`
	`DateRegionServiceKey = HMAC-SHA256(<DateRegionKey>, "<oos-service>")`
	`SigningKey = HMAC-SHA256(<DateRegionServiceKey>, "aws4_request")`

4	使用 HMAC SHA256 算法，对步骤 2 和步骤 3 中的签名字符串和签名密钥进行计算得出签名：HMAC-SHA256(SigningKey, StringToSign)。
5	使用十六进制编码对签名进行编码： X-Amz-Signature=hex(HMAC-SHA256(SigningKey, StringToSign))。

4.4.14 OPTIONS Object

浏览器可以向 OOS 发送预检请求，来判断其是否可以发送特定源、HTTP 方法和头的实际请求。当浏览器发送预检请求时，OOS 根据 Bucket 的跨域配置来返回响应信息。如果 Bucket 没有配置跨域，那么 OOS 返回响应 403 Forbidden。

- **请求语法**

```
OPTIONS /ObjectName HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Origin: Origin
Access-Control-Request-Method: HTTPMethod
Access-Control-Request-Headers: RequestHeader
```

- **请求头**

名称	描述	是否必须
Origin	标示访问OOS的跨域请求的源。 例如：https://docs.oos-cn.ctyun.cn。 类型：字符串。	是
Access-Control-Request-Method	标示在实际请求中，将被用到的HTTP方法。 类型：字符串	是
Access-Control-Request-Headers	在实际请求中将被发送的HTTP请求头，用逗号分隔。 类型：字符串	否

- **响应头**

响应头	描述
Access-Control-Allow-Origin	用户请求中发送的源。如果此源不被允许访问，那么OOS不会返回此响应头。 类型：字符串
Access-Control-Max-Age	预检请求可以被缓存的时间，单位是秒。 类型：字符串

Access-Control-Allow-Methods	用户请求时，允许发送的 HTTP 方法列表。如果用户请求的方法不被允许，那么 OOS 不会返回此响应头。 类型：字符串
Access-Control-Allow-Headers	在实际请求中，浏览器发送的 HTTP 请求头列表，以逗号分隔。如果浏览器未发送此请求头或者发送的此请求头内容为空，OOS 将不会返回此响应头。如果没有任何请求头被允许，OOS 将返回 403 状态码，也不会返回任何以 Access-Control 开头的响应头。 类型：字符串
Access-Control-Expose-Headers	在实际响应中，JavaScript客户端可以访问的响应头列表，以逗号分隔。 类型：字符串

- **请求示例**

浏览器可以向 OOS 发送预检请求，来判断其是否可以从源 https://docs.oos-cn.ctyun.cn 向名为 example-bucket 的 Bucket，发送 PUT 请求。

```
OPTIONS /exampleobject HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
Origin: https://docs.oos-cn.ctyun.cn
Access-Control-Request-Method: PUT
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id: 0e66553608684ccf0b7e71807582848a494b543f4143454749
Date: Wed, 21 Aug 2012 23:09:55 GMT
Access-Control-Allow-Origin: https://docs.oos-cn.ctyun.cn
Access-Control-Allow-Methods: PUT
Access-Control-Expose-Headers: x-amz-request-id
Server: CTYUN
```

4.4.15 生成共享链接

- **生成共享链接**

可以通过生成 Object 的共享链接的方式，将 Object 分享给其他人，同时可以在链接中设置限速以对下载速度进行控制。在 SDK 中调用 AmazonS3 中的 generatePresignedUrl (GeneratePresignedUrlRequest)方法生成共享链接， GeneratePresignedUrlRequest 的参数包含了 BucketName，ObjectName，并且可以设置过期时间和下载速度；如果过期时间传 Null 的话，默认的过期时间是 15 分钟。超出过期时间后，共享链接失效，不能再通过链接下载 Object。

生成共享链接示例：

```
public static void generatePresignedUrl(AmazonS3 oosClient) {
       GeneratePresignedUrlRequest shareUrlRequest = new GeneratePresignedUrlRequest(
              BUCKET_NAME, OBJECT_NAME);
       java.util.Date now = new java.util.Date();
       java.util.Date expire = new Date(now.getTime() + 24 * 3600 * 1000);
//24*3600*1000
       shareUrlRequest.setExpiration(expire);
    //shareUrlRequest.addRequestParameter("x-amz-limitrate", "2048");
       URL url1 = oosClient.generatePresignedUrl(shareUrlRequest);
       System.out.println(url1.toString());
   }
```

以下是一个生成的共享链接：

```
http://oos-cn.ctyunapi.cn/test-
20180604/6aa3df83gw1f35nhhp70pj20gj0r046l.jpg?Signature=8l7F/pabWm2%2Bi8iXyExZIXm/eGY%
3D&AWSAccessKeyId=08f17977afa1a87736ac&Expires=1363760719
```

- **共享链接限速**

如果需要为链接设置下载速度限制，需要新增加自定义参数“x-amz-limitrate”，调用 GeneratePresignedUrlRequest.addRequestParameter("x-amz-limitrate", value)方法，value 值为限速带宽(单位 KiB/s)，将参数加到 GeneratePresignedUrlRequest 文件中，参与共享链接生成，以下为增加了下载速度限制生成的共享链接的示例：

```
http://oos-cn.ctyunapi.cn/test-
20180604/6aa3df83gw1f35nhhp70pj20gj0r046l.jpg?Signature=8l7F/pabWm2%2Bi8iXyExZIXm/eGY%
3D&AWSAccessKeyId=08f17977afa1a87736ac&Expires=1528438576&x-amz-limitrate=2048
```

4.4.16 HEAD Object

此操作用于获取文件的元数据信息，而不返回数据本身。当只希望获取文件的属性信息时，可以使用此操作。

- **请求语法**

```
HEAD /ObjectName HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Date: date
Authorization: SignatureValue
```

- **响应头**

变量	描述
x-amz-expiration	如果文件被配置了到期时间，那么 OOS 返回此响应头。这个响应头包含键值对 *expiry-date* 和 *rule-id*。*rule-id* 的值是 URL 编码的。
x-amz-storage-class	数据存储类型。如果存储类型为 STANDARD，不返回此项。 类型：字符串 取值：STANDARD_IA：低频访问存储。
x-ctyun-metadata-location	文件的索引位置。 **注意**：香港节点不会返回此项。 类型：枚举 取值：对于对象存储网络，取值为：ChengDu、FuZhou、GuiYang、HangZhou、LaSa、LanZhou、QingDao、ShenYang、ShenZhen、WuHan、WuHu、WuLuMuQi、ZhengZhou、SH2、SuZhou；对于对象存储网络 2，取值为：NeiMeng1、HangZhou1。
x-ctyun-data-location	获取文件的数据位置。 **注意**：香港节点不会返回此项。 类型：枚举 取值：对于对象存储网络，取值为：ChengDu、GuiYang、LaSa、LanZhou、QingDao、SH2、ShenYang、ShenZhen、SuZhou、

	WuHan、WuHu、WuLuMuQi、ZhengZhou；对于对象存储网络2，取值为：NeiMeng1、HangZhou1。
x-amz-meta-*	以该前缀开头的用户定义的元数据响应头。每一个都作为一组键值对存储和返回。OOS 不验证或解释用户定义的元数据。 类型：字符串。
x-ctyun-last-access-time	Object 的最后一次访问时间。 如果为 Object 开启了匹配最后一次访问时间的生命周期规则，会返回该响应头，否则不返回该响应头。 **注意**：对于一天内多次访问同一个 Object（GET 操作），仅记录该 Object 的该天最早一次访问时间。

- **请求示例**

```
HEAD /ObjectName HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
Date: Mon, 15 Nov 2021 06:39:41 GMT
Content-Type: application/octet-stream
Connection: Keep-Alive
Authorization: SignatureValue
```

- **响应示例**

```
HTTP/1.1 200 OK
Content-Length: 1467326
ETag: "5db44ee68a1e577907c2699c8f582107"
Last-Modified: Mon, 15 Nov 2021 06:39:39 GMT
Content-Type: application/x-java-archive
Content-MD5: XbRO5ooeV3kHwmmcj1ghBw==
x-ctyun-metadata-location: ChengDu
x-ctyun-data-location: ZhengZhou
Date: Mon, 15 Nov 2021 06:39:41 GMT
x-amz-request-id: 4ae1a5d637614e3d137f838a928a8c9251535747494b4d4f51
Server: CTYUN
```

4.5 Backoff 说明

当 OOS 系统服务繁忙，暂时不可使用时，OOS 会向客户端返回 503 响应码和 Retry-After 响应头。示例如下：

```
HTTP/1.1 503 Service Unavailable
x-amz-request-id: 767bc363031946fa81fbfdf4f6fcbbbdc0b1b3b5b7b9bbbd21
Retry-After: 90
Date: Tue, 6 May 2014 08:02:58 GMT
Content-Length: 133
Content-Type: text/xml
Server: CTYUN

<?xml version='1.0' encoding='UTF-8'?>
<Error>
    <Code>SlowDown</Code>
    <Message>Please reduce your request rate.</Message>
</Error>
```

4.6 错误响应

4.6.1 REST 错误响应

当请求出错时，OOS 返回以下格式的错误响应信息。

```
<?xml version="1.0" encoding="UTF-8"?>
<Error>
    <Code>NoSuchKey</Code>
    <Message>The resource you requested does not exist.</Message>
    <Resource>/mybucket/myfoto.jpg</Resource>
    <RequestId>4442587FB7D0A2F9</RequestId>
</Error>
```

响应结果说明如下。

名称	描述
Code	错误码是描述出错信息的字符串。
Message	错误信息的简单描述。
RequestId	请求的 ID。
Resource	出错相关的 Bucket 或 Object 信息。

4.6.2 错误码列表

状态码	错误码	错误信息	描述
400	AuthorizationHeaderMalformed	The authorization header is malformed.	签名头格式不正确。
400	AuthorizationHeaderMalformed	The authorization header is malformed, missing Credential.	签名头格式不正确，需补充 Credential。
400	AuthorizationHeaderMalformed	The authorization header is malformed, missing SignedHeaders.	签名头格式不正确，需补充 SignedHeaders。
400	AuthorizationHeaderMalformed	The authorization header is malformed, missing Signature.	签名头格式不正确，需补充 Signature。
400	AuthorizationHeaderMalformed	The authorization header is malformed, the Credential is mal-formed, expecting 'YOUR-AKID/YYYYMMDD/REGION/SERVICE/aws4_request'.	签名头中 Credential 格式不正确，格式应该为 'YOUR-AKID/YYYYMMDD/REGION/SERVICE/aws4_request'。
400	AuthorizationHeaderMalformed	The authorization header is malformed, the date format '*date*' is wrong, expecting 'YYYYMMDD'.	签名头中 date 格式不正确，应该为 'YYYYMMDD'。
400	AuthorizationHeaderMalformed	The authorization header is malformed, the region '*region*' is wrong, expecting '*region*'.	签名头中区域名不正确。

400	AuthorizationHeaderMalformed	The authorization header is malformed, incorrect service '*service*'. This endpoint belongs to '*endpoint*'.	签名头不正确，服务域名不对。
400	AuthorizationHeaderMalformed	The authorization header is malformed, incorrect terminal 'terminal'. This endpoint uses 'aws4_request'.	签名头不正确，endpoint 应使用服务 aws4_request。
400	AuthorizationPostFormFieldsError	Form Post authentication version 4 requires the policy, x-amz-algorithm, x-amz-credential, x-amz-signature, x-amz-date form fields.	V4 签名中 policy、x-amz-algorithm、x-amz-credential、x-amz-signature、x-amz-date 不能为空。
400	AuthorizationPostFormFieldsError	X-amz-algorithm only supports 'AWS4-HMAC-SHA256'.	v4 签名算法仅支持 AWS4-HMAC-SHA256。
400	AuthorizationPostFormFieldsError	X-amz-date must be in the ISO8601 Long Format "yyyyMMdd'T'HHmmss'Z'".	x-amz-date 必须是 ISO8601 时间格式 yyyyMMdd'T'HHmmss'Z'。
400	AuthorizationQueryParametersError	X-Amz-Algorithm only supports 'AWS4-HMAC-SHA256'.	V4 预签名签名算法仅支持 AWS4-HMAC-SHA256。
400	AuthorizationQueryParametersError	Query-string authentication version 4 requires the X-Amz-Algorithm, X-Amz-Credential, X-Amz-Signature, X-Amz-Date, X-Amz-SignedHeaders, and X-Amz-Expires parameters.	V4 预签名需要下列参数：X-Amz-Algorithm, X-Amz-Credential, X-Amz-Signature, X-Amz-Date, X-Amz-

			SignedHeaders, and X-Amz-Expires。
400	AuthorizationQueryParametersError	X-Amz-Expires must be less than a week (in seconds); that is, the given X-Amz-Expires must be less than 604800 seconds.	V4 预签名生成时间不能超过 7 天。
400	AuthorizationQueryParametersError	X-Amz-Expires must be non-negative.	V4 预签名过期时间不能为负数。
400	AuthorizationQueryParametersError	X-Amz-Date must be in the ISO8601 Long Format "yyyyMMdd'T'HHmmss'Z'".	V4 预签名 X-Amz-Date 必须是 ISO8601 时间格式 yyyyMMdd'T'HHmmss'Z'。
400	AuthorizationQueryParametersError	Invalid credential date "*date*". This date is not the same as X-Amz-Date: "*date*".	V4 预签名 credential date 与 X-Amz-Date 的日期需要保持一致。
400	BadDigest	The Content-MD5 you specified did not match what we received.	Content-MD5 不正确。
400	BadRequest	Advance cut's position is out of image.	剪切图片时起始坐标超过原图片尺寸。
400	CanNotModifyMetadataLocation	Can not modifiy bucket metadata location.	不能修改元数据位置。
400	IdMismatch	Document ID does not match the specified configuration ID.	请求头和请求体中清单 ID 输入不一致。

400	IncompleteBody	You did not provide the number of bytes specified by the Content-Length HTTP header.	Content-Length 配置的长度与实际收到的数据长度不符，请检查后重发。
400	InvalidArgument	User key must have a length greater than 0.	表单上传文件时 Key 项的值不能为空。
400	InvalidArgument	The authorization header is invalid. Expected AccessKeyId:signature.	v2 签名格式不正确，应该为“*AccessKeyId*:*signature*”。
400	InvalidArgument	x-amz-content-sha256 must be UNSIGNED-PAYLOAD, STREAMING-AWS4-HMAC-SHA256-PAYLOAD, or a valid sha256 value.	v4 签名 x-amz-content-sha256 必须是 UNSIGNED-PAYLOAD, STREAMING-AWS4-HMAC-SHA256-PAYLOAD 或者有效的 sha256 值。
400	InvalidArgument	Unsupported authorization type.	v4 签名 Authentication 标头只支持 AWS4-HMAC-SHA256 算法。

400	InvalidArgument	Bucket POST must contain a field named 'AWSAccessKeyId'. If it is specified, please check the order of the fields.	必须包含字段 AWSAccessKeyId 。
400	InvalidArgument	Bucket POST must contain a field named 'policy'. If it is specified, please check the order of the fields.	必须包含字段 policy。
400	InvalidArgument	Bucket POST must contain a field named 'signature'. If it is specified, please check the order of the fields.	必须包含字段 signature。
400	InvalidArgument	POST requires exactly one file upload per request.	POST 请求每次只能上传一个文件。
400	InvalidArgument	Bucket POST must contain a field named 'key'. If it is specified, please check the order of the fields.	表单上传中必须指定 key。如果已指定，请检查字段顺序。
400	InvalidArgument	If the TargetBucket element does not exist, the TargetPrefix element cannot exist either.	在配置日志规则时，如果 TargetBucket 元素不存在，则 TargetPrefix 也不能存在。
400	InvalidArgument	The target prefix exceeds 900 bytes.	配置日志记录规则时，TargetPrefix 的长度不能超过 900 字节。

400	InvalidArgument	ID length should not exceed allowed limit of 255.	配置生命周期规则时，Rule ID 长度不能超过 255。
400	InvalidArgument	At least one action needs to be specified in a rule.	配置生命周期规则时，必须配置过期时间，Days 或 Date。
400	InvalidArgument	'Days' for [Transition|Expiration] action must be a positive integer.	配置生命周期规则时，Days 必须为正整数。
400	InvalidArgument	Only one action can be specified in the rule.	配置生命周期规则时，Days 或 Date 只能存在一个。
400	InvalidArgument	'Date' must be at midnight GMT.	配置生命周期规则时，Date 必须为 ISO8601 格式，并且为 UTC 的零点。
400	InvalidArgument	Invalid target Storage class for Transition action.	指定的存储类型无效。
400	InvalidArgument	Default retention period must be a positive integer value.	合规保留日期取值必须是正整数。
400	InvalidArgument	Object lock configuration has been enabled and can not be disabled.	合规保留已经开启，不能关闭。
400	InvalidArgument	Default retention period cannot be less than previously value.	设置的合规保留时长不能短于上次设置的时长。

400	InvalidArgument	MaxUploads should be between 1~1000.	max-uploads 的取值范围为[1,1000]。
400	InvalidArgument	The length of delimiter must be 1.	delimiter 长度不能超过 1。并且只能是‘/’或空。
400	InvalidArgument	MaxKeys should be between 1~1000.	参数 maxKeys 取值范围为[1,1000]。
400	InvalidArgument	Max-parts should be between 1~1000.	max-parts 取值范围为[1,1000]。
400	InvalidArgument	Part-number-marker must be an integer between 0 and 2147483647.	part-number-marker 必须是整数，取值范围是(0, 2147483647)。
400	InvalidArgument	x-amz-limitrate invalid. The input should be positive integer.	x-amz-limitrate 取值必须为正整数。
400	InvalidArgument	Invalid request parameter: value.	参数无效。
400	InvalidArgument	The object size should be less than 5TiB.	文件大小不能超过 5TiB。
400	InvalidArgument	The 'Expires' field format invalid. The 'Expires' header format invalid.	Expires 格式不正确。
400	InvalidArgument	The metadata size should be less than 2KiB.	用户自定义元数据不能超过 2KiB。
400	InvalidArgument	Insufficient information. Origin request header needed.	跨域预检时，请求头不正确。
400	InvalidArgument	Please enter an integer not less than -1.	CORS 配置中 MaxAgeSeconds 取

			值为不小于-1 的整数。
400	InvalidArgument	Invalid Access-Control-Request-Method: *value*.	Access-Control-Request-Method 无效。
400	InvalidArgument	The x-amz-copy-source-range value must be of the form bytes=first-last where first and last are the zero-based offsets of the first and last bytes to copy.	x-amz-copy-source-range 形式必须为 bytes=*first-last*，其中第一个和最后一个都是以零为基础开始的偏移量。
400	InvalidArgument	Invalid protocol, protocol can be http or https. If not defined the protocol will be selected automatically.	指定的 http 协议值不正确。
400	InvalidArgument	The provided host name contains *value* characters. The maximum allowed redirect page size is 1024 characters.	HostName 包含的字符个数大于 1024。
400	InvalidArgument	The provided home page contains *value* characters. The maximum allowed home page size is 1024 characters.	首页包含的字符个数大于 1024。
400	InvalidArgument	The provided error page contains *value* characters. The maximum allowed error page size is 1024 characters.	错误页包含的字符个数大于 1024。
400	InvalidArgument	The provided replacement contains *value* characters. The maximum allowed replacement size is 1024 characters.	重定向规则中 ReplaceKeyPrefixWith 或者 ReplaceKeyWith 包

			含的字符个数大于1024。
400	InvalidArgument	The provided prefix contains *value* characters. The maximum allowed prefix size is 1024 characters.	重定向规则中KeyPrefixEquals 包含的字符个数大于1024。
400	InvalidArgument	RedirectAllRequestsTo cannot be provided in conjunction with other Routing Rules.	重定向所有请求不能与其他规则同时存在。
400	InvalidArgument	The IndexDocument Suffix cannot contain the "/" character.	索引页配置不能包含斜杠“/”。
400	InvalidArgument	The IndexDocument Suffix cannot be blank.	索引页配置不能为空。
400	InvalidArgument	Missing required key 'Redirect'.	缺少必需的参数Redirect。
400	InvalidArgument	The ErrorDocument Key cannot be blank.	错误页配置不能为空。
400	InvalidArgument	The HostName cannot be blank.	HostName 配置不能为空。
400	InvalidArgument	The HostName cannot contain the "/" character.	HostName 不能包含斜杠“/”。
400	InvalidArgument	The provided HTTP error code (*value*) is not valid. Valid codes are 4XX or 5XX.	重定向条件中的HttpErrorCodeReturnedEquals 不是有效值。
400	InvalidArgument	Missing required key 'HostName'.	缺少必需的参数HostName。

400	InvalidArgument	The HostName cannot contain spaces.	HostName 配置不能包含空格。
400	InvalidArgument	A value for IndexDocument Suffix must be provided if RedirectAllRequestsTo is empty.	重定向所有请求未配置时，索引文件必须配置。
400	InvalidArgument	The destination bucket does not belong to you.	目的 Bucket 为他人的 Bucket。
400	InvalidArgument	Invalid Format value.	清单的格式不正确，取值只能为 CSV。
400	InvalidArgument	Invalid Field value.	Field 值填写错误。
400	InvalidArgument	The maximum size of a Destination/Prefix is 512.	Prefix 最大长度为 512。
400	InvalidArgument	The maximum size of a Filter/Prefix is 1024.	Prefix 最大长度为 1024。
400	InvalidArgument	IsEnabled value expects true or false.	IsEnabled 的取值只能为 true 或 false。
400	InvalidArgument	Frequency value expects Daily or Weekly.	Frequency 的取值只能为 Daily 或 Weekly。
400	InvalidArgument	Schedule/Frequency missing.	Frequency 未填写。
400	InvalidArgument	InventoryConfiguration/Destination/OOSBucketDestination/Bucket missing.	清单的目的 Bucket 未填写。
400	InvalidArgument	IsEnabled missing.	IsEnabled 缺失。
400	InvalidArgument	InventoryConfiguration/Destination/OOSBucketDestination/Format missing.	清单的输出格式缺失。
400	InvalidArgument	Invalid id.	Id 无效。

400	InvalidArgument	Id missing.	请求体中的 id 缺失。
400	InvalidArgument	The type of 'IsAccessTime' must be Boolean.	IsAccessTime 取值应为布尔型，true 或者 false。
400	InvalidArgument	'Days' in the Expiration action for prefix '*prefix*' must be greater than 'Days' in the Transition action.	两生命周期规则重叠时，过期删除天数必须大于过期转储天数。
400	InvalidBucketCorsConfigure	The XML you provided is not valid.	XML 无效。
400	InvalidBucketName	The bucket name is: *name*.	Bucket 名字无效。
400	InvalidConfigurationId	The specified configuration id is invalid.	请求头的清单 ID 未填写。
400	InvalidLocationConstraint	The specified location constraint is not valid.	指定的数据位置无效。
400	InvalidObjectName	The object name is: *name*.	文件名字无效。
400	InvalidOOSDestinationBucket	Invalid Bucket ARN.	Bucket ARN 的格式不正确。
400	InvalidOOSDestinationBucket	Invalid Bucket Name: *bucketname*.	清单输出的目的 Bucket 不存在。
400	InvalidPart	One or more of the specified parts could not be found. The part may not have been uploaded, or the specified entity tag may not match the part's entity tag.	一个或者多个指定片段无法找到，片段可能没有被上传，或者指定的 ETag 值跟片段的 ETag 值不匹配。

400	InvalidPartNumber	Part number must be an integer between 1 and 10000.	参数 PartNumber 的取值是[1, 10000]。
400	InvalidPartOrder	The list of parts was not in ascending order. The parts list must be specified in order by part number.	分片片段列表没有按升序排列。片段列表必须根据分片号按顺序排列。
400	InvalidPartSize	Your proposed upload is smaller than the minimum allowed object size. Each part must be at least 5 MiB in size, except the last part.	合并分片文件时，只允许最后一片小于 5MiB，其他分片至少 5MiB。
400	InvalidPolicyDocument	Invalid Policy: JSON invalid: *value*.	策略无效，不是标准的 json 格式。
400	InvalidPolicyDocument	Invalid according to Policy: Extra input fields: *value*.	策略无效，输入了额外的字段。
400	InvalidPolicyDocument	Policy could not be parsed as a valid JSON string.	策略必须是 json 格式。
400	InvalidPolicyDocument	Policy is missing required element - Statement.	策略必须配置 Statement。
400	InvalidPolicyDocument	Policy is missing required element - Principal.	策略必须配置 Principal。
400	InvalidPolicyDocument	Policy element has invalid value - Principal.	策略配置的 Principal 值无效。
400	InvalidPolicyDocument	Policy is missing required element - Resource.	策略必须配置 Resource 元素。
400	InvalidPolicyDocument	Policy is missing required element - Resource content.	策略的 Resource 元素必须赋值。

400	InvalidPolicyDocument	Policy has invalid resource - *resource*.	策略配置的 Resource 无效。
400	InvalidPolicyDocument	Policy has an invalid condition key - *key*.	策略配置的 condition key 无效。
400	InvalidPolicyDocument	No such condition type - type.	策略配置的 condition type 无效。
400	InvalidPolicyDocument	Action does not apply to any resource(s) in statement.	策略配置的 Action 无效。
400	InvalidPolicyDocument	Policy is missing required element - Effect.	策略必须配置 Effect。
400	InvalidPolicyDocument	Policy element has invalid value - Effect : *Effect*.	策略配置的 Effect 无效。
400	InvalidPolicyDocument	Policy element has invalid value - Version : *Version*.	策略配置的 Version 无效。
400	InvalidRedirectLocation	The website redirect location must be non-empty.	x-amz-website-redirect-location 头不能为空。
400	InvalidRedirectLocation	The website redirect location must have a prefix of 'http://' or 'https://' or '/'.	x-amz-website-redirect-location 必须有前缀'http://'、'https://'或'/'.。
400	InvalidRedirectLocation	The length of website redirect location cannot exceed 2,048 characters.	x-amz-website-redirect-location 长度不能超过 2048。

400	InvalidRequest	Missing required header for this request: x-amz-content-sha256.	v4 签名必须携带 x-amz-content-sha256。
400	InvalidRequest	Missing required header for this request: Content-MD5.	请求头必须包含 Content-MD5。
400	InvalidRequest	Please reduce your request rate.	请降低您的请求频率。
400	InvalidRequest	The operation failed, please change a resource pool and try again.	请换其他资源池或者重试。
400	InvalidRequest	This copy request is illegal because it is trying to copy an object to itself without changing the object's metadata or storage class.	复制文件时必须修改元数据或者存储类型。
400	InvalidRequest	x-amz-limitrate invalid.The input should be positive integer.	x-amz-limitrate 只能是正整数。
400	InvalidRequest	The source object size cannot be smaller than 5 GiB.	如果源文件取 range 进行拷贝，源文件大小不能小于 5G。
400	InvalidRequest	You can only define ReplaceKeyPrefix or ReplaceKey but not both.	ReplaceKeyPrefix 或者 ReplaceKey 不能同时存在。
400	InvalidRequest	Condition cannot be empty. To redirect all requests without a condition, the condition element shouldn't be present.	Condition 配置不能为空。不带条件的情况下，重定向所有请求不应该出现 condition 元素。

400	InvalidRequest	The maximum size of a prefix is 1024.	配置生命周期规则时，前缀的最大长度为 1024。
400	InvalidRequest	Number of lifecycle rules should not exceed allowed limit of 1000 rules.	同一 Bucket 配置的生命周期规则不能超过 1000 条。
400	InvalidRequest	Rule ID must be unique. Found same ID for more than one rule.	配置生命周期规则时，Rule ID 必须唯一。
400	InvalidRequest	Found two rules with same prefix '*prefix*' for same action type [Transition|Expiration].	同一 Bucket 的两条生命周期规则不能有相同的 prefix。
400	InvalidRequest	Found overlapping prefixes '*prefix*' and '*prefix*' for same action type [Transition|Expiration].	同一 Bucket 的两条生命周期规则不能有前缀重叠的情况。
400	InvalidRequest	Found overlapping prefixes '*prefix*' and '' for same action type [Transition|Expiration].	同一 Bucket 内的不同生命周期规则不能同时存在有前缀和无前缀规则。
400	InvalidStorageClass	The storage class you specified is not valid.	x-amz-storage-class 无效。
400	InvalidStorageClass	Can not specify the storage class.	未合并的分段文件不能指定 x-amz-storage-class 进行拷贝。
400	InvalidTargetBucketForLogging	The target bucket name is: name.	日志记录规则时，目标 Bucket 不存在

			或者无目标 Bucket 的权限。
400	MalformedPOSTRequest	The body of your POST request is not well-formed multipart/form-data.	POST 表单请求中 content-type 配置的 boundary 格式不规范。
400	MalformedXML	CORS Rule is empty.	CORS Rule 不能为空。
400	MalformedXML	The number of CORS Rules you provided cannot exceed 100.	CORS Rules 个数不能超过 100。
400	MalformedXML	The length of CORS Rule ID you provided cannot exceed 255 characters.	CORS Rule ID 长度不能超 255 个字符。
400	MalformedXML	AllowedOrigin does not exist.	CORS 配置中 AllowedOrigin 不存在。
400	MalformedXML	AllowedMethod does not exist.	CORS 配置中 AllowedMethod 不存在。
400	MalformedXML	AllowedOrigin "*value*" can not have more than one wildcard.	CORS 配置中 AllowedOrigin 最多只能携带一个通配符。
400	MalformedXML	AllowedHeader "*value*" can not have more than one wildcard.	CORS 配置中 AllowedHeader 最多只能携带一个通配符。

400	MalformedXML	ExposeHeader "*value*" contains wildcard. We currently do not support wildcard for ExposeHeader.	CORS 配置中 ExposeHeader 不能携带通配符。
400	MalformedXML	CORS Rule ID is not unique.	CORS Rule ID 必须是唯一的。
400	MalformedXML	The XML you provided was not well-formed or did not validate against our published schema.	XML 格式不正确。
400	MaxRequestBodyLengthExceeded	Your request was too big, the maximum size is %*s*.	请求体长度太长。
400	MissingRequestBodyError	Request Body is empty.	请求 body 体为空。
400	NoSuchUpload	The specified upload does not exist. The uploadId may be invalid, or the upload may have been aborted or completed.	指定的分片上传过程不存在，上传 ID 可能非法，分片上传过程可能被终止或者已经完成。
400	PutObjectTooFast	Upload: *value* is updated too fast.	同一文件并发上传过快。
400	TooManyBuckets	You have attempted to create more buckets than allowed.	Bucket 数量已经到达上限。
400	TooManyConfigurations	You are attempting to create a new configuration but have already reached the 10-configuration limit.	超出清单配置规则的数量限制，每个 Bucket 最多配置 10 条清单。
403	InvalidAccessKeyId	InvalidAccessKeyId.	AccessKeyId 无效。

403	InvalidAccessKeyId	The AccessKeyId is invalid.	AccessKeyId 被禁用。
403	InvalidAccessKeyId	The specified accessKey: *accessKey* does not exist.	临时密钥不存在，需重新申请。
403	AccessDenied	Please use primary access key.	需要使用主密钥。
403	AccessDenied	Non-primary key can not list without prefix or prefix is not start with access key.	非主密钥只能 list 以自己 access key 为前缀的文件。
403	AccessDenied	Non-primary key can not operate bucket.	非主密钥，不能操作该 Bucket。
403	AccessDenied	Non-primary key can not operate object that not start with access key.	非主密钥只能操作以自己 access key 为前缀的文件。
403	AccessDenied	The secret token is expired, expiration: *value*.	secret token 过期。
403	AccessDenied	The OOS service has been closed due to overdue payment. If the overdue payment exceed 15 days, the data will be deleted. Please recharge it in time.	由于逾期付款，OOS 服务已关闭。如果逾期付款超过 15 天，数据将被删除。请及时充值。
403	AccessDenied	OOS authentication requires a valid Date or x-amz-date header.	OOS 签名头中需要携带 Date 或 x-amz-date。
403	AccessDenied	OOS authentication requires a valid Date or x-amz-date header, expecting \" YYYYMMDD'T'HHMMSS'Z'\".	OOS 签名头需要携带正确格式的 Date 或 x-amz-date，格式为

			YYYYMMDD'T'HHMMSS'Z'。
403	AccessDenied	Invalid according to Policy: Policy expired.	v4 签名的 x-amz-date 时间不能超过 7 天。
403	AccessDenied	Invalid according to Policy: Policy Condition failed:xxx	签名策略不对。
403	AccessDenied	Query-string authentication requires the Signature, Expires and AWSAccessKeyId parameters.	V2 预签名需要携带以下参数：Signature、Expires、AWSAccessKeyId。
403	AccessDenied	Invalid date (should be seconds since epoch): *date*.	过期时间需要精确到秒。
403	AccessDenied	Request has expired.	请求过期。
403	AccessDenied	Anonymous access is forbidden for this operation.	匿名用户禁止此操作。
403	AccessDenied	The user's permission is not enough.	用户权限不足。
403	AccessDenied	You do not have permission to this resource pool, please try another resource pool.	没有该资源池的权限，请使用其他资源池。
403	AccessDenied	CORSResponse: This CORS request is not allowed. This is usually because the evalution of Origin, request method / Access-Control-Request-Method or Access-Control-Request-Headers are	跨域预检请求没有通过 CORS 规则校验。请检查 Origin、Access-Control-Request-Method、Access-

		not whitelisted by the resource's CORS spec.	Control-Request-Headers 是否在 CORS 列表的白名单中。
403	AccessDenied	CORSResponse: CORS is not enabled for this bucket.	此 Bucket 未配置 CORS。
403	AccessDenied	Access denied by bucket policy.	受 Bucket Policy 的规则影响，被显示拒绝。
403	AccessDenied	Access denied by bucket policy or anonymous access is forbidden.	匿名访问被拒绝，或者被 Bucket Policy 规则隐式拒绝。
403	AccessDenied	User: *user* is not authorized to perform: *action* on resource: *resource*.	没有权限访问此资源。
403	AccessDenied	The user does not belong to the region.	没有此资源池的上传权限，请更换其他资源池重试。
403	AccessDenied	You do not have permission to request the value data region. Please change the data region for the bucket and try again.	Bucket 下配置的数据位置都已失效，请重新配置，或者修改为可以自动调度。
403	AccessDenied	You are not allowed to set the public-read-write permission for the bucket.	您不具备将容器设置为公有的权限，请联系天翼云客服协助开通此功能。

403	AccessDenied	You are not allowed to grant public-write permission to anonymous users.	您不可以向匿名用户授予写权限，请联系天翼云客服协助开通此功能。
403	AccessDenied	Make sure that bucket's permissions are not private before using the static website hosting feature.	在使用静态网站托管功能之前，请确保 Bucket 权限不是私有。
403	Forbidden	CORSResponse: Bucket not found.	如果是 pre-flight CORS 请求，Bucket 必须存在。
403	IllegalObject	The object "*bucket/object*" is illegal, forbid copy.	源文件是非法文件，不允许拷贝。
403	RequestTimeTooSkewed	The difference between the request time and the server's time is too large.	客户端时间和服务器端时间相差超过 15 分钟。
403	SignatureDoesNotMatch	The request signature we calculated does not match the signature you provided. Check your key and signing method.	签名计算错误，需按 OOS 的签名规则，请检查客户端签名计算方法。
403	SignatureDoesNotMatch	The authorization type is malformed.	V2 签名头不是 AWS 开头，或者 V4 签名头不是 AWS4-HMAC-SHA256。
403	SignatureDoesNotMatch	Invalid order of SignedHeaders.	V4 签名 SignedHeasers 需按字典序排序

404	MalformedContinuationToken	The continuation-token you provided invalid.	continuation-token 的值不正确。
404	NoSuchBucket	The resource you requested does not exist.	请求的 Bucket 不存在。
404	NoSuchBucket	The request bucketname is *name*.	复制文件时源 Bucket 不存在。
404	NotSuchBucketPolicy	The bucket policy configuration does not exist.	此 Bucket Policy 配置不存在。
404	NoSuchCORSConfiguration	The CORS configuration does not exist.	Bucket 下没有配置 CORS。
404	NoSuchLifecycleConfiguration	The lifecycle configuration does not exist.	生命周期规则配置不存在。
404	NoSuchKey	The resource you requested does not exist.	请求文件不存在。
404	NoSuchKey	An Error Occurred While Attempting to Retrieve a Custom Error Document.	尝试检索自定义错误文档时发生错误。
404	NoSuchWebsiteConfiguration	The specified bucket does not have a website configuration.	Bucekt 未开启 website 功能。
404	ObjectLockConfigurationNotFoundError	Object lock configuration does not exist for this bucket.	此 Bucket 不存在合规保留配置。
405	MethodNotAllowed	The specified location constraint is not valid.	指定的数据位置无效。
405	MethodNotAllowed	The specified method is not allowed against this resource.	此资源不允许使用该方法。
409	BucketAlreadyExists	BucketAlreadyExists.	Bucket 名字已存在，请更换名字。

409	BucketNotEmpty	The bucket you tried to delete is not empty.	Bucket 非空，不能删除。
411	MissingContentLength	You must provide the Content-Length HTTP header.	需要提供 Content-Length 消息头。
412	PreconditionFailed	Bucket post must be of the enclosure-type multipart/form-data.	POST 表单的 content-type 请求头不正确。
412	PreconditionFailed	The pre-conditions x-amz-copy-source-if-modified-since you specified did not hold.	预处理条件下，指定的 x-amz-copy-source-if-modified-since 不匹配。
503	ServiceUnavailable	Service is busy.Please try again.	服务器繁忙，请重试。
503	SlowDown	Please reduce your request rate.	请降低请求频率。

5 统计 API

说明：以下举例中的域名均以对象存储网络的统计 API 域名 oos-cn-mg.ctyunapi.cn 为例。如果是对象存储网络 2，请使用域名 oos-cn2-mg.ctyunapi.cn。如果是香港精品网和香港普通网，请使用域名 oos-cnhk-mg.ctyunapi.cn。

对象存储网络、对象存储网络 2 和香港节点是相互独立的。如果使用的是对象存储网络的 Host，只能查询对象存储网络内的 Region 的统计信息。如果使用的是对象存储网络 2 的 Host，只能查询对象存储网络 2 内的 Region 的统计信息。如果是香港节点的 Host，只能查询香港节点内的 Region 统计信息。

5.1 统计 API 请求结构

- **接入地址**

对象存储网络接入地址为：oos-cn-mg.ctyunapi.cn。对象存储网络 2 接入地址为：oos-cn2-mg.ctyunapi.cn。香港精品网和香港普通网接入地址为：oos-cnhk-mg.ctyunapi.cn。

- **通信协议**

为了保证通信的安全性，统计支持 HTTP 和 HTTPS。

- **请求方法**

统计支持 GET 请求方法发送请求。

- **请求参数**

每个请求都需要指定如下信息：

- 每个操作接口都需要包含的公共请求参数。
- 操作接口所特有的请求参数。

本节主要描述公共请求参数和请求结果。

说明：在后续提到具体 API 时，举例中都会有**公共请求头**、**公共响应头**，但是不对其进行描述和解释。

5.1.1 公共请求头

在每个请求中，都需要携带公共参数和对应的接口参数。公共请求参数如表所示：

名称	描述	是否必须
Host	统计访问域名。 ● 对象存储网络访问域名为：oos-cn-mg.ctyunapi.cn。 ● 对象存储网络 2 访问域名为：oos-cn2-mg.ctyunapi.cn。 ● 香港精品网和香港普通网访问域名为：oos-cnhk-mg.ctyunapi.cn。	是
Authorization	请求头签名。 支持用户签名验证 V2 和 V4 签名，建议使用 V4 签名。 类型：字符串	是
x-Amz-Date	日期和时间格式必须遵循 ISO 8601 标准，并且必须使用“yyyyMMddT HHmmssZ”格式进行格式化。例如，如果日期和时间是“08/01/2018 15：32：41.982-700”，则必须首先将其转换为 UTC（协调世界时），然后提交为“20180801T083241Z”。	V4 签名必填，V2 签名可以填写 x-Amz-Date 或 Date。
Date	GMT 时间。	V4 签名不必填，V2 签名可以填写 x-Amz-Date 或 Date
Connection	客户端与 OOS 服务器之间的连接状态。 取值： ● keep-alive：长连接，请求结束后继续保持连接。 ● close：短连接，请求结束后关闭连接。 默认值为：keep-alive。	否

5.1.2 公共响应头

每个统计响应结果中都会包含公共响应头。

名称	描述
x-amz-request-id	服务端生成的用于标识请求的 ID。
Content-Type	响应内容类型。
Date	响应日期。
Server	服务器名。
Connection	客户端与 OOS 服务器之间的连接状态。 ● 如果请求时 Connection 值为 keep-alive，请求结束后继续保持连接，不返回此响应头。 ● 如果请求时 Connection 值为 close，请求结束后关闭连接，返回此响应头 Connection: close。

5.2 统计 API 概览

API	描述
GetCapacity	此操作用来查询用户的容量。
GetBilledStorageUsage	此操作用来查询收费容量。
GetRestoreCapacity	此操作用来查询数据取回量。
GetDeleteCapacity	此操作用来查询用户删除的容量。
GetTraffics	此操作用来查询用户的流量。
GetRequests	此操作用来查询用户请求次数。
GetReturnCode	此操作用来查询用户请求返回码次数。
GetConcurrentConnection	此操作用来查询用户的并发连接数。
GetUsage	此操作用来查询用户 Bucket 的使用情况。
GetBandwidth	此操作用来查询用户的已用带宽。

说明：不建议使用 **GetUsage** 和 **GetBandwidth**，建议根据要查询的项，使用下列 API：**GetCapacity**、**GetDeleteCapcity**、**GetRequests**、**GetReturnCode**、**GetConcurrentConnection**。

5.3 统计 API

5.3.1 GetCapacity

此操作用来查询用户的容量。

- **请求参数**

名称	描述	是否必须
Action	GetCapacity。	是
BeginDate	指定查询容量的起始时间，统计数据的起始时间为开始日期（时区为：UTC+8）的 00:00。 **类型**：Time **取值**：格式为 yyyy-MM-dd。	是
EndDate	指定查询容量的结束时间，返回数据的时间为结束日期（时区为：UTC+8）的最后一个数据。 **类型**：Time **取值**：格式为 yyyy-MM-dd。EndDate 不能早于 BeginDate。 **说明**： ● Freq=**by5min**，EndDate 与 BeginDate 的间隔小于 1 天。 ● Freq=**byHour**，EndDate 与 BeginDate 的间隔小于 7 天。 ● Freq=**byDay**，EndDate 与 BeginDate 的间隔小于 30 天。	是
Bucket	指定查询容量的 Bucket 名称，如果不指定 Bucket 名称，则表示查询账户下所有 Bucket 的容量之和。 **类型**：字符串 **取值**：3~63 个字符，只能由小写字母、数字、短横线（-）和点（.）组成。	否
StorageClass	指定查询的存储类型。 **类型**：Enum **取值**： ● **ALL**：所有存储类型。	否

	● **STANDARD**：标准存储。 ● **STANDARD_IA**：低频访问存储。 默认值为 **STANDARD**。	
Region	指定查询容量的数据位置，如果不指定数据位置名称，则查询账户名下所有数据位置的容量之和。 **类型**：字符串 **取值**： ● 对象存储网络：ZhengZhou、ShenYang、ChengDu、WuLuMuQi、LanZhou、QingDao、GuiYang、LaSa、WuHu、WuHan、ShenZhen、SH2、SuZhou。 ● 对象存储网络 2：NeiMeng1、HangZhou1。 ● 香港节点：HongKong。	否
Freq	指定返回统计数据的采样粒度。 **类型**：Enum **取值**： ● **by5min**：统计数据的采样粒度为 5 分钟。 ● **byHour**：统计数据的采样粒度为 1 小时。 ● **byDay**：统计数据的采样粒度为 1 天。 默认值为 **byHour**。	否

● **响应结果**

名称	描述
Account	发出请求的账户名。
UserName	发出请求的用户名。如果根用户发送的该请求，则 UserName 为 root。
StorageClass	存储类型： ● **ALL**：所有存储。 ● **STANDARD**：标准存储。 ● **STANDARD_IA**：低频访问存储。

TimeZone	返回数据的时区，统一为 UTC +0800。
Freq	统计数据的采样粒度： ● **by5min**：统计数据的采样粒度为 5 分钟。 ● **byHour**：统计数据的采样粒度为 1 小时。 ● **byDay**：统计数据的采样粒度为 1 天。
BucketName	Bucket 名字。如果用户在请求中没有携带 Bucket 参数，BucketName 为空。
RegionName	数据位置，如果用户在请求中没有携带 Region 参数，RegionName 为空。
Statistics.Date	返回结果对应的时间： ● 当请求参数 **Freq** 为 **by5min** 或 **byHour** 时，返回格式为 yyyy-MM-dd HH:mm。 ● 当请求参数 **Freq** 为 **byDay** 时，返回格式为 yyyy-MM-dd。
Statistics.Data.MaxCapacity	标准存储数据的容量峰值，单位是 Byte。 如果 **Freq** 值为 **by5min**，该字段值显示为空。
Statistics.Data.AverageCapacity	标准存储数据的容量平均值，单位是 Byte。 如果 **Freq** 值为 **by5min**，该字段值显示为空。
Statistics.Data.SampleCapacity	标准存储数据的实时容量值，单位是 Byte。
Statistics.Standard_ia.MaxCapacity	低频访问存储数据的容量峰值，单位是 Byte。 如果 **Freq** 值为 **by5min**，该字段值显示为空。
Statistics.Standard_ia.AverageCapacity	低频访问存储数据的容量平均值，单位是 Byte。 如果 **Freq** 值为 **by5min**，该字段值显示为空。
Statistics.Standard_ia.SampleCapacity	低频访问存储数据的实时容量值，单位是 Byte。

- **请求示例**

按小时（byHour）查询 Bucket 为 example-bucket、数据位置为 ZhengZhou、2019-10-15 至 2019-10-16 标准存储数据的容量。

```
GET /?Action=GetCapacity&BeginDate=2019-10-15&EndDate=2019-10-
16&StorageClass=STANDARD&Bucket=example-bucket&Freq=byHour&Region=ZhengZhou HTTP/1.1
Date: Wed, 16 Oct 2019 06:24:41 GMT
Authorization: SignatureValue
Host: oos-cn-mg.ctyunapi.cn
```

- 响应示例

```
HTTP/1.1 200 OK
Date: Wed, 16 Oct 2019 06:28:23 GMT
x-amz-request-id: 35560988c43e46d9
Content-Type: application/xml; charset=utf-8
Content-Length: 268
Server: CTYUN

<?xml version="1.0" encoding="UTF-8"?>
<GetCapacityResponse>
         <Account>test@ctyun.cn/Account>
         <UserName>user_test1</UserName>
         <StorageClass>STANDARD</StorageClass>
         <TimeZone>UTC +0800</TimeZone>
         <Freq>byHour</Freq>
         <BucketName>example-bucket</BucketName>
         <RegionName>ZhengZhou</RegionName>
         <Statistics>
           <Date>2019-10-15 00:00</Date>
           <Data>
               <MaxCapacity>10001</MaxCapacity>
               <AverageCapacity>10001</AverageCapacity>
               <SampleCapacity>10001</SampleCapacity>
           </Data>
         </Statistics>
         <Statistics>
           ...
         </Statistics>
</GetCapacityResponse>
```

5.3.2 GetBilledStorageUsage

此操作用来查询收费容量。用户可以根据需求，查询指定存储桶、指定数据位置、指定存储类型的容量。

- **请求参数**

名称	描述	是否必须
Action	GetBilledStorageUsage。	是
BeginDate	指定查询容量的起始时间，统计数据的起始时间为开始日期（时区为：UTC+8）的 00:00。 **类型**：Time **取值**：格式为 yyyy-MM-dd。	是
EndDate	指定查询容量的结束时间，返回数据的时间为当天日期（时区为：UTC+8）的最后一个数据。 **类型**：Time。 **取值**：格式为 yyyy-MM-dd。EndDate 不能早于 BeginDate。 **说明**： ● Freq=**byHour**，EndDate 与 BeginDate 的间隔小于 7 天。 ● Freq=**byDay**，EndDate 与 BeginDate 的间隔小于 30 天。	是
StorageClass	指定查询的存储类型。 类型：Enum 取值： ● **ALL**：所有存储类型。 ● **STANDARAD**：标准类型。 ● **STANDARAD_IA**：低频访问类型。 默认值为 **ALL**。	否
Bucket	指定查询收费容量的 Bucket 名称。如果不指定 Bucket 名称，则表示查询账户下所有 Bucket 的收费容量之和。 **类型**：字符串	否

	取值：3~63 个字符，只能由小写字母、数字、短横线（-）和点（.）组成。	
Region	指定查询收费容量的数据位置。如果不指定数据位置名称，则表示查询账户下所有数据位置的收费容量之和。 **类型**：字符串 **取值**： ● 对象存储网络：ZhengZhou、ShenYang、ChengDu、WuLuMuQi、LanZhou、QingDao、GuiYang、LaSa、WuHu、WuHan、ShenZhen、SH2、SuZhou。 ● 对象存储网络 2：NeiMeng1、HangZhou1。 ● 香港节点：HongKong。	否
Freq	指定返回统计数据的采样粒度。 **类型**：Enum **取值**： ● **byHour**：统计数据的采样粒度为 1 小时。 ● **byDay**：统计数据的采样粒度为 1 天。 默认值为 **byHour**。	否

- **响应结果**

名称	描述
Account	发出请求的账户名。
UserName	发出请求的用户名。如果根用户发送的该请求，则 UserName 为 root。
StorageClass	存储类型： ● **ALL**：所有存储类型。 ● **STANDARAD**：标准类型。 ● **STANDARAD_IA**：低频访问类型。
TimeZone	返回数据的时区，统一为 UTC +0800。
Freq	返回统计数据的采样粒度：

	● **byHour**：统计数据的采样粒度为 1 小时。 ● **byDay**：统计数据的采样粒度为 1 天。
BucketName	Bucket 名字。如果用户在请求中没有携带 Bucket 参数，BucketName 为空。
RegionName	数据位置，如果用户在请求中没有携带 Region 参数，RegionName 为空。
Statistics.Date	返回结果对应的时间： ● 当请求参数 **Freq** 为 **byHour** 时，返回格式为 yyyy-MM-dd HH:mm。 ● 当请求参数 **Freq** 为 **byDay** 时，返回格式为 yyyy-MM-dd。
Statistics.Standard.BilledStorage.BilledStorageUsage	标准存储数据收费容量，单位是 Byte。
Statistics.Standard.BilledStorage.RemainderChargeStorageUsage	标准存储数据补齐容量（时长补齐和大小补齐容量的和值），值为 0。
Statistics.Standard.BilledStorage.RemainderChargeOfDuration	标准存储数据存储时长补齐容量，值为 0。
Statistics.Standard.BilledStorage.RemainderChargeOfSize	标准存储数据大小补齐容量，值为 0。
Statistics.Standard_ia.BilledStorage.BilledStorageUsage	低频访问存储数据收费容量，单位是 Byte。
Statistics.Standard_ia.BilledStorage.RemainderChargeStorageUsage	低频访问存储数据补齐容量（时长补齐和大小补齐容量的和值），单位是 Byte。
Statistics.Standard_ia.BilledStorage.RemainderChargeOfDuration	低频访问存储数据时长补齐容量，单位是 Byte。
Statistics.Standard_ia.BilledStorage.RemainderChargeOfSize	低频访问存储数据大小补齐容量，单位是 Byte。

- **请求示例**

按天（byDay）查询所有存储桶、所有地域 2020-11-17 所有存储类型的收费容量。

```
GET/?Action=GetBilledStorageUsage&BeginDate=2020-11-17&EndDate=2020-11-17&Freq=byDay
HTTP/1.1
Date: Wed, 18 Nov 2020 06:38:27 GMT
Authorization: SignatureValue
Host: oos-cn-mg.ctyunapi.cn
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id: 236A8905248E5A01
Date: Wed, 18Nov 2020 06:39:27 GMT
Content-Length:120
Server: CTYUN

<GetBilledStorageUsageResponse>
  <Account>test@ctyun.cn</Account>
  <UserName>root</UserName>
  <StorageClass>ALL</StorageClass>
  <TimeZone>UTC +0800</TimeZone>
  <Freq>byDay</Freq>
  <BucketName></BucketName>
  <RegionName> </RegionName>
  <Statistics>
    <Date>2020-11-17</Date>
    <Standard_ia>
      <BilledStorage>
        <BilledStorageUsage>40048</BilledStorageUsage>
        <RemainderChargeStorageUsage>30046</RemainderChargeStorageUsage>
        <RemainderChargeOfDuration>20029</RemainderChargeOfDuration>
        <RemainderChargeOfSize>10017</RemainderChargeOfSize>
      </BilledStorage>
    </Standard_ia>
    <Standard>
      <BilledStorage>
        <BilledStorageUsage>10002</BilledStorageUsage>
        <RemainderChargeStorageUsage>0</RemainderChargeStorageUsage>
        <RemainderChargeOfDuration>0</RemainderChargeOfDuration>
        <RemainderChargeOfSize>0</RemainderChargeOfSize>
```

```
      </BilledStorage>
    </Standard>
  </Statistics>
</GetBilledStorageUsageResponse>
```

5.3.3 GetRestoreCapacity

此操作用来查询数据取回量。用户可以根据需求，查询指定存储桶、指定存储类型的数据取回量。

- **请求参数**

名称	描述	是否必须
Action	GetRestoreCapacity。	是
BeginDate	指定查询数据取回量的起始时间，统计数据的起始时间为开始日期（时区为：UTC+8）的 00:00。 **类型**：Time **取值**：格式为 yyyy-MM-dd。	是
EndDate	指定查询数据取回量的结束时间，返回数据的时间为当天日期（时区为：UTC+8）的最后一个数据。 **类型**：Time。 **取值**：格式为 yyyy-MM-dd。EndDate 不能早于 BeginDate。 **说明**： ● Freq=**by5min**，EndDate 与 BeginDate 的间隔小于 1 天。 ● Freq=**byHour**，EndDate 与 BeginDate 的间隔小于 7 天。 ● Freq=**byDay**，EndDate 与 BeginDate 的间隔小于 30 天。	是
StorageClass	指定查询的存储类型。 类型：Enum 取值： ● **ALL**：所有存储类型。 ● **STANDARAD_IA**：低频访问类型。 默认值为 **ALL**。	否
Bucket	指定查询数据取回量的 Bucket 名称。如果不指定 Bucket 名称，则表示查询账户下所有 Bucket 的数据取回量之和。 **类型**：字符串	否

	取值：3~63 个字符，只能由小写字母、数字、短横线（-）和点（.）组成。	
Region	指定查询数据取回量的数据位置。如果不指定数据位置名称，则表示查询账户下所有数据位置的数据取回量之和。 **类型**：字符串 **取值**： ● 对象存储网络：ZhengZhou、ShenYang、ChengDu、WuLuMuQi、LanZhou、QingDao、GuiYang、LaSa、WuHu、WuHan、ShenZhen、SH2、SuZhou。 ● 对象存储网络 2：NeiMeng1、HangZhou1。 ● 香港节点：HongKong。	否
Freq	指定返回统计数据的采样粒度。 **类型**：Enum **取值**： ● **by5min**：统计数据的采样粒度为 5 分钟。 ● **byHour**：统计数据的采样粒度为 1 小时。 ● **byDay**：统计数据的采样粒度为 1 天。 默认值为 **byHour**。	否

- **响应结果**

名称	描述
Account	发出请求的账户名。
UserName	发出请求的用户名。如果根用户发送的该请求，则 UserName 为 root。
StorageClass	存储类型： ● **ALL**：所有存储类型。 ● **STANDARAD_IA**：低频访问类型。
TimeZone	返回数据的时区，统一为 UTC +0800。
Freq	返回统计数据的采样粒度：

	● **by5min**：统计数据的采样粒度为 5 分钟。 ● **byHour**：统计数据的采样粒度为 1 小时。 ● **byDay**：统计数据的采样粒度为 1 天。
BucketName	Bucket 名字。如果用户在请求中没有携带 Bucket 参数，BucketName 为空。
RegionName	数据位置，如果用户在请求中没有携带 Region 参数，RegionName 为空。
Statistics.Date	返回结果对应的时间： ● 当请求参数 **Freq** 为 **by5min** 或 **byHour** 时，返回格式为 yyyy-MM-dd HH:mm。 ● 当请求参数 **Freq** 为 **byDay** 时，返回格式为 yyyy-MM-dd。
Statistics.Standard_ia.RestoreCapacity	低频访问存储取回量，单位是 Byte。

- **请求示例**

按天（byDay）查询所有存储桶、所有地域 2020-11-17 所有存储类型的数据取回量。

```
GET/?Action= GetRestoreCapacity&BeginDate=2020-11-17&EndDate=2020-11-
17&StorageClass=ALL&Freq=byDay HTTP/1.1
Date: Wed, 18 Nov 2020 07:38:27 GMT
Authorization: SignatureValue
Host: oos-cn-mg.ctyunapi.cn
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id: 236A8905248E5A01
Date: Wed, 18Nov 2020 07:39:02 GMT
Content-Length:120
Server: CTYUN

<GetRestoreCapacityResponse>
    <Account>example@ctyun.cn</Account>
    <UserName>user1</UserName>
    <StorageClass>ALL</StorageClass>
```

```
    <TimeZone>utc+8:00</TimeZone>
    <Freq>byDay</Freq>
    <BucketName></BucketName>
    <RegionName></RegionName>
    <Statistics>
      <Date>2020-11-17</Date>
      <Standard_ia>
        <RestoreCapacity>731920486</RestoreCapacity>
      </Standard_ia>
    </Statistics>
</GetRestoreCapacityResponse>
```

5.3.4 GetDeleteCapacity

此操作用来查询用户删除的容量。

- **请求参数**

名称	描述	是否必须
Action	GetDeleteCapacity。	是
BeginDate	指定查询容量的起始时间，统计数据的起始时间为开始日期（时区为：UTC+8）的 00:00。 **类型**：Time **取值**：格式为 yyyy-MM-dd。	是
EndDate	指定查询容量的结束时间，返回数据的时间为当天日期（时区为：UTC+8）的最后一个数据。 **类型**：Time **取值**：格式为 yyyy-MM-dd。EndDate 不能早于 BeginDate。 **说明**： ● Freq=**by5min**，EndDate 与 BeginDate 的间隔小于 1 天。 ● Freq=**byHour**，EndDate 与 BeginDate 的间隔小于 7 天。 ● Freq=**byDay**，EndDate 与 BeginDate 的间隔小于 30 天。	是
StorageClass	指定查询的存储类型。 类型：Enum 取值: ● **ALL**：所有存储类型。 ● **STANDARD**：标准存储。	否

	● **STANDARD_IA**：低频访问存储。 默认值为 **STANDARD**。	
Bucket	指定查询删除容量的 Bucket 名称。如果不指定 Bucket 名称，则表示查询账户下所有 Bucket 的删除容量之和。 **类型**：字符串 **取值**：3~63 个字符，只能由小写字母、数字、短横线（-）和点（.）组成。	否
Region	指定查询删除容量的数据位置。如果不指定数据位置名称，则表示查询账户下所有数据位置的删除容量之和。 **类型**：字符串 **取值**： ● 对象存储网络：ZhengZhou、ShenYang、ChengDu、WuLuMuQi、LanZhou、QingDao、GuiYang、LaSa、WuHu、WuHan、ShenZhen、SH2、SuZhou。 ● 对象存储网络 2：NeiMeng1、HangZhou1。 ● 香港节点：HongKong。	否
Freq	指定返回统计数据的采样粒度。 **类型**：Enum **取值**： ● **by5min**：统计数据的采样粒度为 5 分钟。 ● **byHour**：统计数据的采样粒度为 1 小时。 ● **byDay**：统计数据的采样粒度为 1 天。 默认值为 **byHour**。	否

● **响应结果**

名称	描述
Account	发出请求的账户名。
UserName	发出请求的用户名。如果根用户发送的该请求，则 UserName 为 root。

StorageClass	存储类型： ● **ALL**：所有存储。 ● **STANDARD**：标准存储。 ● **STANDARD_IA**：低频访问存储。
TimeZone	返回数据的时区，统一为 UTC +0800。
Freq	返回统计数据的采样粒度： ● **by5min**：统计数据的采样粒度为 5 分钟。 ● **byHour**：统计数据的采样粒度为 1 小时。 ● **byDay**：统计数据的采样粒度为 1 天。
BucketName	Bucket 名字。如果用户在请求中没有携带 Bucket 参数，BucketName 为空。
RegionName	数据位置，如果用户在请求中没有携带 Region 参数，RegionName 为空。
Statistics.Date	返回结果对应的时间： ● 当请求参数 **Freq** 为 **by5min** 或 **byHour** 时，返回格式为 yyyy-MM-dd HH:mm。 ● 当请求参数 **Freq** 为 **byDay** 时，返回格式为 yyyy-MM-dd。
Statistics.Data.DeleteCapacity	删除标准存储数据的容量。
Statistics.Standard_ia.DeleteCapacity	删除低频访问存储数据的容量。

- **请求示例**

按天（byDay）查询 Bucket 名为 example-bucket、数据位置为 ZhengZhou、2019-10-15 至 2019-10-16 标准存储数据的删除容量。

```
GET /?Action=GetDeleteCapacity&BeginDate=2019-10-15&EndDate=2019-10-16&Bucket=example-bucket&Freq=byDay&Region=ZhengZhou HTTP/1.1
Date: Wed, 16 Oct 2019 06:38:27 GMT
Authorization: SignatureValue
Host: oos-cn-mg.ctyunapi.cn
```

- 响应示例

```
HTTP/1.1 200 OK
Date: Wed, 16 Oct 2019 06:38:28 GMT
x-amz-request-id: da66aa927f744dc1
Content-Length: 440
Content-Type: application/xml; charset=utf-8
Server: CTYUN

<?xml version="1.0" encoding="UTF-8"?>
<GetDeleteCapacityResponse>
    <Account>test@ctyun</Account>
    <UserName>user_test1</UserName>
    <StorageClass>STANDARD</StorageClass >
    <TimeZone>UTC +0800</TimeZone>
    <Freq>byDay</Freq>
    <BucketName>example-bucket</BucketName>
    <RegionName>ZhengZhou</RegionName>
    <Statistics>
        <Date>2019-10-15</Date>
        <Data>
            <DeleteCapacity>1641600</DeleteCapacity>
        </Data>
    </Statistics>
    <Statistics>
        <Date>2019-10-16</Date>
        <Data>
            <DeleteCapacity>1641600</DeleteCapacity>
        </Data>
    </Statistics>
</GetDeleteCapacityResponse>
```

5.3.5 GetTraffics

此操作用来查询用户的流量。

- **请求参数**

名称	描述	是否必须
Action	GetTraffics。	是
BeginDate	指定查询流量的起始时间，统计数据的起始时间为开始日期（时区为：UTC+8）的 00:00。 类型：Time 取值：格式为 yyyy-MM-dd。	是
EndDate	指定查询流量的结束时间，返回数据的时间为当天日期（时区为：UTC+8）的最后一个数据。 **类型**：Time **取值**：格式为 yyyy-MM-dd。EndDate 不能早于 BeginDate。 **说明**： ● Freq=**by5min**，EndDate 与 BeginDate 的间隔小于 1 天。 ● Freq=**byHour**，EndDate 与 BeginDate 的间隔小于 7 天。 ● Freq=**byDay**，EndDate 与 BeginDate 的间隔小于 30 天。	是
StorageClass	指定查询的存储类型。 类型：Enum 取值: ● **ALL**：所有存储类型。 ● **STANDARD**：标准存储。 ● **STANDARD_IA**：低频访问存储。 默认值为 **STANDARD**。	否
Bucket	指定查询流量的 Bucket 名称。如果不指定 Bucket 名称，则表示查询账户下所有 Bucket 的流量之和。 **类型**：字符串	否

	取值：3~63 个字符，只能由小写字母、数字、短横线（-）和点（.）组成。	
Region	指定查询流量的数据位置。如果不指定数据位置，则查询账户名下所有数据位置的流量之和。 **类型**：字符串 **取值**： ● 对象存储网络：ZhengZhou、ShenYang、ChengDu、WuLuMuQi、LanZhou、QingDao、GuiYang、LaSa、WuHu、WuHan、ShenZhen、SH2、SuZhou。 ● 对象存储网络 2：NeiMeng1、HangZhou1。 ● 香港节点：HongKong。	否
InOutType	指定返回的流量类型-上行/下行。 **类型**：字符串 **取值**： ● **all**：指定返回上行流量和下行流量。 ● **inbound**：指定返回上行流量。 ● **outbound**：指定返回所下行流量。 默认值为 **all**。	否
InternetType	指定返回的流量类型-互联网/非互联网。 **类型**：字符串 **取值**： ● **all**：指定返回互联网流量和非互联网流量。 ● **internet**：指定返回互联网流量。 ● **noninternet**：指定返回非互联网流量。 默认值为 **all**。	否
TrafficsType	指定返回的流量类型-直接/漫游。 **类型**：字符串 **取值**：	否

	● **all**：指定返回直接流量和漫游流量。 ● **direct**：指定返回直接流量。 ● **roam**：指定返回漫游流量。 默认值为 **all**。	
NetType	指定返回的网络类型-精品网（CN2）/普通网（163）。 **注意**：仅香港节点支持本参数。 **类型**：字符串 **取值**： ● **all**：指定返回精品网和普通网的流量。 ● **highqualitynet**：指定返回精品网的流量。 ● **normalqualitynet**：指定返回普通网的流量。 默认值为 **all**。	否
Freq	指定返回统计数据的采样粒度。 **类型**：Enum **取值**： ● **by5min**：统计数据的采样粒度为 5 分钟。 ● **byHour**：统计数据的采样粒度为 1 小时。 ● **byDay**：统计数据的采样粒度为 1 天。 默认值为 **byHour**。	否

- **响应结果**

名称	描述
Account	发出请求的账户名。
UserName	发出请求的用户名。如果根用户发送的该请求，则 UserName 为 root。
StorageClass	存储类型: ● **ALL**：所有存储类型。 ● **STANDARD**：标准存储。

	● **STANDARD_IA**：低频访问存储。
TimeZone	返回数据的时区，统一为 UTC +0800。
InOutType	流量类型-上行/下行: ● **all**：上行流量和下行流量。 ● **inbound**：上行流量。 ● **outbound**：下行流量。
InternetType	流量类型-互联网/非互联网: ● **all**：互联网流量和非互联网流量。 ● **internet**：互联网流量。 ● **noninternet**：非互联网流量。
TrafficsType	流量类型-直接/漫游： ● **all**：直接流量和漫游流量。 ● **direct**：直接流量。 ● **roam**：漫游流量。
NetType	网络类型-精品网（CN2）/普通网（163）： ● **all**：返回精品网和普通网的流量。 ● **highqualitynet**：返回精品网的流量。 ● **normalqualitynet**：返回普通网的流量。 **注意**：仅香港节点返回此项。
Freq	返回数据的频率： ● **by5min**：每 5 分钟统计 1 次。 ● **byHour**：每 1 小时统计 1 次。 ● **byDay**：每天统计 1 次。
BucketName	Bucket 名字。如果用户在请求中没有携带 Bucket 参数，BucketName 为空。
RegionName	数据位置，如果用户在请求中没有携带 Region 参数，RegionName 为空。
Statistics.Date	返回结果对应的时间：

	● 当请求参数 **Freq** 为 **by5min** 或 **byHour** 时，返回格式为 yyyy-MM-dd HH:mm。 ● 当请求参数 **Freq** 为 **byDay** 时，返回格式为 yyyy-MM-dd。
Statistics.Data.InternetDirectInbound	标准存储数据的互联网直接上行流量，单位是 Byte。
Statistics.Data.InternetRoamInbound	标准存储数据的互联网漫游上行流量，单位是 Byte。
Statistics.Data.NonInternetDirectInbound	标准存储数据的非互联网直接上行流量，单位是 Byte。
Statistics.Data.NonInternetRoamInbound	标准存储数据的非互联网漫游上行流量，单位是 Byte。
Statistics.Data.InternetDirectOutbound	标准存储数据的互联网直接下行流量，单位是 Byte。
Statistics.Data.InternetRoamOutbound	标准存储数据的互联网漫游下行流量，单位是 Byte。
Statistics.Data.NonInternetDirectOutbound	标准存储数据的非互联网直接下行流量，单位是 Byte。
Statistics.Data.NonInternetRoamOutbound	标准存储数据的非互联网漫游下行流量，单位是 Byte。
Statistics.Standard_ia.InternetDirectInbound	低频访问存储数据的互联网直接上行流量，单位是 Byte。
Statistics.Standard_ia.InternetRoamInbound	低频访问存储数据的互联网漫游上行流量，单位是 Byte。
Statistics.Standard_ia.NonInternetDirectInbound	低频访问存储数据的非互联网直接上行流量，单位是 Byte。
Statistics.Standard_ia.NonInternetRoamInbound	低频访问存储数据的非互联网漫游上行流量，单位是 Byte。

Statistics.Standard_ia.InternetDirectOutbound	低频访问存储数据的互联网直接下行流量，单位是 Byte。
Statistics.Standard_ia.InternetRoamOutbound	低频访问存储数据的互联网漫游下行流量，单位是 Byte。
Statistics.Standard_ia.NonInternetDirectOutbound	低频访问存储数据的非互联网直接下行流量，单位是 Byte。
Statistics.Standard_ia.NonInternetRoamOutbound	低频访问存储数据的非互联网漫游下行流量，单位是 Byte。

- **请求示例**

按天（byDay）查询 Bucket 为 example-bucket、数据位置为 ZhengZhou、2019-10-15 至 2019-10-16 标准存储数据的流量。

```
GET /?Action=GetTraffics&BeginDate=2019-10-15&EndDate=2019-10-16&
StorageClass=STANDARD&Bucket=example-bucket&Freq=byDay&Region=ZhengZhou HTTP/1.1
Date: Wed, 16 Oct 2019 06:41:57 GMT
Authorization: SignatureValue
Host: oos-cn-mg.ctyunapi.cn
```

- **响应示例**

```
HTTP/1.1 200 OK
Date: Wed, 16 Oct 2019 06:41:57 GMT
x-amz-request-id: c1bd7162b01e4a45
Content-Length: 1316
Content-Type: application/xml; charset=utf-8
Server: CTYUN

<?xml version="1.0" encoding="UTF-8"?>
<GetTrafficsResponse>
    <Account>test@ctyun.cn</Account>
     <UserName>user_test1</UserName>
    <StorageClass>STANDARD</StorageClass >
    <TimeZone>UTC +0800</TimeZone>
    <InOutType>all</InOutType>
    <InternetType>all</InternetType>
    <TrafficsType>all</TrafficsType>
```

```
    <Freq>byDay</Freq>
    <BucketName>example-bucket</BucketName>
    <RegionName>ZhengZhou</RegionName>
    <Statistics>
        <Date>2019-10-15</Date>
        <Data>
            <InternetDirectInbound>86400</InternetDirectInbound>
            <InternetRoamInbound>259200</InternetRoamInbound>
            <NonInternetDirectInbound>432000</NonInternetDirectInbound>
            <NonInternetRoamInbound>604800</NonInternetRoamInbound>
            <InternetDirectOutbound>172800</InternetDirectOutbound>
            <InternetRoamOutbound>345600</InternetRoamOutbound>
            <NonInternetDirectOutbound>518400</NonInternetDirectOutbound>
            <NonInternetRoamOutbound>691200</NonInternetRoamOutbound>
        </Data>
    </Statistics>
    <Statistics>
        ...
    </Statistics>
</GetTrafficsResponse>
```

5.3.6 GetRequests

此操作用来查询用户的请求次数。

- **请求参数**

名称	描述	是否必须
Action	GetRequests。	是
BeginDate	指定查询请求次数的起始时间，统计数据的起始时间（时区为：UTC+8）为开始日期的 00:00。 **类型**：Time **取值**：格式为 yyyy-MM-dd。	是
EndDate	指定查询请求的结束时间，返回数据的时间为当天日期（时区为：UTC+8）的最后一个数据。 **类型**：Time **取值**：格式为 yyyy-MM-dd。EndDate 不能早于 BeginDate。 **说明**： ● Freq=**by5min**，EndDate 与 BeginDate 的间隔小于 1 天。 ● Freq=**byHour**，EndDate 与 BeginDate 的间隔小于 7 天。 ● Freq=**byDay**，EndDate 与 BeginDate 的间隔小于 30 天。	是
StorageClass	指定查询的存储类型。 类型：Enum 取值: ● **ALL**：所有存储类型。 ● **STANDARD**：标准存储。 ● **STANDARD_IA**：低频访问存储。 默认值为 **STANDARD**。	否

Bucket	指定查询请求次数的 Bucket 名称。如果不指定 Bucket 名称，则表示查询账户下所有的 Bucket 的请求次数之和。 **类型**：字符串 **取值**：3~63 个字符，只能由小写字母、数字、短横线（-）和点（.）组成。	否
Region	指定查询请求次数的数据位置。如果不指定数据位置，则查询账户名下所有数据位置的请求次数之和。 **类型**：字符串 **取值**： ● 对象存储网络 ZhengZhou、ShenYang、ChengDu、WuLuMuQi、LanZhou、QingDao、GuiYang、LaSa、WuHu、WuHan、ShenZhen、SH2、SuZhou。 ● 对象存储网络 2：NeiMeng1、HangZhou1。 ● 香港节点：HongKong。	否
InternetType	指定返回的请求次数网络类型-互联网/非互联网。 **类型**：字符串 **取值**： ● all：指定返回互联网请求次数和非互联网请求次数。 ● internet：指定返回互联网的请求次数。 ● noninternet：指定返回非互联网的请求次数。 默认值为 **all**。	否
RequestsType	指定返回的请求类型。 **类型**：字符串 **取值**： ● all：指定返回所有类型的请求次数。	否

	● get：指定返回 get 类型的请求次数。 ● head：指定返回 head 类型的请求次数。 ● put：指定返回 put 类型的请求次数。 ● post：指定返回 post 类型的请求次数。 ● delete：指定返回 delete 类型的请求次数。 ● others：指定返回除上述外的其他类型的请求次数。 默认值为 **all**。	
Freq	指定返回统计数据的采样粒度。 **类型**：Enum **取值**： ● **by5min**：统计数据的采样粒度为 5 分钟。 ● **byHour**：统计数据的采样粒度为 1 小时。 ● **byDay**：统计数据的采样粒度为 1 天。 默认值为 **byHour**。	否

- **响应结果**

名称	描述
Account	发出请求的账户名。
UserName	发出请求的用户名。如果根用户发送的该请求，则 UserName 为 root。
StorageClass	存储类型： ● **ALL**：所有存储类型。 ● **STANDARD**：标准存储。 ● **STANDARD_IA**：低频访问存储。

TimeZone	返回数据的时区，统一为 UTC +0800。
InternetType	返回的请求次数网络类型-互联网/非互联网： ● all：互联网请求次数和非互联网请求次数。 ● internet：互联网的请求次数。 ● noninternet：非互联网的请求次数。
RequestsType	请求类型： ● all:所有类型的请求次数。 ● get：get 类型的请求次数。 ● head：head 类型的请求次数。 ● put：put 类型的请求次数。 ● post：post 类型的请求次数。 ● delete：delete 类型的请求次数。 ● others：除上述外的其他类型的请求次数。
Freq	返回数据的频率： ● **by5min**：每 5 分钟统计 1 次。 ● **byHour**：每 1 小时统计 1 次。 ● **byDay**：每天统计 1 次。
BucketName	Bucket 名字。如果用户在请求中没有携带 Bucket 参数，BucketName 为空。
RegionName	数据位置，如果用户在请求中没有携带 Region 参数，RegionName 为空。
Statistics.Date	返回结果对应的时间： ● 当请求参数 **Freq** 为 **by5min** 或 **byHour** 时，返回格式为 yyyy-MM-dd HH:mm。

	● 当请求参数 **Freq** 为 **byDay** 时，返回格式为 yyyy-MM-dd。
Statistics.Requests	请求次数。
Statistics.Data.Internet.GetRequest	标准存储数据的互联网的 Get 请求次数。
Statistics.Data.Internet.HeadRequest	标准存储数据的互联网的 Head 请求次数。
Statistics.Data.Internet.PutRequest	标准存储数据的互联网的 Put 请求次数。
Statistics.Data.Internet.PostRequest	标准存储数据的互联网的 Post 请求次数。
Statistics.Data.Internet.DeleteRequest	标准存储数据的互联网的 Delete 请求次数。
Statistics.Data.Internet.OthersRequest	标准存储数据的互联网的 Others 请求次数。
Statistics.Data.NonInternet.GetRequest	标准存储数据的非互联网的 Get 请求次数。
Statistics.Data.NonInternet.HeadRequest	标准存储数据的非互联网的 Head 请求次数。
Statistics.Data.NonInternet.PutRequest	标准存储数据的非互联网的 Put 请求次数。
Statistics.Data.NonInternet.PostRequest	标准存储数据的非互联网的 Post 请求次数。
Statistics.Data.NonInternet.DeleteRequest	标准存储数据的非互联网的 Delete 请求次数。
Statistics.Data.NonInternet.OthersRequest	标准存储数据的非互联网的 Others 请求次数。
Statistics.Standard_ia.Internet.GetRequest	低频访问存储数据的互联网的 Get 请求次数。
Statistics.Standard_ia.Internet.HeadRequest	低频访问存储数据的互联网的 Head 请求次数。
Statistics.Standard_ia.Internet.PutRequest	低频访问存储数据的互联网的 Put 请求次数。
Statistics.Standard_ia.Internet.PostRequest	低频访问存储数据的互联网的 Post 请求次数。

Statistics.Standard_ia.Internet.DeleteRequest	低频访问存储数据的互联网的 Delete 请求次数。
Statistics.Standard_ia.Internet.OthersRequest	低频访问存储数据的互联网的 Others 请求次数。
Statistics.Standard_ia.NonInternet.GetRequest	低频访问存储数据的非互联网的 Get 请求次数。
Statistics.Standard_ia.NonInternet.HeadRequest	低频访问存储数据的非互联网的 Head 请求次数。
Statistics.Standard_ia.NonInternet.PutRequest	低频访问存储数据的非互联网的 Put 请求次数。
Statistics.Standard_ia.NonInternet.PostRequest	低频访问存储数据的非互联网的 Post 请求次数。
Statistics.Standard_ia.NonInternet.DeleteRequest	低频访问存储数据的非互联网的 Delete 请求次数。
Statistics.Standard_ia.NonInternet.OthersRequest	低频访问存储数据的非互联网的 Others 请求次数。

- **请求示例**

按天（byDay）查询 Bucket 为 example-bucket、数据位置为 ZhengZhou、2019-10-15 至 2019-10-16 的所有标准存储数据请求次数。

```
GET /?Action=GetRequests&BeginDate=2019-10-15&EndDate=2019-10-16&Bucket=example-
bucket&Freq=byDay&Region=ZhengZhou HTTP/1.1
Date: Wed, 16 Oct 2019 06:51:49 GMT
Authorization: SignatureValue
Host: oos-cn-mg.ctyunapi.cn
```

- **响应示例**

```
HTTP/1.1 200 OK
Date: Wed, 16 Oct 2019 06:51:50 GMT
x-amz-request-id: 60d2744539e84af5
```

```
Content-Length: 1244
Content-Type: application/xml; charset=utf-8
Server: CTYUN

<?xml version="1.0" encoding="UTF-8"?>
<GetRequestsResponse>
    <Account>test@ctyun.cn</Account>
    <UserName>user_test1</UserName>
    <TimeZone>UTC +0800</TimeZone>
    <InternetType>all</InternetType>
    <RequestsType>all</RequestsType>
    <Freq>byDay</Freq>
    <BucketName>example-bucket</BucketName>
    <RegionName>ZhengZhou</RegionName>
    <Statistics>
        <Date>2019-10-15</Date>
        <Data>
            <Internet>
                <GetRequest>101</GetRequest>
                <HeadRequest>102</HeadRequest>
                <PutRequest>103</PutRequest>
                <PostRequest>104</PostRequest>
                <DeleteRequest>105</DeleteRequest>
                <OthersRequest>106</OthersRequest>
            </Internet>
            <NonInternet>
                <GetRequest>107</GetRequest>
                <HeadRequest>108</HeadRequest>
                <PutRequest>109</PutRequest>
                <PostRequest>110</PostRequest>
                <DeleteRequest>111</DeleteRequest>
                <OthersRequest>112</OthersRequest>
            </NonInternet>
        </Data>
    </Statistics>
    <Statistics>
        ...
```

```
    </Statistics>
</GetRequestsResponse>
```

5.3.7 GetReturnCode

此操作用来查询用户请求返回码次数。

- **请求参数**

名称	描述	是否必须
Action	GetReturnCode。	是
BeginDate	指定查询返回码次数的起始时间，统计数据的起始时间为开始日期（时区为：UTC+8）的 00:00。 **类型**：Time **取值**：格式为 yyyy-MM-dd。	是
EndDate	指定查询返回码统计次数的结束时间，返回数据的时间为当天日期（时区为：UTC+8）的最后一个数据。 **类型**：Time **取值**：格式为 yyyy-MM-dd。EndDate 不能早于 BeginDate。 **说明**： ● Freq=**by5min**，EndDate 与 BeginDate 的间隔小于 1 天。 ● Freq=**byHour**，EndDate 与 BeginDate 的间隔小于 7 天。 ● Freq=**byDay**，EndDate 与 BeginDate 的间隔小于 30 天。	是
StorageClass	指定查询的存储类型。 类型：Enum 取值: ● **ALL**：所有存储类型。 ● **STANDARD**：标准存储。 ● **STANDARD_IA**：低频访问存储。 默认值为 **STANDARD**。	否

Bucket	指定查询返回码的 Bucket 名称。如果不指定 Bucket 名称，则表示查询账户下所有的 Bucket 的请求返回码次数之和。 **类型**：字符串 **取值**：3~63 个字符，只能由小写字母、数字、短横线（-）和点（.）组成。	否
Region	指定查询返回码的数据位置。如果不指定数据位置名称，则查询账户名下所有数据位置的请求返回码次数之和。 **类型**：字符串 **取值**： ● 对象存储网络：ZhengZhou、ShenYang、ChengDu、WuLuMuQi、LanZhou、QingDao、GuiYang、LaSa、WuHu、WuHan、ShenZhen、SH2、SuZhou。 ● 对象存储网络 2：NeiMeng1、HangZhou1。 ● 香港节点：HongKong。	否
InternetType	指定返回码的网络类型-互联网/非互联网。 **类型**：字符串 **取值**： ● **all**：指定返回互联网返回码和非互联网返回码。 ● **internet**：指定返回互联网的返回码。 ● **noninternet**：指定返回非互联网的返回码。 默认值为 all。	否
RequestsType	指定返回返回码次数的请求类型，可以指定一个或多个返回码的请求类型，当申请多个 RequestsType 时，用“+”将类型分开。 **类型**：字符串	否

	取值： ● all：指定返回所有请求类型的返回码次数。 ● get：指定返回 get 类型的返回码次数。 ● head：指定返回 head 类型的返回码次数。 ● put：指定返回 put 类型的返回码次数。 ● post：指定返回 post 类型的返回码次数。 ● delete：指定返回 delete 类型的返回码次数。 ● others：指定返回除上述外的其他类型的返回码次数。 默认值为 all。	
ResponseType	指定返回的返回码类型，可以指定一个或多个返回码的请求类型，当申请多个 ResponseType 时，用“+”将类型分开。 **类型**：字符串 **取值：** ● **all**：指定返回所有类型的返回码次数。 ● **Response200**。 ● **Response204**：仅 **delete** 支持 **Response204**。 ● **Response206**：仅 **get** 和 **head** 支持 **Response206**。 ● **Response403**。 ● **Response404**。 ● **Response4xx**。 ● **Response500**。	否

	● **Response503**。 默认值为 **all**。	
Freq	指定返回统计数据的采样粒度。 **类型**：Enum **取值**： ● **by5min**：统计数据的采样粒度为 5 分钟。 ● **byHour**：统计数据的采样粒度为 1 小时。 ● **byDay**：统计数据的采样粒度为 1 天。 默认值为 **byHour**。	否

- **响应结果**

名称	描述
Account	发出请求的账户名。
UserName	发出请求的用户名。如果根用户发送的该请求，则 UserName 为 root。
StorageClass	指定查询的存储类型： ● **ALL**：所有存储类型。 ● **STANDARD**：标准存储。 ● **STANDARD_IA**：低频访问存储。
TimeZone	返回数据的时区，统一为 UTC +0800。
InternetType	返回的返回码类型-互联网/非互联网: ● **all**:互联网返回码和非互联网返回码。 ● **internet**：互联网返回码。 ● **noninternet**：非互联网返回码。

RequestsType	返回码次数的请求类型，当返回多个时，用分隔符“+”连接： ● **all**：所有请求类型的返回码次数。 ● **get**：get 类型的返回码次数。 ● **head**：head 类型的返回码次数。 ● **put**：put 类型的返回码次数。 ● **post**：post 类型的返回码次数。 ● **delete**：delete 类型的返回码次数。 ● **others**：除上述外的其他类型的返回码次数。
ResponseType	返回的返回码类型，当返回多个时，用分隔符“+”连接： ● **all**：所有类型的返回码。 ● **Response200**。 ● **Response204**。 ● **Response206**。 ● **Response403**。 ● **Response404**。 ● **Response4xx**。 ● **Response500**。 ● **Response503**。
Freq	返回统计数据的采样粒度： ● **by5min**：统计数据的采样粒度为 5 分钟。 ● **byHour**：统计数据的采样粒度为 1 小时。 ● **byDay**：统计数据的采样粒度为 1 天。
BucketName	请求的 Bucket 名字。如果用户在请求中没有携带 Bucket 参数，BucketName 为空。
RegionName	数据位置，如果用户在请求中没有携带 Region 参数，RegionName 为空。

Statistics.Date	返回结果对应的时间： ● 当请求参数 **Freq** 为 **by5min** 或 **byHour** 时，返回格式为 yyyy-MM-dd HH:mm。 ● 当请求参数 **Freq** 为 **byDay** 时，返回格式为 yyyy-MM-dd。
Statistics.Data.Internet.Get	标准存储数据互联网 Get 类型请求码的次数。可以具体到下列请求类的次数： ● Response200。 ● Response206。 ● Response403。 ● Response404。 ● Response4xx。 ● Response500。 ● Response503。
Statistics.Data.Internet.Head	标准存储数据互联网 Head 类型请求码的次数。可以具体到下列请求类的次数： ● Response200。 ● Response206。 ● Response403。 ● Response404。 ● Response4xx。 ● Response500。 ● Response503。
Statistics.Data.Internet.Put	标准存储数据互联网 Put 类型请求码的次数。可以具体到下列请求类的次数： ● Response200。 ● Response403。 ● Response404。

	● Response4xx。 ● Response500。 ● Response503。
Statistics.Data.Internet.Post	标准存储数据互联网 Post 类型请求码的次数。可以具体到下列请求类的次数： ● Response200。 ● Response403。 ● Response404。 ● Response4xx。 ● Response500。 ● Response503。
Statistics.Data.Internet.Delete	标准存储数据互联网 Delete 类型请求码的次数。可以具体到下列请求类的次数： ● Response200。 ● Response204。 ● Response403。 ● Response404。 ● Response4xx。 ● Response500。 ● Response503。
Statistics.Data.Internet.Others	标准存储数据互联网 Others 类型请求码的次数。可以具体到下列请求类的次数： ● Response200。 ● Response403。 ● Response404。 ● Response4xx。 ● Response500。 ● Response503。

Statistics.Data.NonInternet.Get	标准存储数据非互联网 Get 类型请求码的次数。可以具体到下列请求类的次数： ● Response200。 ● Response206。 ● Response403。 ● Response404。 ● Response4xx。 ● Response500。 ● Response503。
Statistics.Data.NonInternet.Head	标准存储数据非互联网 Head 类型请求码的次数。可以具体到下列请求类的次数： ● Response200。 ● Response206。 ● Response403。 ● Response404。 ● Response4xx。 ● Response500。 ● Response503。
Statistics.Data.NonInternet.Put	标准存储数据非互联网 Put 类型请求码的次数。可以具体到下列请求类的次数： ● Response200。 ● Response403。 ● Response404。 ● Response4xx。 ● Response500。 ● Response503。
Statistics.Data.NonInternet.Post	标准存储数据非互联网 Post 类型请求码的次数。可以具体到下列请求类的次数：

	● Response200。 ● Response403。 ● Response404。 ● Response4xx。 ● Response500。 ● Response503。
Statistics.Data.NonInternet.Delete	标准存储数据非互联网 Delete 类型请求码的次数。 可以具体到下列请求类的次数： ● Response200。 ● Response204。 ● Response403。 ● Response404。 ● Response4xx。 ● Response500。 ● Response503。
Statistics.Data.NonInternet.Others	标准存储数据非互联网 Others 类型请求码的次数。 可以具体到下列请求类的次数： ● Response200。 ● Response403。 ● Response404。 ● Response4xx。 ● Response500。 ● Response503。
Statistics.Standard_ia.Internet.Get	低频访问存储数据互联网 Get 类型请求码的次数。 可以具体到下列请求类的次数： ● Response200。 ● Response206。 ● Response403。

	● Response404。 ● Response4xx。 ● Response500。 ● Response503。
Statistics.Standard_ia.Internet.Head	低频访问存储数据互联网 Head 类型请求码的次数。 可以具体到下列请求类的次数： ● Response200。 ● Response206。 ● Response403。 ● Response404。 ● Response4xx。 ● Response500。 ● Response503。
Statistics.Standard_ia.Internet.Put	低频访问存储数据互联网 Put 类型请求码的次数。 可以具体到下列请求类的次数： ● Response200。 ● Response403。 ● Response404。 ● Response4xx。 ● Response500。 ● Response503。
Statistics.Standard_ia.Internet.Post	低频访问存储数据互联网 Post 类型请求码的次数。 可以具体到下列请求类的次数： ● Response200。 ● Response403。 ● Response404。 ● Response4xx。 ● Response500。

	● Res ponse503。
Statistics.Standard_ia.Internet.Delete	低频访问存储数据互联网 Delete 类型请求码的次数。可以具体到下列请求类的次数： ● Response200。 ● Response204。 ● Response403。 ● Response404。 ● Response4xx。 ● Response500。 ● Response503。
Statistics.Standard_ia.Internet.Others	低频访问存储数据互联网 Others 类型请求码的次数。可以具体到下列请求类的次数： ● Response200。 ● Response403。 ● Response404。 ● Response4xx。 ● Response500。 ● Response503。
Statistics.Standard_ia.NonInternet.Get	低频访问存储数据非互联网 Get 类型请求码的次数。可以具体到下列请求类的次数： ● Response200。 ● Response206。 ● Response403。 ● Response404。 ● Response4xx。 ● Response500。 ● Response503。

Statistics.Standard_ia.NonInternet.Head	低频访问存储数据非互联网 Head 类型请求码的次数。可以具体到下列请求类的次数： ● Response200。 ● Response206。 ● Response403。 ● Response404。 ● Response4xx。 ● Response500。 ● Response503。
Statistics.Standard_ia.NonInternet.Put	低频访问存储数据非互联网 Put 类型请求码的次数。可以具体到下列请求类的次数： ● Response200。 ● Response403。 ● Response404。 ● Response4xx。 ● Response500。 ● Response503。
Statistics.Standard_ia.NonInternet.Post	低频访问存储数据非互联网 Post 类型请求码的次数。可以具体到下列请求类的次数： ● Response200。 ● Response403。 ● Response404。 ● Response4xx。 ● Response500。 ● Response503。
Statistics.Standard_ia.NonInternet.Delete	低频访问存储数据非互联网 Delete 类型请求码的次数。可以具体到下列请求类的次数： ● Response200。

	● Response204。 ● Response403。 ● Response404。 ● Response4xx。 ● Response500。 ● Response503。
Statistics.Standard_ia.NonInternet.Others	低频访问存储数据非互联网 Others 类型请求码的次数。可以具体到下列请求类的次数： ● Response200。 ● Response403。 ● Response404。 ● Response4xx。 ● Response500。 ● Response503。

- **请求示例**

按天（byDay）查询 Bucket 为 example-bucket、数据位置为 ZhengZhou、2019-10-15 至 2019-10-16 标准存储数据的返回码次数。

```
GET /?Action=GetReturnCode&BeginDate=2019-10-15&EndDate=2019-10-
16&StorageClass=STANDARD&Bucket=example-
bucket&Freq=byDay&Region=ZhengZhou&RequestsType=get%2Bput&ResponseType=Response200
HTTP/1.1
Date: Wed, 16 Oct 2019 07:03:54 GMT
Authorization: SignatureValue
Host: oos-cn-mg.ctyunapi.cn
```

- **响应示例**

```
HTTP/1.1 200 OK
Date: Wed, 16 Oct 2019 07:03:54 GMT
x-amz-request-id: a79028c963e4d3a
Content-Length: 1114
Content-Type: application/xml; charset=utf-8
```

```
Server: CTYUN

<?xml version="1.0" encoding="UTF-8"?>
<GetReturnCodeReponse>
    <Account>test@ctyun.cn</Account>
    <UserName>user_test1</UserName>
    <StorageClass>STANDARD</StorageClass>
    <TimeZone>UTC +0800</TimeZone>
    <InternetType>all</InternetType>
    <RequestsType>get+put</RequestsType>
    <ResponseType>Response200 </ResponseType>
    <Freq>byDay</Freq>
    <BucketName>example-bucket</BucketName>
    <RegionName>ZhengZhou</RegionName>
    <Statistics>
        <Date>2019-10-15</Date>
        <Data>
        <Internet>
            <Get>
                <Response200>1</Response200>
            </Get>
            <Put>
                <Response200>11</Response200>
            </Put>
        </Internet>
        <NonInternet>
            <Get>
                <Response200>61</Response200>
            </Get>
            <Put>
                <Response200>71</Response200>
            </Put>
        </NonInternet>
        </Data>
    </Statistics>
    <Statistics>
        ...
    </Statistics>
</GetReturnCodeReponse>
```

5.3.8 GetConcurrentConnection

此操作用来查询用户的并发连接数。

- **请求参数**

名称	描述	是否必须
Action	GetConcurrentConnection。	是
BeginDate	指定查询并发连接数的起始时间，统计数据的起始时间为开始日期（时区为：UTC+8）的 00:00。 **类型**：Time **取值**：格式为 yyyy-MM-dd。	是
EndDate	指定查询并发连接数的结束时间，返回数据的时间为当天日期（时区为：UTC+8）的最后一个数据。 **类型**：Time **取值**：格式为 yyyy-MM-dd。EndDate 不能早于 BeginDate。 说明：EndDate 与 BeginDate 的间隔小于 1 天。	是
Bucket	指定查询并发连接数 Bucket 的名称。如果不指定 Bucket 名称，则表示查询账户下所有 Bucket 的并发连接数之和。 **类型**：字符串 **取值**：3~63 个字符，只能由小写字母、数字、短横线（-）和点（.）组成。	否
Region	指定查询并发连接数的数据位置。如果不指定数据位置名称，则查询账户名下所有数据位置的并发连接数之和。 **类型**：字符串 **取值**： ● 对象存储网络：ZhengZhou、ShenYang、ChengDu、WuLuMuQi、LanZhou、QingDao、GuiYang、LaSa、WuHu、WuHan、ShenZhen、SH2、SuZhou。 ● 对象存储网络 2：NeiMeng1、HangZhou1。	否

	● 香港节点：HongKong。	
InternetType	指定返回并发连接数的网络类型-互联网/非互联网。 **类型**：字符串 **取值**： ● **all**：指定返回互联网并发连接数和非互联网并发连接数。 ● **internet**：指定返回互联网的并发连接数。 ● **noninternet**：指定返回非互联网的并发连接数。 默认值为 **all**。	否
Freq	指定返回统计数据的采样粒度。 **类型**：Enum **取值**：**by5min**：统计数据的采样粒度为 5 分钟。 默认值为 **by5min**。	否

● **响应结果**

名称	描述
Account	发出请求的账户名。
UserName	发出请求的用户名。如果根用户发送的该请求，则 UserName 为 root。
TimeZone	返回数据的时区，统一为 UTC +0800。
InternetType	并发连接数网络类型-互联网/非互联网： ● all：互联网并发连接数和非互联网并发连接数。 ● internet：互联网的并发连接数。 ● noninternet：非互联网的并发连接数。
Freq	返回数据的频率，by5min：每 5 分钟统计一次。
BucketName	Bucket 名字。如果用户在请求中没有携带 Bucket 参数，BucketName 为空。
RegionName	数据位置，如果用户在请求中没有携带 Region 参数，RegionName 为空。
Statistics.Date	返回结果对应的时间。

Statistics.Data.InternetConnection	互联网的并发连接数。
Statistics.Data.NonInternetConnection	非互联网的并发连接数。

- **请求示例**

按 5 分钟（by5min）查询 Bucket 为 example-bucket、数据位置为 ZhengZhou、2019-10-15 至 2019-10-16 的并发量。

```
GET /?Action=GetConcurrentConnection&BeginDate=2019-10-16&EndDate=2019-10-16&Bucket=example-bucket&Freq=by5min&Region=ZhengZhou HTTP/1.1
Date: Wed, 16 Oct 2019 07:11:39 GMT
Authorization: SignatureValue
Host: oos-cn-mg.ctyunapi.cn
```

- **响应示例**

```
HTTP/1.1 200 OK
Date: Wed, 16 Oct 2019 07:11:39 GMT
x-amz-request-id: 43f5cbc9bef047d3
Content-Length: 24493
Server: CTYUN

<?xml version="1.0" encoding="UTF-8"?>
<GetConcurrentConnectionResponse>
    <Account>test@ctyun.cn</Account>
    <UserName>user_test1</UserName>
    <TimeZone>UTC +0800</TimeZone>
    <InternetType>all</InternetType>
    <Freq>by5min</Freq>
    <BucketName>example-bucket</BucketName>
    <RegionName>ZhengZhou</RegionName>
    <Statistics>
        <Date>2019-10-16 00:00</Date>
        <Data>
            <InternetConnection>0</InternetConnection>
            <NonInternetConnection>0</NonInternetConnection>
        </Data>
    </Statistics>
    <Statistics>
        ...
    </Statistics>
```

```
</GetConcurrentConnectionResponse>
```

5.3.9 GetUsage

此操作用来查询用户 Bucket 的使用情况。

说明：不建议使用此 API，建议根据要查询的项，使用下列 API：GetCapacity、GetDeleteCapcity、GetRequests、GetReturnCode、GetConcurrentConnection。

- **请求参数**

名称	描述	是否必须
Action	GetUsage。	是
BeginDate	指定查询的起始时间，统计数据的起始时间为开始日期（时区为：UTC+8）的 00:00。 **类型**：Time **取值**： ● 若以格式 YYYY-MM-DD 查询，返回以天为单位的数据。 ● 若以格式 YYYY-MM-DD-hh-mm 查询，返回以 5 分钟为单位的数据，mm 只能取 00、05、10.....55。 **注意**：若以格式 YYYY-MM-DD-hh-mm 查询，日期不能早于当前时间一个月。	是
EndDate	指定查询的结束时间，返回数据的时间为当天日期（时区为：UTC+8）的最后一个数据。 **类型**：Time **取值**：EndDate 既不能早于 BeginDate，也不能晚于当前时间。 ● 若以格式 YYYY-MM-DD 查询，返回以天为单位的数据。 ● 若以格式 YYYY-MM-DD-hh-mm 查询，返回以 5 分钟为单位的数据，mm 只能取 00、05、10.....55。	是

Bucket	指定查询使用情况的 Bucket 的名称，如果不指定 Bucket 名称，则表示查询账户下所有的 Bucket 的统计信息之和。 **类型**：字符串 **取值**：3~63 个字符，只能由小写字母、数字、短横线（-）和点（.）组成。	否

- **响应结果**

名称	描述
UserName	发出请求的账户名。
BucketName	Bucket 名字。
Region.Name	数据位置。
Region.Data.Date	统计的日期。
Region.Data.Capacity	已使用的容量，单位为 Byte。
Region.Data.Upload	互联网直接上行流量，单位为 Byte。
Region.Data.Download	互联网直接下行流量，单位为 Byte。
Region.Data.DeleteFlow	互联网删除流量，单位为 Byte。
Region.Data.RoamUpload	互联网漫游上行流量，单位为 Byte。
Region.Data.RoamDownload	互联网漫游下行流量，单位为 Byte。
Region.Data.GetRequest	互联网 Get 请求次数。
Region.Data.HeadRequest	互联网 Head 请求次数。
Region.Data.PutRequest	互联网 Put 请求次数。
Region.Data.PostRequest	互联网 Post 请求次数。
Region.Data.DeleteRequest	互联网 Delete 请求次数。
Region.Data.OtherRequest	互联网除 Get、Head、Put、Post、Delete 外的其他请求次数。
Region.Data.NonInternetUpload	非互联网直接上行流量，单位为 Byte。
Region.Data.NonInternetDownload	非互联网直接下行流量，单位为 Byte。

Region.Data.NonInternetDeleteFlow	非互联网删除流量，单位为 Byte。
Region.Data.NonInternetRoamUpload	非互联网漫游上行流量，单位为 Byte。
Region.Data.NonInternetRoamDownload	非互联网漫游下行流量，单位为 Byte。
Region.Data.NonInternetGeRequest	非互联网 Get 请求次数。
Region.Data.NonInternetHeadRequest	非互联网 Head 请求次数。
Region.Data.NonInternetPutRequest	非互联网 Put 请求次数。
Region.Data.NonInternetPostRequest	非互联网 Post 请求次数。
Region.Data.NonInternetDeleteRequest	非互联网 Delete 请求次数。
Region.Data.NonInternetOtherRequest	非互联网除 Get、Head、Put、Post、Delete 外的其他请求次数。

- **请求示例**

查询 Bucket 名为 example-bucket，2018-12-24 的流量。

```
GET /?Action=GetUsage&BucketName=example-bucket&EndDate=2018-12-24&BeginDate=2018-12-24
HTTP/1.1
Host: oos-cn-mg.ctyunapi.cn
Authorization: SignatureValue
Date: Wed, 26 Dec 2018 09:01:30 GMT
Content-Type: application/xml; charset=utf-8
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id: 236A8905248E5A01
Date: Wed, 26 Dec 2018 09:01:30 GMT
Content-Length:1200
Content-Type: application/xml; charset=utf-8
Server: CTYUN

<GetUsageResponse>
<UserName>test@ctyun.cn</UserName>
<BucketName>example-bucket</BucketName>
  <Region>
      <Name>ZhengZhou</Name>
      <Data>
```

```
            <Date>2018-12-24</Date>
            <Capacity>3048576</Capacity>
            <Upload>40000</Upload>
            <Download>60000</Download>
            <DeleteFlow>0</DeleteFlow>
            <RoamUpload>20</RoamUpload>
            <RoamDownload>30</RoamDownload>
            <GetRequest>60</GetRequest>
            <HeadRequest>60</HeadRequest>
            <PutRequest>60</PutRequest>
            <PostRequest>60</PostRequest>
            <DeleteRequest>60</DeleteRequest>
            <OtherRequest>80</OtherRequest>
            <NonInternetUpload>0</NonInternetUpload>
            <NonInternetDownload>0</NonInternetDownload>
            <NonInternetDeleteFlow>0</NonInternetDeleteFlow>
            <NonInternetRoamUpload>0</NonInternetRoamUpload>
            <NonInternetRoamDownload>0</NonInternetRoamDownload>
            <NonInternetGeRequest>0</NonInternetGetRequest>
            <NonInternetHeadRequest>0</NonInternetHeadRequest>
            <NonInternetPutRequest>0</NonInternetPutRequest>
            <NonInternetPostRequest>0</NonInternetPostRequest>
            <NonInternetDeleteRequest>0</NonInternetDeleteRequest>
            <NonInternetOtherRequest>0</NonInternetOtherRequest>
      </Data>
    </Region>
    <Region>
        <Name>global</Name>
        <Data>
          ...
        </Data>
    </Region>
</GetUsageResponse>
```

5.3.10 GetBandwidth

此操作用来查询用户的已用带宽。

- **请求参数**

名称	描述	是否必须
Action	GetBandwidth。	是
BeginDate	指定查询已用带宽的起始时间（时区为：UTC+8）。 **类型**：Time **取值**：格式为 yyyy-MM-dd-HH-mm，mm 只能取 00、05、10.....55。日期早于当前时间不能超过一个月。	是
EndDate	指定查询已用带宽的结束时间（时区为：UTC+8），返回数据时间间隔为 5 分钟。 **类型**：Time **取值**：格式为 yyyy-MM-dd-HH-mm，mm 只能取 00、05、10.....55。EndDate 既不能早于 BeginDate，也不能晚于当前时间。	是
BucketName	指定查询已用带宽的 Bucket 名称，如果不指定 Bucket 名称，则表示查询账户下所有的 Bucket 的统计信息。 **类型**：字符串。 **取值**：3~63 个字符，只能由小写字母、数字、短横线（-）和点（.）组成。	否
NetType	指定返回的网络类型-精品网（CN2）/普通网（163）。 **注意**：仅香港节点支持本参数。 **类型**：字符串。 **取值**： ● **all**：指定返回精品网和普通网的带宽。 ● **highqualitynet**：指定返回精品网的带宽。 ● **normalqualitynet**：指定返回普通网的带宽。	否

	默认值为 **all**。	

- **响应结果**

名称	描述
UserName	发出请求的账户名。
BucketName	Bucket 名称。
NetType	网络类型-精品网（CN2）/普通网（163）： ● **all**：精品网和普通网。 ● **highqualitynet**：精品网。 ● **normalqualitynet**：普通网。 **注意**：仅香港节点会返回此项。
Region.Name	数据位置。
Region.Data.Date	统计数据时间。
Region.Data.UploadBW	互联网直接上行已用带宽，单位是 Byte/s。
Region.Data.DownloadBW	互联网直接下行已用带宽，单位是 Byte/s。
Region.Data.RoamUploadBW	互联网漫游上行已用带宽，单位是 Byte/s。
Region.Data.RoamDownloadBW	互联网漫游下行已用带宽，单位是 Byte/s。
Region.Data.NonInternetUploadBW	非互联网直接上行已用带宽，单位是 Byte/s。
Region.Data.NonInternetDownloadBW	非互联网直接下行已用带宽，单位是 Byte/s。
Region.Data.NonInternetRoamUploadBW	非互联网漫游上行已用带宽，单位是 Byte/s。
Region.Data.NonInternetRoamDownloadBW	非互联网漫游下行已用带宽，单位是 Byte/s。

- **请求示例**

查询 2018 年 12 月 26 日 16 点 45 分到 2018 年 12 月 26 日 16 点 50 分 example-bucket 的已用带宽。

```
GET /?Action=GetBandwidth&BucketName=example-bucket&EndDate=2018-12-26-16-
50&BeginDate=2018-12-26-16-45 HTTP/1.1
Host: oos-cn-mg.ctyunapi.cn
x-amz-content-sha256: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

```
Authorization: SignatureValue
x-amz-date: 20181226T090108Z
Content-Type: application/xml; charset=utf-8
```

- 响应示例

```
HTTP/1.1 200 OK
Date: Wed, 26 Dec 2018 09:01:30 GMT
x-amz-request-id: cabf8789cbe74b2e
Content-Length: 1694
Content-Type: application/xml; charset=utf-8
Server: CTYUN

<?xml version="1.0" encoding="UTF-8"?>
<GetBandwidthResult>
    <UserName>ctyuntest@ctyun.cn</UserName>
    <BucketName>example-bucket</BucketName>
    <Region>
        <Name>ShenZhen</Name>
        <Data>
            <Date>2018-12-26 16:45</Date>
            <UploadBW>1000</UploadBW>
            <DownloadBW>2000</DownloadBW>
            <RoamUploadBW>0</RoamUploadBW>
            <RoamDownloadBW>0</RoamDownloadBW>
            <NonInternetUploadBW>3000</NonInternetUploadBW>
            <NonInternetDownloadBW>4000</NonInternetDownloadBW>
            <NonInternetRoamUploadBW>0</NonInternetRoamUploadBW>
            <NonInternetRoamDownloadBW>0</NonInternetRoamDownloadBW>
        </Data>
        <Data>
            <Date>2018-12-26 16:50</Date>
            <UploadBW>5000</UploadBW>
            <DownloadBW>6000</DownloadBW>
            <RoamUploadBW>0</RoamUploadBW>
            <RoamDownloadBW>0</RoamDownloadBW>
            <NonInternetUploadBW>7000</NonInternetUploadBW>
            <NonInternetDownloadBW>8000</NonInternetDownloadBW>
            <NonInternetRoamUploadBW>0</NonInternetRoamUploadBW>
            <NonInternetRoamDownloadBW>0</NonInternetRoamDownloadBW>
        </Data>
    </Region>
```

```
    <Region>
        <Name>QingDao</Name>
        <Data><Date>2018-12-26 16:45</Date>
        <UploadBW>1000</UploadBW>
        <DownloadBW>2000</DownloadBW>
        <RoamUploadBW>0</RoamUploadBW>
        <RoamDownloadBW>0</RoamDownloadBW>
        <NonInternetUploadBW>3000</NonInternetUploadBW>
        <NonInternetDownloadBW>4000</NonInternetDownloadBW>
        <NonInternetRoamUploadBW>0</NonInternetRoamUploadBW>
        <NonInternetRoamDownloadBW>0</NonInternetRoamDownloadBW>
    </Data>
    <Data>
        <Date>2018-12-26 16:50</Date>
        <UploadBW>5000</UploadBW>
        <DownloadBW>6000</DownloadBW>
        <RoamUploadBW>0</RoamUploadBW>
        <RoamDownloadBW>0</RoamDownloadBW>
        <NonInternetUploadBW>7000</NonInternetUploadBW>
        <NonInternetDownloadBW>8000</NonInternetDownloadBW>
        <NonInternetRoamUploadBW>0</NonInternetRoamUploadBW>
        <NonInternetRoamDownloadBW>0</NonInternetRoamDownloadBW>
    </Data>
    </Region>
</GetBandwidthResult>
```

5.4 错误码列表

响应码	错误码(code)	错误信息(message)	描述
403	AccessDenied	User: *user* is not authorized to perform: *action* on resource: *resource*.	没有权限访问此资源。
405	MethodNotAllowed	Method does not exist or is not attachable.	方法不被允许。
500	InternalError	We encountered an internal error. Please try again.	发生内部错误。
400	InvalidArgument	Invalid BeginDate or EndDate on querying by day	按天查询时，时间请求参数不合法。
400	InvalidArgument	Invalid BeginDate or EndDate.	时间请求参数格式无效。
400	InvalidArgument	Invalid BeginDate and EndDate.	beginDate 不早于 endDate。
400	InvalidArgument	Invalid EndDate.	（旧接口）结束时间请求参数不合法，时间点不是整 5 分钟或 EndDate 晚于当前时刻。
400	InvalidArgument	Invalid BeginDate.	（旧接口）开始时间请求参数不合法，时间点不是整 5 分钟或 BeginDate 早于一个月前。
400	InvalidArgument	Invalid BeginDate and EndDate. Too large time gap.	beginDate 和 endDate 之间时间间隔不能大于 5 分钟（旧 api 获取连接数）。
400	InvalidArgument	Invalid Freq.	频率 Freq 请求参数格式不正确。

400	InvalidArgument	Invalid NetType.	网络类型（精品网/普通网）请求参数不合法。
400	InvalidArgument	Missing Freq.	没有输入参数 Freq。

6 操作跟踪 API

说明：以下举例中的域名均以对象存储网络的操作跟踪 API 域名 oos-cn-cloudtrail.ctyunapi.cn 为例。如果是对象存储网络 2，请使用域名 oos-cn2-cloudtrail.ctyunapi.cn。如果是香港精品网和香港普通网，请使用域名 oos-cnhk-cloudtrail.ctyunapi.cn。

6.1 操作跟踪 API 请求结构

- **接入地址**

对象存储网络：oos-cn-cloudtrail.ctyunapi.cn，对象存储网络 2：oos-cn2-cloudtrail.ctyunapi.cn，香港精品网和香港普通网：oos-cnhk-cloudtrail.ctyunapi.cn。

- **通信协议**

为了保证通信的安全性，操作跟踪仅支持 HTTPS。

- **HTTP 请求方法**

操作跟踪支持 HTTP POST 请求方法发送请求。

- **请求参数**

每个请求都需要指定如下信息：

- 每个操作接口都需要包含的公共请求参数。
- 操作接口所特有的请求参数。

本节主要描述公共请求参数和请求结果。

说明：在后续提到具体操作跟踪 API 时，举例中都会有**公共请求头**、**公共响应头**，但是不对其进行描述和解释。

6.1.1 公共请求头

在每个请求中，都需要携带公共参数和对应的接口参数。公共请求参数如表所示：

名称	描述	是否必须
Host	操作跟踪访问域名。 ● 对象存储网络：oos-cn-cloudtrail.ctyunapi.cn。 ● 对象存储网络 2：oos-cn2-cloudtrail.ctyunapi.cn。 ● 香港精品网和香港普通网：oos-cnhk-cloudtrail.ctyunapi.cn。	是

Authorization	V4 请求头签名。 **类型**：字符串。	是
X-Amz-Date	日期和时间格式必须遵循 ISO 8601 标准，并且必须使用“yyyyMMddT HHmmssZ”格式进行格式化。例如，如果日期和时间是“08/01/2018 15：32：41.982-700”，则必须首先将其转换为 UTC（协调世界时），然后提交为“20180801T083241Z”。	是
X-Amz-Target	操作跟踪请求目标，格式为 cn.ctyunapi.oos-cn-cloudtrail.v20131101.CloudTrail_20131101.*Target*。 例如创建操作跟踪的请求为： X-Amz-Target:cn.ctyunapi.oos-cn-cloudtrail.v20131101.CloudTrail_20131101.CreateTrail	是
Content-Type	请求内容类型。 **类型**：字符串。 **取值**：application/json	是
Connection	客户端与 OOS 服务器之间的连接状态。 取值： ● keep-alive：长连接，请求结束后继续保持连接。 ● close：短连接，请求结束后关闭连接。 默认值为：keep-alive。	否

6.1.2 公共响应头

每个操作跟踪 API 响应结果中都会包含公共响应头。

名称	描述
x-amz-request-id	服务端生成的用于标识请求的 ID。
Content-Type	响应内容类型。
Date	响应日期。
Server	服务器名。

Connection	客户端与 OOS 服务器之间的连接状态。 ● 如果请求时 Connection 值为 keep-alive，请求结束后继续保持连接，不返回此响应头。 ● 如果请求时 Connection 值为 close，请求结束后关闭连接，返回此响应头 Connection: close。

6.2 操作跟踪 API 概览

API	描述
CreateTrail	此操作用来创建一个跟踪，并将跟踪日志保存到指定的 OOS Bucket。
DeleteTrail	此操作用来删除指定的跟踪。
DescribeTrails	此操作用来获取跟踪的设置信息。
GetTrailStatus	此操作用来获取跟踪状态信息。
PutEventSelectors	此操作用来配置跟踪的管理事件筛选器。
GetEventSelectors	此操作用来查看管理事件筛选器的设置信息。
UpdateTrail	此操作用来更新跟踪设置参数。
StartLogging	此操作用来开启跟踪。
StopLogging	此操作用来停止跟踪。
LookupEvents	此操作用来查看账户中的管理事件。

6.3 操作跟踪 API

6.3.1 CreateTrail

此操作用来创建一个跟踪，并将跟踪日志保存到指定的 OOS Bucket。

注意：

- 一个账户最多创建 10 个跟踪。
- 使用 CreateTrail 创建跟踪后，跟踪默认是关闭状态，需要调用 StartLogging 开启跟踪。

- **请求参数**

名称	描述	是否必须
Name	指定跟踪的名称。 **类型：**字符串 **取值：**3~128 个字符。 ● 可以包含 ASCII 字母（a-z，A-Z），数字（0-9），句点（.），下划线（_）或短划线（-）。 ● 必须以字母或数字开头，以字母或数字结尾。 ● 不能是 IP 地址格式（例如：192.168.5.4）。 ● 不能包含相邻句点（.）、下划线（_）、短划线（-）任意组合。如不能包含类似点点（..）,点下划线（._）的组合。	是
S3BucketName	指定 OOS Bucket 名称。	是
S3KeyPrefix	指定跟踪日志的名称前缀。 ● 指定日志前缀的存储路径为：<BucketName>/<日志的名称前缀>/OOSLogs/<账号 ID>/CloudTrail/<年>/<月>/<日>/<日志数据文件>。 ● 未指定日志前缀的存储路径为：<BucketName>/OOSLogs/<账号 ID>/CloudTrail/<年>/<月>/<日>/<日志数据文件>。 **类型：**字符串 **取值：**0-200 个字符。	否

- **响应结果**

名称	描述
Name	跟踪的名称。
S3BucketName	OOS Bucket 名称。
S3KeyPrefix	跟踪日志的名称前缀。
TrailARN	跟踪的 ARN。

- **请求示例**

创建名称为 test_cloud_trail 的跟踪，生成的跟踪日志保存在名为 example-bucket 的 Bucket 中，日志前缀为 auditlog。

```
POST / HTTP/1.1
Host:oos-cn-cloudtrail.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190418T100110Z
X-Amz-Target:cn.ctyunapi.oos-cn-cloudtrail.v20131101.CloudTrail_20131101.CreateTrail
Content-Type:application/json

{"Name":"test_cloud_trail","S3BucketName":"example-bucket","S3KeyPrefix":"auditlog"}
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:ea219738be1a4688
Content-Type:application/json;charset=UTF-8
Content-Length:254
Date:Thu, 18 Apr 2019 10:01:10 GMT
Server:CTYUN

{"Name":"test_cloud_trail","S3BucketName":"example-
bucket","TrailARN":"arn:ctyun:cloudtrail::10rc2arpn6306:trail/test_cloud_trail","S3Key
Prefix":"auditlog"}
```

6.3.2 DeleteTrail

此操作用来删除指定的跟踪。

- **请求参数**

名称	描述	是否必须
Name	指定跟踪的名称。 **类型**：字符串 **取值**：3~128 个字符。 ● 可以包含 ASCII 字母（a-z，A-Z），数字（0-9），句点（.），下划线（_）或短划线（-）。 ● 必须以字母或数字开头，以字母或数字结尾。 ● 不能是 IP 地址格式（例如：192.168.5.4）。 ● 不能包含相邻句点（.）、下划线（_）、短划线（-）任意组合。如不能包含类似点点（..）,点下划线（._）的组合。	是

- **请求示例**

删除名为 test_cloud_trail 的跟踪。

```
POST / HTTP/1.1
Host:oos-cn-cloudtrail.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190419T053842Z
X-Amz-Target:cn.ctyunapi.oos-cn-cloudtrail.v20131101.CloudTrail_20131101.DeleteTrail
Content-Type:application/json

{"Name":"test_cloud_trail"}
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:48e6e27fa7a34a55
Date:Fri, 19 Apr 2019 05:38:42 GMT
Server:CTYUN
```

6.3.3 DescribeTrails

此操作用来获取操作跟踪的设置信息。

- **请求参数**

名称	描述	是否必须
TrailNameList	指定跟踪的名称列表。可以指定一个或多个跟踪的名称。 **类型**：字符串 **取值**：3~128 个字符。 ● 可以包含 ASCII 字母（a-z，A-Z），数字（0-9），句点（.），下划线（_）或短划线（-）。 ● 必须以字母或数字开头，以字母或数字结尾。 ● 不能是 IP 地址格式（例如：192.168.5.4）。 ● 不能包含相邻句点（.）、下划线（_）、短划线（-）任意组合。如不能包含类似点点（..）,点下划线（._）的组合。 **说明**： ● 如果名称列表为空{"TrailNameList":[]}，则返回所有跟踪。 ● 如果指定的任一 trail 名称不存在，则不返回此 trail。 ● 如果指定的所有 trail 名称都不存在，则返回空列表，即{"trailList":[]}。	是

- **响应结果**

名称	描述
trailList.Name	跟踪的名称。

trailList.S3BucketName	OOS Bucket 名称。
trailList.S3KeyPrefix	跟踪日志的名称前缀。
trailList.TrailARN	跟踪的 ARN。

- **请求示例**

获取名称为 test_cloud_trail 的跟踪设置信息。

```
POST / HTTP/1.1
Host:oos-cn-cloudtrail.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190419T073339Z
X-Amz-Target:cn.ctyunapi.oos-cn-cloudtrail.v20131101.CloudTrail_20131101.DescribeTrails
Content-Type:application/json

{"TrailNameList":["test_cloud_trail"]}
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:570a1b70a73f4b3c
Content-Type:application/json;charset=UTF-8
Content-Length:325
Date:Fri, 19 Apr 2019 07:33:39 GMT
Server:CTYUN

{"trailList":[{"Name":"test_cloud_trail","S3BucketName":"test_bucket","TrailARN":"arn:
ctyun:cloudtrail::10rc2arpn6306:trail/test_cloud_trail"}]}
```

6.3.4 GetTrailStatus

此操作用来获取跟踪状态信息。

- **请求参数**

名称	描述	是否必须
Name	指定跟踪的名称。 **类型**：字符串 **取值**：3~128 个字符。 ● 可以包含 ASCII 字母（a-z，A-Z），数字（0-9），句点（.），下划线（_）或短划线（-）。 ● 必须以字母或数字开头，以字母或数字结尾。 ● 不能是 IP 地址格式（例如：192.168.5.4）。 ● 不能包含相邻句点（.）、下划线（_）、短划线（-）任意组合。如不能包含类似点点（..）,点下划线（._）的组合。	是

- **响应结果**

名称	描述
IsLogging	是否开启跟踪： ● `true`：已经开启跟踪。 ● `false`：未开启跟踪。
LatestDeliveryTime	跟踪最近一次向 Bucket 保存日志的时间。 说明：如果没有开启过跟踪，则不会返回此字段。
StartLoggingTime	最后一次跟踪的起始时间。unix 时间戳（UTC），时间精确到毫秒。 说明：如果没有开启过跟踪，则不会返回此字段。
StopLoggingTime	最后一次跟踪的停止时间。unix 时间戳（UTC），时间精确到毫秒。 说明：如果没有开启过跟踪，则不会返回此字段。
LatestDeliveryError	最近一次向 Bucket 保存日志时遇到的错误。如果没有错误，不返回此项。

	● 当目标 Bucket 不存在时，返回：NoSuchBucket。 ● 当目标 Bucket 已无权访问，返回：AccessDenied。

- **请求示例**

获取跟踪名为 test_cloud_trail 的跟踪状态信息。

```
POST / HTTP/1.1
Host:oos-cn-cloudtrail.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190423T021257Z
X-Amz-Target:cn.ctyunapi.oos-cn-cloudtrail.v20131101.CloudTrail_20131101.GetTrailStatus
Content-Type:application/json

{"Name":"test_cloud_trail"}
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amzn-RequestId:12feb8d48aa84d09
Content-Type:application/json;charset=UTF-8
Content-Length:386
Date:Tue, 23 Apr 2019 02:12:57 GMT
Server:CTYUN

{"IsLogging":true,"LatestDeliveryTime":1555985394434,"StartLoggingTime":1555984663079,
"StopLoggingTime":1555987231038}
```

6.3.5 PutEventSelectors

此操作用来配置跟踪的管理事件筛选器。

默认情况下，未设置管理事件筛选器的跟踪，会记录所有的管理事件，每个跟踪最多创建 1 个管理事件筛选器。

- **请求参数**

名称	描述	是否必须
TrailName	指定跟踪的名称。 **类型**：字符串 **取值**：3~128 个字符。 ● 可以包含 ASCII 字母（a-z，A-Z），数字（0-9），句点（.），下划线（_）或短划线（-）。 ● 必须以字母或数字开头，以字母或数字结尾。 ● 不能是 IP 地址格式（例如：192.168.5.4）。 ● 不能包含相邻句点（.）、下划线（_）、短划线（-）任意组合。如不能包含类似点点（..）,点下划线（._）的组合。	是
EventSelectors	指定管理事件筛选器。 **取值**：ReadWriteType	是
ReadWriteType	指定管理事件的读写类型。 **类型**：字符串 **取值**： ● **ReadOnly**：只读。 ● **WriteOnly**：只写。 ● **All**：所有管理事件。 默认值为 **All**。	否

- **响应结果**

名称	描述

EventSelectors.ReadWriteType	管理事件的读写类型： ● **ReadOnly**：只读。 ● **WriteOnly**：只写。 ● **All**：所有管理事件。
TrailARN	跟踪的 ARN。

- **请求示例**

为跟踪 test_cloud_trail 创建只写（WriteOnly）类型跟踪的管理事件筛选器。

```
POST / HTTP/1.1
Host:oos-cn-cloudtrail.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190422T032847Z
X-Amz-Target:cn.ctyunapi.oos-cn-
cloudtrail.v20131101.CloudTrail_20131101.PutEventSelectors
Content-Type:application/json

{"TrailName":"test_cloud_trail","EventSelectors":[{"ReadWriteType":"WriteOnly"}]}
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:771c9193addb4861
Content-Type:application/json;charset=UTF-8
Content-Length:231
Date:Mon, 22 Apr 2019 03:28:47 GMT
Server:CTYUN

{"EventSelectors":[{"ReadWriteType":"WriteOnly"}],"TrailARN":"arn:ctyun:cloudtrail::10
rc2arpn6306:trail/test_cloud_trail"}
```

6.3.6 GetEventSelectors

此操作用来查看管理事件筛选器的设置信息。

- **请求参数**

名称	描述	是否必须
TrailName	跟踪的名称。 **类型**：字符串 **取值**：3~128 个字符。 ● 可以包含 ASCII 字母（a-z，A-Z），数字（0-9），句点（.），下划线（_）或短划线（-）。 ● 必须以字母或数字开头，以字母或数字结尾。 ● 不能是 IP 地址格式（例如：192.168.5.4）。 ● 不能包含相邻句点（.）、下划线（_）、短划线（-）任意组合。如不能包含类似点点（..）,点下划线（._）的组合。	是

- **响应结果**

名称	描述
EventSelectors.ReadWriteType	管理事件的读写类型： ● **ReadOnly**：只读。 ● **WriteOnly**：只写。 ● **All**：所有管理事件。
TrailARN	跟踪的 ARN。

- **请求示例**

查看名称为 test_cloud_trail 的跟踪管理事件筛选器信息。

```
POST / HTTP/1.1
Host:oos-cn-cloudtrail.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190422T053340Z
```

```
X-Amz-Target:cn.ctyunapi.oos-cn-
cloudtrail.v20131101.CloudTrail_20131101.GetEventSelectors
Content-Type:application/json

{"TrailName":"test_cloud_trail"}
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:a97ab880f40c4242
Content-Type:application/json;charset=UTF-8
Content-Length:231
Date:Mon, 22 Apr 2019 05:33:40 GMT
Server:CTYUN

{"EventSelectors":[{"ReadWriteType":"WriteOnly"}],"TrailARN":"arn:ctyun:cloudtrail::10
rc2arpn6306:trail/test_cloud_trail"}
```

6.3.7 UpdateTrail

此操作用来更新跟踪设置参数，包括跟踪日志数据保存的 OOS Bucket、跟踪日志前缀等。

说明：

- 更新跟踪时，无需暂停跟踪。
- 如果 S3BucketName 和 S3KeyPrefix 都未指定，该操作不起作用，但是不报错，返回 200 OK。

- **请求参数**

名称	描述	是否必须
Name	跟踪的名称。 **类型**：字符串 **取值**：3~128 个字符。 ● 可以包含 ASCII 字母（a-z，A-Z），数字（0-9），句点（.），下划线（_）或短划线（-）。 ● 必须以字母或数字开头，以字母或数字结尾。 ● 不能是 IP 地址格式（例如：192.168.5.4）。 ● 不能包含相邻句点（.）、下划线（_）、短划线（-）任意组合。如不能包含类似点点（..）,点下划线（._）的组合。	是
S3BucketName	OOS Bucket 名称。	否
S3KeyPrefix	指定跟踪日志的名称前缀。 **类型**：字符串 **取值**：0-200 个字符。	否

- **响应结果**

名称	描述
Name	跟踪的名称。
S3BucketName	OOS Bucket 名称。
S3KeyPrefix	跟踪日志的名称前缀。
TrailARN	跟踪的 ARN。

- **请求示例**

  更新名为 test_cloud_trail 的跟踪信息，将跟踪日志前缀更新为 auditLog2。

```
POST / HTTP/1.1
Host:oos-cn-cloudtrail.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190422T092219Z
X-Amz-Target:cn.ctyunapi.oos-cn-cloudtrail.v20131101.CloudTrail_20131101.UpdateTrail
Content-Type:application/json

{"Name":"test_cloud_trail","S3KeyPrefix":"auditLog2"}
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amzn-RequestId:6a452ca08cad49b5
Content-Type:application/json;charset=UTF-8
Content-Length:278
Date:Mon, 22 Apr 2019 09:22:19 GMT
Server:CTYUN

{"Name":"test_cloud_trail","S3BucketName":"bucket_name","S3KeyPrefix":"auditLog2","Tra
ilARN":"arn:ctyun:cloudtrail::10rc2arpn6306:trail/test_cloud_trail"}
```

6.3.8 StartLogging

此操作用来开启跟踪。

说明：使用 CreateTrail 创建跟踪后，跟踪默认是关闭状态，需要调用 StartLogging 开启跟踪。

- **请求参数**

名称	描述	是否必须
Name	跟踪的名称。 **类型：**字符串 **取值：**3~128 个字符。 ● 可以包含 ASCII 字母（a-z，A-Z），数字（0-9），句点（.），下划线（_）或短划线（-）。 ● 必须以字母或数字开头，以字母或数字结尾。 ● 不能是 IP 地址格式（例如：192.168.5.4）。 ● 不能包含相邻句点（.）、下划线（_）、短划线（-）任意组合。如不能包含类似点点（..）,点下划线（._）的组合。	是

- **请求示例**

启动名为 test_cloud_trail 的跟踪。

```
POST / HTTP/1.1
Host:oos-cn-cloudtrail.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190422T054520Z
X-Amz-Target:cn.ctyunapi.oos-cn-cloudtrail.v20131101.CloudTrail_20131101.StartLogging
Content-Type:application/json

{"Name":"test_cloud_trail"}
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:e543b10f6249483c
Date:Mon, 22 Apr 2019 05:45:20 GMT
Server:CTYUN
```

6.3.9 StopLogging

此操作用来停止跟踪功能。

- **请求参数**

名称	描述	是否必须
Name	跟踪的名称。 **类型**：字符串 **取值**：3~128 个字符。 ● 可以包含 ASCII 字母（a-z，A-Z），数字（0-9），句点（.），下划线（_）或短划线（-）。 ● 必须以字母或数字开头，以字母或数字结尾。 ● 不能是 IP 地址格式（例如：192.168.5.4）。 ● 不能包含相邻句点（.）、下划线（_）、短划线（-）任意组合。如不能包含类似点点（..）,点下划线（._）的组合。	是

- **请求示例**

  停止名为 test_cloud_trail 的跟踪。

```
POST / HTTP/1.1
Host:oos-cn-cloudtrail.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190422T070134Z
X-Amz-Target:cn.ctyunapi.oos-cn-cloudtrail.v20131101.CloudTrail_20131101.StopLogging
Content-Type:application/json

{"Name":"test_cloud_trail"}
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:734105e4f0d740bf
Date:Mon, 22 Apr 2019 07:01:34 GMT
Server:CTYUN
```

6.3.10 LookupEvents

此操作用来查看账户中的管理事件。用户可以查看近 6 个月内发生的管理事件。

支持查询以下属性：

- OOS 访问密钥。
- 管理事件 ID。
- 管理事件名称。
- 管理事件来源。
- 是否只读。
- 资源名称。
- 资源类型。
- 子用户名。

所有的属性都是可选项，默认返回 50 个查询结果，一次最多只能返回 50 个结果，如果想查看其他结果，可以通过设置 MaxResults 和 NextToken 来进行查看。

- **请求参数**

名称	描述	是否必须
StartTime	指定返回管理事件的起始时间。如果同时指定了起始时间和结束时间，则起始时间必须早于结束时间，否则将返回错误信息。 **类型**：时间戳 **取值**：unix 时间戳（UTC），精确到毫秒。默认起始时间为距当前时间往前数的第 184 天。 ● 如果起始时间晚于结束时间，会报错。 ● 如果起始时间是早于当前时间 184 天之前的时间，则返回数据的起始时间为距当前时间往前数的第 184 天。	否
EndTime	指定返回管理事件的结束时间。如果同时指定了起始时间和结束时间，则起始时间必须早于结束时间，否则将返回错误信息。	否

	类型：时间戳 **取值**：unix 时间戳（UTC），精确到毫秒。默认结束时间为目前时间。 如果 StartTime 和 EndTime 都为早于当前时间 184 天前的时间，则返回距当前时间往前数的第 184 天那天的数据。	
LookupAttributes	指定管理事件的查询属性。 **类型**：数组。 **取值**：Attributekey 和 AttributeValue 成对出现。	否
AttributeKey	指定查询管理事件的属性 Key。 **注意**：如果指定 AttributeKey，每次只能指定一个 **AttributeKey，且取值唯一**。 **类型**：字符串 **取值**： ● EventId ● EventName ● ReadOnly ● UserName ● ResourceType ● ResourceName ● EventSource ● AccessKeyId	否
AttributeValue	指定 AttributeKey 对应的值。 **类型**：字符串 **取值**：根据 AttributeKey 确定。其中 ResourceName 根据前缀匹配查询，区分大小写。EventId、UserName、EventName、ResourceType、EventSource、AccessKeyId 全字匹配查询，并且区分大小写。	否

MaxResults	设置响应中最多返回的条数。如果存在超出您指定的返回项，响应结果中会返回 NextToken 的值。 **类型**：整数类型。 **取值**：1-50，默认值为 50。	否
NextToken	分页标识。还有需要返回的管理事件时，上条响应结果中会返回该参数。查看未显示项时，请求参数中需要携带此参数。 **类型**：字符串 **取值**：与上条响应结果中的参数值一样。	否

- **响应结果**

名称	描述
Events	根据查询条件，返回的管理事件。
NextToken	分页标识。如果没有返回该值，则说明已经返回所有的查询结果。

- **请求示例**

查询账户中的只读事件。

```
POST / HTTP/1.1
Host:oos-cn-cloudtrail.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20240725T092748Z
X-Amz-Target:cn.ctyunapi.oos-cn-cloudtrail.v20131101.CloudTrail_20131101.LookupEvents
Content-Type:application/json
Content-Length: 153

{
    "MaxResults": 2,
    "LookupAttributes": [
        {
            "AttributeKey": "ReadOnly",
            "AttributeValue": "true"
        }
```

```
    ]
}
```

- 响应示例

```
HTTP/1.1 200 OK
x-amz-request-id: f7d150a0f43b4681
Content-Type: application/json; charset=utf-8
Content-Length: 1498
Date: Thu, 25 Jul 2024 09:27:50 GMT
Server:CTYUN

{
    "NextToken": "1817854200671071211|7501472477088833290",
    "Events": [
        {
            "CloudTrailEvent": {
                "eventId": "7501472382852511156",
                "eventVersion": "1.06",
                "eventSource": "oos-cn-cloudtrail.ctyunapi.cn",
                "requestParameters": {
                    "MaxResults": 2,
                    "LookupAttributes": [
                        {
                            "AttributeKey": "ReadOnly",
                            "AttributeValue": "true"
                        }
                    ]
                },
                "userAgent": "PostmanRuntime/7.6.0",
                "readOnly": true,
                "userIdentity": {
                    "accessKeyId": "6c1ceb071a848d7d14d5",
                    "accountId": "0dt7bj6yazfmz",
                    "principalId": "0dt7bj6yazfmz",
                    "type": "Root",
                    "arn": "arn:ctyun:iam::0dt7bj6yazfmz:root"
                },
                "eventType": "ApiCall",
                "serviceName": "CloudTrail",
                "sourceIp": "10.183.14.70",
                "requestId": "40a279e6142e4a3f",
```

```
            "requestURL": "http://oos-cn-cloudtrail.ctyunapi.cn/",
            "eventTime": "2024-07-25T09:27:34Z",
            "eventName": "LookupEvents",
            "requestRegion": "cn",
            "managementEvent": true
        }
    },
    {
        "CloudTrailEvent": {
            "eventId": "7501472428925466693",
            "eventVersion": "1.06",
            "eventSource": "oos-cn-cloudtrail.ctyunapi.cn",
            "requestParameters": {
                "LookupAttributes": [
                    {
                        "AttributeKey": "ReadOnly",
                        "AttributeValue": "true",
                        "MaxResults": 2
                    }
                ]
            },
            "userAgent": "PostmanRuntime/7.6.0",
            "readOnly": true,
            "userIdentity": {
                "accessKeyId": "6c1ceb071a848d7d14d5",
                "accountId": "0dt7bj6yazfmz",
                "principalId": "0dt7bj6yazfmz",
                "type": "Root",
                "arn": "arn:ctyun:iam::0dt7bj6yazfmz:root"
            },
            "eventType": "ApiCall",
            "serviceName": "CloudTrail",
            "sourceIp": "10.183.14.70",
            "requestId": "5730fff074354dd9",
            "requestURL": "http://oos-cn-cloudtrail.ctyunapi.cn/",
            "eventTime": "2024-07-25T09:26:47Z",
            "eventName": "LookupEvents",
            "requestRegion": "cn",
            "managementEvent": true
        }
    }
]
```

```
}
```

6.4 错误码列表

响应码	错误码(code)	错误信息(message)	错误描述
400	InvalidTrailNameException	Trail name too short. Minimum allowed length: 3 characters. Specified name length: 2 characters.	跟踪的名称非法，字符长度不足。
400	InvalidTrailNameException	Trail name too long. Maximum allowed length: 128 characters. Specified name length: 140 characters.	跟踪的名称非法，超过最大字符长度。
400	TrailAlreadyExistsException	Trail *trailName* already exists for the customer: *accountId*	跟踪已经存在。
400	TrailNotFoundException	Unknown trail: arn:ctyun:CloudTrail:ctyun:accountId:trail/*trailName* for the customer: *accountId*	跟踪不存在。
400	MaximumNumberOfTrailsExceededException	User: *accountId* already has 10 trails.	跟踪超过最大数量限制。
400	S3BucketDoesNotExistException	Bucket name does not exist: *bucketName*	创建跟踪时，bucketName 不存在。
400	InvalidTrailNameException	Trail name cannot be blank!	跟踪名称为空。
400	InvalidS3PrefixException	S3 prefix is longer than maximum length: *s3Prefix*	前缀超过长度限制。

400	InvalidLookupAttributesException	Lookup attribute key is not valid.	Lookup 属性 key 非法。
400	InvalidLookupAttributesException	Lookup attribute value is not valid.	Lookup 属性值非法。
400	InvalidMaxResultsException	A max results value of {maxNumber} is invalid. Specify max results between 1 and 50.	设置的 lookupEvents 返回的最大结果数超过限制。
400	InvalidEventSelectorsException	Specify a valid number of selectors for your trail	一个管理事件追踪的筛选器只能有一个。
403	AccessDenied	Access Denied	权限认证不通过。
403	AccessDenied	Can not access bucket {bucketName}	创建 trail 时，指定的 Bucket 不是当前账户下的 Bucket。
400	InvalidTrailNameException	Trail name must not be formatted as an IP address.	trail 名称不合法，是 ip 格式。
400	InvalidTrailNameException	Trail name must end with a letter or number.	trail 名称不合法，不是以数字或字母结尾。
400	InvalidTrailNameException	Trail name must start with a letter or number.	trail 名称不合法，不是以数字或字母开头。
400	InvalidTrailNameException	Trail name or ARN cannot have adjacent periods (.), hyphens (-), or underscores (_)."	tail 名称不合法，包含连续的“.”“-”“_”。

400	InvalidEventSelectorsException	Value {ReadWriteType} for ReadWriteType is invalid.	创建筛选器时，ReadWriteType 的值不正确。
400	InvalidTimeRangeException	The start time precedes the end time.	Lookup 时，起始时间晚于结束时间。
500	InternalError	We encountered an internal error. Please try again.	发生内部错误，请重试。
405	MethodNotAllowed	Method is not allowed.	操作不能被识别。
400	InvalidTrailNameException	Trail name or ARN can only contain uppercase letters, lowercase letters, numbers, periods (.), hyphens (-), and underscores (_).	trail名称不合法，包含非法字符。
400	InvalidS3BucketNameException	Bucket name cannot be blank!	创建trail时，Bucket名称不能为空。
400	InvalidLookupAttributesException	Lookup attribute key can not be multiple.	lookup属性key只能设置一个。

6.5 操作跟踪记录事件列表

类别	事件
Bucket	DeleteBucket
	DeleteBucketLifecycle
	GetBucketLifecycle
	GetBucketLocation
	CreateBucket
	PutBucketLifecycle
	PutBucketLogging
	GetBucketAcl
	PutBucketAcl
	GetBucketPolicy
	PutBucketPolicy
	DeleteBucketPolicy
	GetBucketWebsite
	PutBucketWebsite
	DeleteBucketWebsite
	GetBucketLogging
	GetBucketCors
	PutBucketCors
	DeleteBucketCors
	PutBucketObjectLockConfiguration
	GetBucketObjectLockConfiguration
	DeleteBucketObjectLockConfiguration
	PutBucketInventoryConfiguration
	GetBucketInventoryConfiguration
	DeleteBucketInventoryConfiguration

Services	GetService
	GetRegions
统计	GetCapacity
	GetBilledStorageUsage
	GetRestoreCapacity
	GetDeleteCapacity
	GetTraffics
	GetRequests
	GetReturnCode
	GetConcurrentConnection
	GetUsage
	GetBandwidth
控制台	ConsoleLogin
	LogoutUser
	CheckMfa
操作跟踪	CreateTrail
	DeleteTrail
	DescribeTrails
	GetTrailStatus
	PutEventSelectors
	GetEventSelectors
	UpdateTrail
	StartLogging
	StopLogging
	LookupEvents
访问控制	CreateGroup
	DeleteGroup

	GetGroup
	ListGroups
	AddUserToGroup
	RemoveUserFromGroup
	CreateUser
	DeleteUser
	GetUser
	ListUsers
	ListUserTags
	ListGroupsForUser
	CreateAccessKey
	ListAccessKeys
	GetAccessKeyLastUsed
	UpdateAccessKey
	DeleteAccessKey
	GetSessionToken
	TagUser
	ChangePassword
	CreateLoginProfile
	CreateVirtualMFADevice
	DeactivateMFADevice
	DeleteAccountPasswordPolicy
	DeleteLoginProfile
	DeleteVirtualMFADevice
	EnableMFADevice
	GetAccountPasswordPolicy
	GetLoginProfile

	ListVirtualMFADevices
	UpdateAccountPasswordPolicy
	UpdateAccountLoginSecurityPolicy
	GetAccountLoginSecurityPolicy
	DeleteAccountLoginSecurityPolicy
	UpdateLoginProfile
	CreatePolicy
	DeletePolicy
	AttachGroupPolicy
	DetachGroupPolicy
	GetPolicy
	ListAttachedUserPolicies
	AttachUserPolicy
	ListAttachedGroupPolicies
	ListPolicies
	GetAccountSummary
	DetachUserPolicy
	ListEntitiesForPolicy
	UnTagUser
	ListMFADevices

7 访问控制（IAM）API

说明：以下举例中的域名均以对象存储网络的访问控制（IAM）API 域名 oos-cn-iam.ctyunapi.cn 为例。如果是对象存储网络 2，请使用域名 oos-cn2-iam.ctyunapi.cn。如果是香港精品网和香港普通网，请使用域名 oos-cnhk-iam.ctyunapi.cn。

7.1 IAM API 请求结构

- **接入地址**

对象存储网络接入地址为：oos-cn-iam.ctyunapi.cn，对象存储网络 2 接入地址为：oos-cn2-iam.ctyunapi.cn，香港精品网和香港普通网的接入地址为 oos-cnhk-iam.ctyunapi.cn。

- **通信协议**

为了保证通信的安全性，IAM 仅支持 HTTPS。

- **请求方法**

IAM 支持 POST 请求方法发送请求。

- **请求参数**

每个请求都需要指定如下信息：

- 要执行的操作：Action 参数。
- 每个操作接口都需要包含的公共请求参数。
- 操作接口所特有的请求参数。

注意：请求的参数都需要 url encode，服务端进行 url decode。

- **字符编码**

请求及返回结果都使用 UTF-8 字符集进行编码。

7.1.1 公共参数

本节主要描述公共请求参数和请求结果。

说明：在后续提到具体 IAM API 时，举例中都会有**公共请求头、公共响应头、公共响应结果参数**，但是不对其进行描述和解释。

- **公共请求头**

在每个请求中，都需要携带公共参数和对应的接口参数。公共请求参数如表所示：

名称	描述	是否必须
Host	IAM 访问域名： ● 对象存储网络：oos-cn-iam.ctyunapi.cn。 ● 对象存储网络 2：oos-cn2-iam.ctyunapi.cn。 ● 香港精品网和香港普通网：oos-cnhk-iam.ctyunapi.cn。	是
Authorization	请求头签名。 支持 V4 签名认证。 **类型：**字符串	是
X-Amz-Date	日期和时间格式必须遵循 ISO 8601 标准，并且必须使用“yyyyMMddTHHmmssZ”格式进行格式化。例如，如果日期和时间是“08/01/2018 15：32：41.982-700”，则必须首先将其转换为 UTC（协调世界时），然后提交为“20180801T083241Z”。	是
Content-Type	请求内容类型。 **类型：**字符串 **取值：**application/octet-stream	是
Connection	客户端与 OOS 服务器之间的连接状态。 取值： ● keep-alive：长连接，请求结束后继续保持连接。 ● close：短连接，请求结束后关闭连接。 默认值为：keep-alive。	否

- **公共响应头**

每个 IAM API 响应结果中都会包含公共响应头。

名称	描述
x-amz-request-id	服务端生成的用于标识请求的 ID。
Content-Type	响应内容类型。
Date	响应日期。
Server	服务器名。
Content-Length	响应体的长度，单位为 Byte。
Connection	客户端与 OOS 服务器之间的连接状态。 ● 如果请求时 Connection 值为 keep-alive，请求结束后继续保持连接，不返回此响应头。 ● 如果请求时 Connection 值为 close，请求结束后关闭连接，返回此响应头 Connection: close。

7.1.2 响应结果

调用 IAM API 后返回数据采用统一格式，响应结果格式为 XML 格式。本文档中的响应示例为了便于用户查看，做了格式化处理，实际返回结果是没有进行换行、缩进等处理的。

- **成功响应结果**

调用 IAM API 成功后，如果响应结果 HTTP 状态为：200 OK，代表 IAM API 调用成功。示例如下：

```
HTTP/1.1 200 OK
<公共响应头>

<API 具体响应结果>
<公共响应参数>
```

- **失败响应结果**

调用 IAM API 失败后，如果响应结果 HTTP 状态为：4xx 或者 5xx，代表调用失败。示例如下：

```
HTTP/1.1 4xx 或者 5xx
<公共响应头>

<API 具体响应结果>
<公共响应参数>
```

7.2 访问控制 API 概览

列出所有 IAM 可以使用的 API 接口及相关描述。

- 用户管理接口

API	描述
CreateUser	此操作用来创建新的 IAM 用户。
GetUser	此操作用户查看 IAM 用户信息。
ListUsers	此操作用来列出 IAM 用户。
DeleteUser	此操作用来删除指定的 IAM 用户。
TagUser	此操作用来为 IAM 用户添加标签。
UntagUser	此操作用来删除用户的指定标签。
ListUserTags	此操作用来列出指定 IAM 用户的标签。
ListGroupsForUser	此操作用来列出指定 IAM 用户所属的 IAM 用户组。
CreateAccessKey	此操作用来为指定的 IAM 用户创建新的 AccessKey。
ListAccessKeys	此操作用来返回指定 IAM 用户的 AccessKey 的详细信息。
GetAccessKeyLastUsed	此操作用来查询指定密钥最后一次使用的时间及服务名称。
UpdateAccessKey	此操作用来更新指定访问密钥的状态，从 Active 到 Inactive，或者从 Inactive 到 Active。
DeleteAccessKey	此操作用来删除指定 IAM 用户关联的 AccessKey。
GetSessionToken	此操作用来为用户提供临时授权访问。
CreateLoginProfile	此操作用来为指定 IAM 用户创建控制台的登录密码。
GetLoginProfile	此操作用来获取 IAM 用户控制台登录密码创建的时间、及用户首次登录后再次登录是否需要修改密码。
UpdateLoginProfile	此操作用来更改指定 IAM 用户控制台的登录密码。
DeleteLoginProfile	此操作用来删除指定 IAM 用户控制台的登录密码。
ChangePassword	此操作用来修改 IAM 用户控制台的登录密码。
CreateVirtualMFADevice	此操作用来创建虚拟 MFA 设备。
EnableMFADevice	此操作用来启用指定的虚拟 MFA 设备，并将该虚拟 MFA 设备与指定的 IAM 用户关联。

ListVirtualMFADevices	此操作用来按分配状态列出 OOS 账户中定义的虚拟 MFA 设备。
ListMFADevices	此操作用来列出 IAM 用户的虚拟 MFA 设备。
DeactivateMFADevice	此操作用来去激活指定的 MFA 设备，并与用户解除关联。
DeleteVirtualMFADevice	此操作用来删除指定的虚拟 MFA 设备。

- **用户组管理接口**

API	描述
CreateGroup	此操作用来创建新的 IAM 用户组。
GetGroup	此操作用来获取指定 IAM 用户组及组内 IAM 用户列表。
AddUserToGroup	此操作用来将指定的 IAM 用户加入到指定的 IAM 用户组，每次只能将一个用户加入到指定用户组。
RemoveUserFromGroup	此操作用来将指定用户从指定用户组移除。
ListGroups	此操作用来列出所有的 IAM 用户组。
DeleteGroup	此操作用来删除指定的 IAM 用户组。

- **权限策略管理接口**

API	描述
CreatePolicy	此操作用来为账户创建策略。
GetPolicy	此操作用来获取策略相关信息。
ListPolicies	此操作用来列出账户下所有的策略。
ListEntitiesForPolicy	此操作用来列出指定策略所附加的所有 IAM 用户或 IAM 用户组。
DeletePolicy	此操作用来删除指定的策略。
AttachUserPolicy	此操作用来将指定的策略与指定的 IAM 用户关联。
ListAttachedUserPolicies	此操作用来列出与指定用户关联的策略。
DetachUserPolicy	此操作用来解除指定用户关联的指定策略。
AttachGroupPolicy	此操作用来将指定的策略与指定的 IAM 用户组关联。

ListAttachedGroupPolicies	此操作用来列出与指定 IAM 用户组关联的策略。
DetachGroupPolicy	此操作用来解除指定 IAM 用户组关联的指定策略。
UpdateAccountPasswordPolicy	此操作用来更新账户的密码规则设置。
GetAccountPasswordPolicy	此操作用来获取账户的密码策略。
DeleteAccountPasswordPolicy	此操作用来将账户的密码规则恢复到默认密码规则。
UpdateAccountLoginSecurityPolicy	此操作用来更新 IAM 用户登录安全策略设置。
GetAccountLoginSecurityPolicy	此操作用来获取 IAM 用户登录安全策略。
DeleteAccountLoginSecurityPolicy	此操作用来将 IAM 用户登录安全策略恢复为默认值。

服务数量查询接口

API	描述
GetAccountSummary	此操作用来获取账户中的实体数量和服务限制信息。

7.3 用户管理接口

7.3.1 CreateUser

此操作用来创建新的 IAM 用户。

- **请求参数**

名称	描述	是否必须
Action	CreateUser。	是
Version	请求版本。 **取值**：2010-05-08。默认值为 2010-05-08。	否
UserName	创建 IAM 用户的用户名，用户名在账户中必须唯一。 **类型**：字符串 **取值**：1~64 个字符组成，字符只能包含字母、数字或特殊字符，字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	是
Tags.member.N.Key& Tags.member.N.Value	附加到新创建 IAM 用户的标签列表。标签包括标签键和值。Tags.member.N.Key 为标签键，Tags.member.N.Value 为标签值。 **类型**：字符串 **取值**：Tags.member.N.Key 由 1~128 个字符组成，字符串符合表达式[\p{L}\p{Z}\p{N}_.:/=+\-@]+。 Tags.member.N.Value 由 0~256 个字符组成，字符串符合表达式[\p{L}\p{Z}\p{N}_.:/=+\-@]*。	否

	注意：最多包含 10 个标签。如果任何一个标签无效或者如果超过每个用户允许的标签数，则整个请求将失败，并且不会创建用户。	

- **响应结果**

名称	描述
Arn	IAM 用户的资源名称。
UserName	IAM 用户的名称。
UserId	IAM 用户 ID。
Tags.member.Key	IAM 用户标签键。
Tags.member.Value	IAM 用户标签值。
CreateDate	IAM 用户创建时间。

- **请求示例**

创建用户名为 test_user，标签键为 test_key，标签值为 test_value 的 IAM 用户。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190131T012759Z
Content-Type: application/octet-stream
Content-Length: 129

Action=CreateUser&Version=2010-05-
08&UserName=test_user&Tags.member.1.Key=test_key&Tags.member.1.Value=test_value
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:3dcc0919e80b422c
Content-Type:text/xml;charset=UTF-8
Content-Length:489
Date:Thu, 31 Jan 2019 01:27:59 GMT
```

```
Server: CTYUN

<CreateUserResponse>
    <CreateUserResult>
        <User>
            <Arn>arn:ctyun:iam::10rc2arpn6306:user/test_user</Arn>
            <UserName>test_user</UserName>
            <UserId>623387e786314d3f973359c9de61a39d</UserId>
           <Tags>
                <member>
                    <Value>test_value</Value>
                    <Key>test_key</Key>
                </member>
            </Tags>
            <CreateDate>2019-01-31T01:28:00Z</CreateDate>
        </User>
    </CreateUserResult>
    <ResponseMetadata>
        <RequestId>3dcc0919e80b422c</RequestId>
    </ResponseMetadata>
</CreateUserResponse>
```

7.3.2 GetUser

此操作用来获取 IAM 用户信息。

- **请求参数**

名称	描述	是否必须
Action	GetUser。	是
Version	请求版本。 **取值**：2010-05-08。默认值为 2010-05-08。	否
UserName	查询用户的用户名。 **类型**：字符串 **取值**：1~64 个字符组成，字符只能包含字母、数字或特殊字符，字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	否

- **响应结果**

名称	描述
Arn	IAM 用户的资源名称。
UserName	IAM 用户名。
UserId	IAM 用户 ID
CreateDate	创建 IAM 用户的时间，采用 ISO 8601 日期时间格式。
PasswordLastUsed	上次使用用户密码登录 OOS 网站，采用 ISO 8601 日期时间格式。 **说明：** ● 如果从未使用密码登录，不返回此字段。 ● 如果用户当前没有密码，但过去有密码，则此字段包含最近使用密码的时间。
IPLastUsed	最近一次登录控制台的 IP
Tags.member.Key	标签键。

Tags.member.Value	标签值。

- **请求示例**

获取 IAM 用户信息。

```
POST / HTTP/1.1
Host:oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190131T020935Z
Content-Type: application/octet-stream

Action=GetUser&Version=2010-05-08
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:c8ea4d156d0f44aa
Content-Type:text/xml;charset=UTF-8
Content-Length:638
Date:Thu, 31 Jan 2019 02:09:35 GMT
Server: CTYUN

<GetUserResponse>
    <GetUserResult>
        <User>
            <PasswordLastUsed>2019-01-07T05:57:35Z</PasswordLastUsed>
            <UserName>test_user</UserName>
            <Arn>arn:ctyun:iam::10rc2arpn6306:user/test_user</Arn>
            <UserId>dd3cf79eda454f04871a87386792914f</UserId>
            <IPLastUsed>10.1.1.1</IPLastUsed>
            <Tags>
                <member>
                    <Value>test_value</Value>
                    <Key>test_tag</Key>
                </member>
            </Tags>
            <CreateDate>2019-01-07T05:53:20Z</CreateDate>
        </User>
    </GetUserResult>
    <ResponseMetadata>
```

```
        <RequestId>c8ea4d156d0f44aa</RequestId>
    </ResponseMetadata>
</GetUserResponse>
```

7.3.3 ListUsers

此操作用来列出 IAM 用户。如果账户中没有创建 IAM 用户，则返回空列表。

- **请求参数**

名称	描述	是否必须
Action	ListUsers。	是
Version	请求版本。 **取值**：2010-05-08。默认值为 2010-05-08。	否
Marker	分页标识。还有需要返回的用户时，上条响应结果中会返回该参数。查看未显示项时，请求参数中需要携带此参数。 **类型**：字符串 **取值**：与上条响应中返回的结果值相同。	否
MaxItems	设置响应中最多返回的条数。如果存在超出您指定的返回项，则 IsTruncated 响应结果为 true，表示需要再次发送请求查看未显示的项。查看未显示的项时，需要携带响应参数 Marker 的值。 **类型**：整型 **取值**：1~1000，默认值为 100。	否
UserName	查询 IAM 用户的用户名，进行模糊匹配搜索。如果请求参数中同时出现 UserName、AccessKeyId，查询的结果需要同时满足这两个条件。 **类型**：字符串 **取值**：1~64 个字符组成，字符只能包含字母、数字或特殊字符，字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	否
AccessKeyId	按指定的 AccessKeyId，进行模糊匹配搜索。	否

	UserName、AccessKeyId 共用时，查询出同时满足多个查询条件的结果。	

- **响应结果**

名称	描述
Users.member.UserName	IAM 用户名。
Users.member.UserId	IAM 用户 ID。
Users.member.Arn	IAM 用户的资源名称。
Users.member. MFADeviceCount	IAM 用户关联的 MFA 数量。
Users.member.CreateDate	创建 IAM 用户的时间，采用 ISO 8601 日期时间格式。
Users.member.PasswordLastUsed	上次使用用户密码登录 OOS 网站的时间，采用 ISO 8601 日期时间格式。 **说明：** ● 如果从未使用密码登录，不返回此字段。 ● 如果用户当前没有密码，但过去有密码，则此字段包含最近使用密码的时间。
Users.member.AccessKeyCount	用户的 AK 数量
Users.member.PasswordCreateDate	密码的创建时间。 **说明：** ● 如果更新过密码，返回密码的最近一次更新的时间。 ● 若无密码，则不返回此项。
IsTruncated	用户是否已经都返回： ● true：有未返回的用户。 ● false：已经返回所有的用户。
Marker	分页标识。当 **IsTruncated** 是 **true** 时，该项存在，并且包含要用于后续分页请求的参数 Marker 值。

- **请求示例**

  列出所有 IAM 用户。

```
POST / HTTP/1.1
Host:oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190131T025354Z
Content-Type: application/octet-stream

Action=ListUsers&Version=2010-05-08
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:826cfcba138c4af3
Content-Type:text/xml;charset=UTF-8
Content-Length:2215
Date:Thu, 31 Jan 2019 02:53:54 GMT
Server: CTYUN

<ListUsersResponse>
    <ListUsersResult>
        <IsTruncated>false</IsTruncated>
        <Users>
            <member>
                <PasswordLastUsed>2019-01-07T05:57:35Z</PasswordLastUsed>
                <UserName>test_user</UserName>
                <Arn>arn:ctyun:iam::10rc2arpn6306:user/test_user</Arn>
                <UserId>ed3cf79eda454f04871a87386792914f</UserId>
                <CreateDate>2019-01-07T05:53:20Z</CreateDate>
                < AccessKeyCount>1</AccessKeyCount>
                < MFADeviceCount>1</MFADeviceCount>
                <PasswordCreaeDate>2019-01-07T05:53:20Z</PasswordCreaeDate>
            </member>
            <member>
                <UserName>test_user_1</UserName>
                <Arn>arn:ctyun:iam::10rc2arpn6306:user/test_user_1</Arn>
                <UserId>dd3cf79eda454f04871a87386792914f</UserId>
                <CreateDate>2019-01-31T01:40:20Z</CreateDate>
                <AccessKeyCount>0</AccessKeyCount>
                <MFADeviceCount>0</MFADeviceCount>
```

```
            </member>
        </Users>
    </ListUsersResult>
    <ResponseMetadata>
        <RequestId>826cfcba138c4af3</RequestId>
    </ResponseMetadata>
</ListUsersResponse>
```

7.3.4 DeleteUser

此操作用来删除指定的 IAM 用户。

注意：被删除的 IAM 用户必须符合下列条件：没有控制台的登录密码，没有 AK/SK，没有加入用户组，没有绑定 MFA，没有绑定策略。

- **请求参数**

名称	描述	是否必须
Action	DeleteUser。	是
Version	请求版本。 **取值**：2010-05-08。默认值为 2010-05-08。	否
UserName	删除 IAM 用户的用户名。 **类型**：字符串 **取值**：1~64 个字符组成，字符只能包含字母、数字或特殊字符，字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	是

- **请求示例**

  删除用户名为 test_user 的用户。

```
POST / HTTP/1.1
Host:oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190131T015526Z
Content-Type: application/octet-stream

Action=DeleteUser&UserName=test_user&Version=2010-05-08
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:945bbd5c7da94fea
Content-Type:text/xml;charset=UTF-8
Content-Length:200
```

```
Date:Thu, 31 Jan 2019 01:55:26 GMT
Server: CTYUN

<DeleteUserResponse>
    <ResponseMetadata>
        <RequestId>945bbd5c7da94fea</RequestId>
    </ResponseMetadata>
</DeleteUserResponse>
```

7.3.5 TagUser

此操作用来为 IAM 用户添加标签。

说明：

- 可以同时添加一个或多个标签，一个 IAM 用户最多能添加 10 个标签。
- 如果 Tags.member.N.Key 已经存在，其值则会被新添加的 value 覆盖。

- **请求参数**

名称	描述	是否必须
Action	TagUser。	是
Version	请求版本。 **取值**：2010-05-08。默认值为 2010-05-08。	否
UserName	IAM 用户名。 **类型**：字符串 **取值**：1~64 个字符组成，字符只能包含字母、数字或特殊字符，字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	是
Tags.member.N.Key	用户标签键。 **类型**：字符串 **取值**：由 1~128 个字符组成，字符串符合表达式 [\p{L}\p{Z}\p{N}_.:/=+\-@]+。 **注意**： ● 一个用户最多包含 10 个标签。如果标签总数超过 10 个，则本次请求失败。	是

	● 如果新添加的标签键与原有标签重复，新添加的标签会覆盖原有标签值。	
Tags.member.N.Value	用户标签值。 **类型**：字符串 **取值**：由 0~256 个字符组成。字符串符合表达式 [\p{L}\p{Z}\p{N}_.:/=+\-@]*。	是

- **请求示例**

为 IAM 用户 test_user 添加标签，其中标签键为 test_key1，标签值为 test_value1。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190325T025056Z
Content-Type: application/octet-stream
Content-Length: 110

Action=TagUser&Version=2010-05-
08&UserName=test_user&Tags.member.1.Key=test_key1&Tags.member.1.Value=test_value1
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:97b3800f2fa4563
Content-Type:text/xml;charset=UTF-8
Content-Length:194
Date:Mon, 25 Mar 2019 02:50:56 GMT
Server: CTYUN

<TagUserResponse>
    <ResponseMetadata>
        <RequestId>97b3800f2fa4563</RequestId>
    </ResponseMetadata>
</TagUserResponse>
```

7.3.6 UntagUser

此操作用来删除用户的指定标签。

说明：如果删除的标签不存在时，返回状态 200（成功）。

- **请求参数**

名称	描述	是否必须
Action	UntagUser。	是
Version	请求版本。 **取值：**2010-05-08。默认值为 2010-05-08。	否
UserName	IAM 用户名。 **类型：**字符串 **取值：**1~64 个字符组成，字符只能包含字母、数字或特殊字符，字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	是
TagKeys.member.N	用户标签键。 **类型：**字符串 **取值：**由 1~128 个字符组成，字符串符合表达式 [\p{L}\p{Z}\p{N}_.:/=+\-@]+。	是

- **请求示例**

删除 IAM 用户 test_user 的标签键 test_key1 和 test_key2。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190325T030013Z
Content-Type: application/octet-stream
Content-Length: 79

Action=UntagUser&Version=2010-05-
08&UserName=test_user&TagKeys.member.1=test_key1&TagKeys.member.2=tag_key2
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:aa4347cbcabb4f95
Content-Type:text/xml;charset=UTF-8
Content-Length:198
Date:Mon, 25 Mar 2019 03:00:13 GMT
Server: CTYUN

<UntagUserResponse>
    <ResponseMetadata>
        <RequestId>aa4347cbcabb4f95</RequestId>
    </ResponseMetadata>
</UntagUserResponse>
```

7.3.7 ListUserTags

此操作用来列出指定 IAM 用户的标签。

- **请求参数**

名称	描述	是否必须
Action	ListUserTags	是
Version	请求版本。 **取值**：2010-05-08。默认值为 2010-05-08。	否
UserName	查询用户的用户名。 **类型**：字符串 **取值**：1~64 个字符组成，字符只能包含字母、数字或特殊字符，字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	是
Marker	分页标识。还有需要返回的标签时，上条响应结果中会返回该参数。查看未显示标签时，请求参数中需要携带此参数。 **类型**：字符串 **取值**：与上条响应中返回的结果值相同。	否
MaxItems	设置响应中最多返回的条数。如果存在超出您指定的返回项，则 IsTruncated 响应结果为 true，表示需要再次发送请求查看未显示的项。查看未显示的项时，需要携带响应参数 Marker 的值。 **类型**：整型 **取值**：1~1000，默认值为 100。	否

- **响应结果**

名称	描述

IsTruncated	标签是否已经都返回： ● **true**：有未返回的标签。 ● **false**：已经返回所有的标签。
Marker	分页标识。当 **IsTruncated** 是 **true** 时，该项存在，其值用于下一次请求的参数 **Marker** 的取值。
Tags.member.Key	标签键。
Tags.member.Value	标签值。

- **请求示例**

  列出 IAM 用户名为 test_user 的标签。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190325T023158Z
Content-Type: application/octet-stream
Content-Length: 55

Action=ListUserTags&Version=2010-05-08&UserName=test_user
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:bc3c8e298bf44f24
Content-Type:text/xml
Content-Length:404
Date:Mon, 25 Mar 2019 02:31:58 GMT
Server: CTYUN

<ListUserTagsResponse>
    <ListUserTagsResult>
        <IsTruncated>false</IsTruncated>
        <Tags>
           <member>
               <Value>test_value</Value>
               <Key>test_tag</Key>
           </member>
```

```
        </Tags>
    </ListUserTagsResult>
    <ResponseMetadata>
        <RequestId>bc3c8e298bf44f24</RequestId>
    </ResponseMetadata>
</ListUserTagsResponse>
```

7.3.8 ListGroupsForUser

此操作用来列出指定 IAM 用户所属的 IAM 用户组。

- **请求参数**

名称	描述	是否必须
Action	ListGroupsForUser。	是
Version	请求版本。 **取值**：2010-05-08。默认值为 2010-05-08。	否
UserName	IAM 用户名。 **类型**：字符串 **取值**：1~64 个字符组成，字符只能包含字母、数字或特殊字符，字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	是
Marker	分页标识。还有需要返回的用户组时，上条响应结果中会返回该参数。查看未显示项时，请求参数中需要携带此参数。 **类型**：字符串。 **取值**：与上条响应中返回的结果值相同。	否
MaxItems	设置响应中最多返回的条数。如果存在超出您指定的返回项，则 IsTruncated 响应结果为 true，表示需要再次发送请求查看未显示的项。查看未显示的项时，需要携带响应参数 Marker 的值。 **类型**：整型 **取值**：1~1000，默认值为 100。	否

- **响应结果**

名称	描述
Groups.member.GroupName	IAM 用户组名。
Groups.member.GroupId	IAM 用户组 ID。
Groups.member.Arn	IAM 用户组的资源名称。

Groups.member.CreateDate	IAM 用户组创建的时间。
IsTruncated	用户组是否已经都返回： ● **true**：有未返回的用户组。 ● **false**：已经返回所有的用户组。
Marker	分页标识。当 **IsTruncated** 是 **true** 时，该项存在，其值用于下一次请求的参数 **Marker** 的取值。

- **请求示例**

  列出 IAM 用户 test_user_2 所属的 IAM 用户组。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190214T073546Z
Content-Type: application/octet-stream
Content-Length: 64

Action=ListGroupsForUser&Version=2010-05-08&UserName=test_user_2
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:2451e997f937470a
Content-Type:text/xml;charset=UTF-8
Content-Length:616
Date:Thu, 14 Feb 2019 07:35:46 GMT
Server: CTYUN

<ListGroupsForUserResponse>
    <ListGroupsForUserResult>
        <IsTruncated>false</IsTruncated>
        <Groups>
            <member>
                <GroupName>test_group1</GroupName>
                <Arn>arn:ctyun:iam::10rc2arpn6306:group/test_group1</Arn>
                <GroupId>514648bfbc4e423f867a25281642cdfc</GroupId>
                <CreateDate>2019-01-30T06:11:39Z</CreateDate>
```

```
            </member>
        </Groups>
    </ListGroupsForUserResult>
    <ResponseMetadata>
        <RequestId>2451e997f937470a</RequestId>
    </ResponseMetadata>
</ListGroupsForUserResponse>
```

7.3.9 CreateAccessKey

此操作用来为指定的 IAM 用户创建新的 AccessKey。

说明：

- 新密钥的默认状态是 Active。
- 如果未指明 IAM 用户名，则为请求者创建新的 AccessKey。

- **请求参数**

名称	描述	是否必须
Action	CreateAccessKey。	是
Version	请求版本。 **取值**：2010-05-08。默认值为 2010-05-08。	否
UserName	IAM 用户名。 **类型**：字符串 **取值**：1~64 个字符组成，字符只能包含字母、数字或特殊字符，字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	否

- **响应结果**

名称	描述
UserName	IAM 用户名。
AccessKeyId	访问密钥的 ID。
Status	访问密钥的状态： ● **Active**：密钥启用，可以使用该密钥调用 API。 ● **Inactive**：密钥禁用，不可以使用该密钥调用 API。
SecretAccessKey	用于签名请求的密钥。

CreateDate	密钥的创建时间。

- **请求示例**

为用户名为 test_user_2 的用户创建 AccessKey。

```
POST / HTTP/1.1
Host:oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190131T091759Z
Content-Type: application/octet-stream

Action=CreateAccessKey&UserName=test_user_2&Version=2010-05-08
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:865ffbe722704365
Content-Type:text/xml;charset=UTF-8
Content-Length:603
Date:Thu, 31 Jan 2019 09:17:59 GMT
Server: CTYUN

<CreateAccessKeyResponse>
    <CreateAccessKeyResult>
        <AccessKey>
            <AccessKeyId>535d97103885f22be7a1</AccessKeyId>
            <SecretAccessKey>9e2cde278d6eeac48f50a5d276381379478c0169</SecretAccessKey>
            <UserName>test_user_2</UserName>
            <CreateDate>2019-01-31T09:18:00Z</CreateDate>
            <Status>Active</Status>
        </AccessKey>
    </CreateAccessKeyResult>
    <ResponseMetadata>
        <RequestId>865ffbe722704365</RequestId>
    </ResponseMetadata>
</CreateAccessKeyResponse>
```

7.3.10 ListAccessKeys

此操作用来返回指定 IAM 用户的 AK 的详细信息。

说明：

- 如果指定用户没有 AK，则操作返回空列表。
- 如果未指定 IAM 用户，则返回请求者的 AK。

- **请求参数**

名称	描述	是否必须
Action	ListAccessKeys。	是
Version	请求版本。 **取值**：2010-05-08。默认值为 2010-05-08。	否
UserName	IAM 用户名。 **类型**：字符串 **取值**：1~64 个字符组成，字符只能包含字母、数字或特殊字符，字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	否
Marker	分页标识。还有需要返回的密钥时，上条响应结果中会返回该参数。查看未显示项时，请求参数中需要携带此参数。 **类型**：字符串 **取值**：与上条响应中返回的结果值相同。	否
MaxItems	设置响应中最多返回的条数。如果存在超出您指定的返回项，则 IsTruncated 响应结果为 true，表示还有未返回项。查看未显示的项时，需要携带响应参数 Marker 的值。 **类型**：整型 **取值**：1~1000，默认值为 100。	否

- **响应结果**

名称	描述
UserName	用户名称。
AccessKeyMetadata.member.UserName	与访问密钥关联的 IAM 用户名。
AccessKeyMetadata.member.AccessKeyId	访问密钥的 ID。
AccessKeyMetadata.member.Status	访问密钥的状态： ● **Active**：密钥启用，可以使用该密钥调用 API。 ● **Inactive**：密钥禁用，不可以使用该密钥调用 API。
AccessKeyMetadata.member.CreateDate	创建访问密钥的日期。
IsTruncated	密钥是否已经都返回： ● **true**：有未返回的密钥。 ● **false**：已经返回所有的密钥。
Marker	分页标识。当 **IsTruncated** 是 **true** 时，该项存在，其值用于下一次请求的参数 **Marker** 的取值。
IsPrimary	显示是否为主key： ● **true**：是主key。 ● **false**：不是主key。 **说明**：仅在IAM上线前创建的密钥有此项。

- **请求示例**

列出所有的 AccessKey 的详细信息。

```
POST / HTTP/1.1
Host:oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190202T013629Z
```

```
Content-Type: application/octet-stream

Action=ListAccessKeys&Version=2010-05-08
```

- 响应示例

```
HTTP/1.1 200 OK
x-amz-request-id:524be065a8d84e68
Content-Type:text/xml;charset=UTF-8
Content-Length:559
Date:Sat, 02 Feb 2019 01:36:29 GMT
Server: CTYUN

<ListAccessKeysResponse>
    <ListAccessKeysResult>
        <UserName>test_user_2</UserName>
        <IsTruncated>false</IsTruncated>
        <AccessKeyMetadata>
            <member>
                <AccessKeyId>1e6139134373f86ec4ea</AccessKeyId>
                <UserName>test_user_2</UserName>
                <CreateDate>2019-02-02T00:47:36Z</CreateDate>
                <Status>Active</Status>
            </member>
        </AccessKeyMetadata>
    </ListAccessKeysResult>
    <ResponseMetadata>
        <RequestId>524be065a8d84e68</RequestId>
    </ResponseMetadata>
</ListAccessKeysResponse>
```

7.3.11 GetAccessKeyLastUsed

此操作用来查询指定密钥最后一次使用的时间及服务名称。

- **请求参数**

名称	描述	是否必须
Action	GetAccessKeyLastUsed。	是
Version	请求版本。 **取值：**2010-05-08。默认值为 2010-05-08。	否
AccessKeyId	访问密钥 ID。	是

- **响应结果**

名称	描述
UserName	用户名称。
AccessKeyLastUsed.LastUsedDate	最后一次使用密钥的时间。 **说明：**如果没有使用过此密钥，不显示此字段。
AccessKeyLastUsed.ServiceName	最后一次使用此密钥的服务名称。 **说明：**如果没有使用过此密钥，此字段值为 **N/A**。

- **请求示例**

最后一次使用密钥“3a7d89097c98020c8185”的时间及服务名称。

```
POST / HTTP/1.1
Host:oos-cn-iam.ctyunapi.cn
Content-Length:60
Content-MD5:gc5/jA0+njpNoeT5e3k7Fg==
Content-Type:application/x-www-form-urlencoded; charset=utf-8
x-amz-content-sha256:UNSIGNED-PAYLOAD
x-amz-date:20210707T103104Z
Authorization: SignatureValue

Action=GetAccessKeyLastUsed&AccessKeyId=3a7d89097c98020c8185
```

- 响应示例

```
HTTP/1.1 200 OK
x-amz-request-id: b567218fa9334082
Date: Wed, 07 Jul 2021 10:32:43 GMT
Content-Type: text/xml;charset=UTF-8
Content-Length: 341
Server: CTYUN

<GetAccessKeyLastUsedResponse>
    <GetAccessKeyLastUsedResult>
        <UserName>stsUser1</UserName>
        <AccessKeyLastUsed>
            <LastUsedDate>2021-06-02T07:34:45Z</LastUsedDate>
            <ServiceName>oos</ServiceName>
        </AccessKeyLastUsed>
    </GetAccessKeyLastUsedResult>
    <ResponseMetadata>
        <RequestId>b567218fa9334082</RequestId>
    </ResponseMetadata>
</GetAccessKeyLastUsedResponse>
```

7.3.12 UpdateAccessKey

此操作用来更新指定访问密钥（AK）的状态，从 Active 到 Inactive，或者从 Inactive 到 Active。

说明：

- 如果请求中未携带 IAM 用户名，则更新请求者的密钥状态。
- 此操作可以管理根用户的密钥。

- **请求参数**

名称	描述	是否必须
Action	UpdateAccessKey。	是
Version	请求版本。 **取值：**2010-05-08。默认值为 2010-05-08。	否
UserName	指定需要更新密钥状态用户的用户名。 **类型：**字符串 **取值：**1~64 个字符组成，字符只能包含字母、数字或特殊字符，字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	否
AccessKeyId	需要更新的访问密钥 ID。 **类型：**字符串	是
Status	访问密钥的状态。 **类型：**字符串 ● **Active：**密钥启用，可以使用该密钥调用 API。 ● **Inactive：**密钥禁用，不可以使用该密钥调用 API。	是

- **请求示例**

  将 AccessKeyId=535d97103885f22be7a1 的用户 AK 状态修改为 Inactive。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190202T015600Z
Content-Type: application/octet-stream

Action=UpdateAccessKey&Status=Inactive&Version=2010-05-
08&AccessKeyId=535d97103885f22be7a1
```

- 响应示例

```
HTTP/1.1 200 OK
x-amz-request-id:999e8e61dc9049bc
Content-Type:text/xml;charset=UTF-8
Content-Length:210
Date:Sat, 02 Feb 2019 01:56:03 GMT
Server: CTYUN

<UpdateAccessKeyResponse>
    <ResponseMetadata>
        <RequestId>999e8e61dc9049bc</RequestId>
    </ResponseMetadata>
</UpdateAccessKeyResponse>
```

7.3.13 DeleteAccessKey

此操作用来删除指定 IAM 用户关联的 AccessKey。

说明：

- 如果未指定用户名，IAM 将根据签名请求的 OOS AccessKeyID 确定用户名。
- 此操作也可以用来删除根用户的 AccessKey。

- **请求参数**

名称	描述	是否必须
Action	DeleteAccessKey。	是
Version	请求版本。 **取值：**2010-05-08。默认值为 2010-05-08。	否
UserName	指定删除密钥的用户名。 **类型：**字符串 **取值：**1~64 个字符组成，字符只能包含字母、数字或特殊字符，字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	否
AccessKeyId	需要删除的访问密钥 ID。 **类型：**字符串。	是

- **请求示例**

删除 AccessKeyId=535d97103885f22be7a1 对应用户的 AccessKey。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190202T011219Z
Content-Type: application/octet-stream
```

```
Action=DeleteAccessKey&Version=2010-05-08&AccessKeyId=535d97103885f22be7a1
```

- 响应示例

```
HTTP/1.1 200 OK
x-amz-request-id:66375df54d24461d
Content-Type:text/xml;charset=UTF-8
Content-Length:210
Date:Sat, 02 Feb 2019 01:12:20 GMT
Server: CTYUN

<DeleteAccessKeyResponse>
    <ResponseMetadata>
        <RequestId>66375df54d24461d</RequestId>
    </ResponseMetadata>
</DeleteAccessKeyResponse>
```

7.3.14 GetSessionToken

OOS 为用户提供临时授权访问。此操作用来获取临时访问密钥。子用户默认拥有调用此接口的权限。如果配置了禁止子用户调用该接口的 IAM 策略，该策略不会生效。

STS（Security Token Service）是为云计算用户提供临时访问令牌的 Web 服务。通过 STS，可以为第三方应用或用户颁发一个自定义时效的访问凭证。第三方应用或用户可以使用该访问凭证直接调用 OOS API，或者使用 OOS 提供的 SDK 来访问 OOS API。

使用临时授权访问 OOS API 时，用户需要将安全令牌（SessionToken）携带在请求 header 中或者预签名 URL 中。携带在请求 header 中，V4 和 V2 签名的 X-Amz-Security-Token 请求头不区分大小写。携带在预签名 URL 中，V4 签名的标头为“X-Amz-Security-Token”，V2 签名的标头为“x-amz-security-token”。

注意：使用临时访问令牌（AccessKeyId、SecretAccessKey、SessionToken）的用户不能调用该接口。使用临时访问令牌调用其他接口时，权限同生成该临时访问令牌的用户。为了安全起见，不建议根用户调用该接口。

- **请求语法**

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
Date: Date
Authorization: SignatureValue
Content-Type: application/octet-stream

Action=GetSessionToken&DurationSeconds=seconds
```

- **请求参数**

名称	描述	是否必须
Action	GetSessionToken。	是
DurationSeconds	令牌的有效期限。 类型：整型 取值：[900, 129600]，单位是秒。	是

PolicyDocument	用 JSON 语言描述的策略内容，为临时访问密钥增加权限策略限制。 携带该权限策略时：生成的临时访问密钥的权限等于调用此接口用户的权限策略和该权限策略的交集。 未携带该权限策略时：生成的临时访问密钥的权限等于调用此接口用户的权限策略。 **注意**：该参数仅对 IAM 用户生效，对根用户不生效。 PolicyDocument 的赋值需要做 urlencode 处理。 **类型**：字符串 **取值**：长度为 1~2048 的字符串。可以包含以下字符： ● 任何可打印的 ASCII 字符，范围从空格字符（\u0020）到 ASCII 字符范围的结尾。 ● Basic Latin 和 Latin-1 Supplement 字符集中的可打印字符（到\u00FF）。 ● 特殊字符 Tab（\u0009），换行符（\u000A）和回车符（\u000D）。 **说明**：各操作权限对应的具体资源、API 详见**操作权限与 API 对应关系**。	否

- **响应结果**

名称	描述
GetSessionTokenResult.Credentials.SessionToken	安全令牌。
GetSessionTokenResult.Credentials.AccessKeyId	临时访问 AK。
GetSessionTokenResult.Credentials.SecretAccessKey	临时访问 SK。
GetSessionTokenResult.Credentials.Expiration	过期时间。
ResponseMetadata.RequestId	请求ID。

- **请求示例 1**

```
POST / HTTP/1.1
Content-Length:43
Content-MD5:MQsAw5tPAXYddIR5jTmaiQ==
Content-Type:application/x-www-form-urlencoded; charset=utf-8
Host:oos-cn-iam.ctyunapi.cn
x-amz-content-sha256:UNSIGNED-PAYLOAD
x-amz-date:20210528T065932Z
Content-Type: application/octet-stream
Authorization: SignatureValue

Action=GetSessionToken&DurationSeconds=3600
```

- **响应示例 1**

```
HTTP/1.1 200 OK
x-amz-request-id: 9d9fbac87ab54310
date: Fri, 28 May 2021 06:59:33 GMT
content-type: text/xml;charset=UTF-8
content-length: 533
server: CTYUN

<GetSessionTokenResponse>
    <GetSessionTokenResult>
        <Credentials>

<SessionToken>804fcb8bbbeef97ab9326557ae61f885a3f9af49a6401f1e3d5aea3d35340eacc37ce8cd
9c370bdee8d64ba5a424a98deea1bba86507d1e2ad0653eef0d3c905</SessionToken>
            <AccessKeyId>sts.e9043b8d93c704bce974</AccessKeyId>
            <SecretAccessKey>8930a3c60dc4c9e2a3d18c221acdf4ea0d1aecc3</SecretAccessKey>
            <Expiration>2021-05-28T07:59:33.438Z</Expiration>
        </Credentials>
    </GetSessionTokenResult>
    <ResponseMetadata>
        <RequestId>9d9fbac87ab54310</RequestId>
    </ResponseMetadata>
</GetSessionTokenResponse>
```

- **请求示例 2**

  生成临时访问密钥，并对该密钥权限进行权限策略限制。权限策略为：允许 OOS 所有操作，禁止 IAM 所有操作。urlencode 前的权限策略为：`{"Version":"2012-10-`

17","Statement":[{"Effect":"Allow","Action":"oos:*","Resource":"arn:ctyun:oos::1pqvmpcd9dmxp:*"},{"Effect":"Deny","Action":"iam:*","Resource":"arn:ctyun:iam::1pqvmpcd9dmxp:*"}]}

使用时，需要使用 urlencode 工具将权限策略进行编码，编码后的权限策略为：%7B%22Version%22%3A%222012-10-17%22%2C%22Statement%22%3A%5B%7B%22Effect%22%3A%22Allow%22%2C%22Action%22%3A%22oos%3A*%22%2C%22Resource%22%3A%22arn%3Actyun%3Aoos%3A%3A1pqvmpcd9dmxp%3A*%22%7D%2C%7B%22Effect%22%3A%22Deny%22%2C%22Action%22%3A%22iam%3A*%22%2C%22Resource%22%3A%22arn%3Actyun%3Aiam%3A%3A1pqvmpcd9dmxp%3A*%22%7D%5D%7D%0A

```
POST / HTTP/1.1
Host:oos-cn-iam.ctyunapi.cn
X-Amz-Content-Sha256: 5f673b31306acbcfd7b0ba1414d27cd377f0b1d683937be4fbc773744a0573e1
X-Amz-Date: 20231218T024929Z
Authorization: SignatureValue
Content-Length:386
Content-Type: application/octet-stream

Action=GetSessionToken&DurationSeconds=900&PolicyDocument=%7B%22Version%22%3A%222012-10-17%22%2C%22Statement%22%3A%5B%7B%22Effect%22%3A%22Allow%22%2C%22Action%22%3A%22oos%3A*%22%2C%22Resource%22%3A%22arn%3Actyun%3Aoos%3A%3A1pqvmpcd9dmxp%3A*%22%7D%2C%7B%22Effect%22%3A%22Deny%22%2C%22Action%22%3A%22iam%3A*%22%2C%22Resource%22%3A%22arn%3Actyun%3Aiam%3A%3A1pqvmpcd9dmxp%3A*%22%7D%5D%7D%0A
```

- **响应示例 2**

```
HTTP/1.1 200 OK
x-amz-request-id: 3a3f41d6b9e64c03
date: Mon, 18 Dec 2023 02:49:29 GMT
content-type: text/xml;charset=UTF-8
content-length: 533
server: CTYUN

<GetSessionTokenResponse>
  <GetSessionTokenResult>
    <Credentials>
      <SessionToken>c5c6d33ca5fa55fbd7d1e42f0157b2547f37e0cdf492c0a7548a0c38084ae48040538bf9edcf54138c22a7f2e973bd5f8a0591ea8014507b0941e07995d05d3f</SessionToken>
      <AccessKeyId>sts.28b828b384a22d8aebc2</AccessKeyId>
```

```
      <SecretAccessKey>34d5a7da1e56243c494c77afd93412d5b46395af</SecretAccessKey>
      <Expiration>2023-12-18T03:04:29.856Z</Expiration>
    </Credentials>
  </GetSessionTokenResult>
  <ResponseMetadata>
    <RequestId>3a3f41d6b9e64c03</RequestId>
  </ResponseMetadata>
</GetSessionTokenResponse>
```

7.3.15 CreateLoginProfile

此操作用来为指定 IAM 用户创建控制台的登录密码。

- **请求参数**

名称	描述	是否必须
Action	CreateLoginProfile。	是
Version	请求版本。 **取值**：2010-05-08。默认值为 2010-05-08。	否
UserName	需要创建登录密码的用户名。 **类型**：字符串 **取值**：1~64 个字符组成，字符只能包含字母、数字或特殊字符，字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	是
Password	指定用户登录控制台的密码。 **类型**：字符串 **取值**：密码必须符合 OOS 账户的密码策略（如果存在）。默认密码规则为 8-128 位密码，必须是小写字母和数字的组合。	是
PasswordResetRequired	IAM 用户使用初始密码登录控制台后是否需要重置密码。 **类型**：布尔型 **取值**： - **true**：使用初始密码登录后，需要重置密码后才可以执行操作。	否

	• **false**：使用初始密码登录后，无需重置密码也可以执行操作。 默认值为 **false**。	

- **响应结果**

名称	描述
UserName	IAM 用户名。
PasswordResetRequired	IAM 用户使用初始密码登录控制台后是否需要重置密码。 • **true**：使用初始密码登录后，需要重置密码后才可以执行操作。 • **false**：使用初始密码登录后，无需重置密码也可以执行操作。
CreateDate	IAM 用户控制台登录密码创建时间。

- **请求示例**

  为 IAM 用户 test_ch_passwd 设置登录密码为 a12345678，首次登录后，再次登录时需要设置新密码才可以登录。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190315T081653Z
Content-Type: application/octet-stream
Content-Length: 112

Action=CreateLoginProfile&Version=2010-05-
08&UserName=test_ch_passwd&Password=a12345678&PasswordResetRequired=true
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:46951e81d14e4da7
Content-Type:text/xml;charset=UTF-8
```

```
Content-Length:466
Date:Fri, 15 Mar 2019 08:16:54 GMT
Server: CTYUN

<CreateLoginProfileResponse>
    <CreateLoginProfileResult>
        <LoginProfile>
            <UserName>test_ch_passwd</UserName>
            <PasswordResetRequired>true</PasswordResetRequired>
            <CreateDate>2019-03-15T08:16:55Z</CreateDate>
        </LoginProfile>
    </CreateLoginProfileResult>
    <ResponseMetadata>
        <RequestId>46951e81d14e4da7</RequestId>
    </ResponseMetadata>
</CreateLoginProfileResponse>
```

7.3.16 GetLoginProfile

此操作用来获取 IAM 用户控制台登录密码创建的时间、用户首次登录后是否需要修改密码。

- **请求参数**

名称	描述	是否必须
Action	GetLoginProfile。	是
Version	请求版本。 **取值**：2010-05-08。默认值为 2010-05-08。	否
UserName	需要查询的 IAM 用户名。 **类型**：字符串 **取值**：1~64 个字符组成，字符只能包含字母、数字或特殊字符，字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	是

- **响应结果**

名称	描述
UserName	IAM 用户名。
PasswordResetRequired	IAM 用户使用初始密码登录控制台后是否需要重置密码： ● **true**：使用初始密码登录后，需要重置密码后才可以执行操作。 ● **false**：使用初始密码登录后，无需重置密码也可以执行操作。
CreateDate	IAM 用户控制台密码创建的时间。

- **请求示例**

  获取 IAM 用户 test_user 控制台登录密码创建的时间、及用户首次登录后是否需要修改密码。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190322T032610Z
Content-Type: application/octet-stream
Content-Length: 58

Action=GetLoginProfile&Version=2010-05-08&UserName=test_user
```

- 响应示例

```
HTTP/1.1 200 OK
x-amz-request-id:9a4476f78a704d62
Content-Type:text/xml;charset=UTF-8
Content-Length:448
Date:Fri, 22 Mar 2019 03:26:11 GMT
Server: CTYUN

<GetLoginProfileResponse>
    <GetLoginProfileResult>
        <LoginProfile>
            <UserName>test_user</UserName>
            <PasswordResetRequired>false</PasswordResetRequired>
            <CreateDate>2019-01-07T05:53:22Z</CreateDate>
        </LoginProfile>
    </GetLoginProfileResult>
    <ResponseMetadata>
        <RequestId>9a4476f78a704d62</RequestId>
    </ResponseMetadata>
</GetLoginProfileResponse>
```

7.3.17 UpdateLoginProfile

此操作用来更改指定 IAM 用户控制台的登录密码。

- **请求参数**

名称	描述	是否必须
Action	UpdateLoginProfile。	是
Version	请求版本。 **取值：**2010-05-08。默认值为 2010-05-08。	否
UserName	需要更改控制台登录密码的 IAM 用户名。 **类型：**字符串 **取值：**1~64 个字符组成，字符只能包含字母、数字或特殊字符，字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	是
Password	指定用户控制台登录的新密码。 **类型：**字符串 **取值：**新密码必须符合 OOS 账户的密码策略（如果存在）。默认密码规则为 8-128 位密码，必须是小写字母和数字的组合。	否
PasswordResetRequired	IAM 用户使用初始密码登录控制台后是否需要重置密码。 **类型：**布尔型 **取值：** ● **true**：使用初始密码登录后，需要重置密码后才可以执行操作。 ● **false**：使用初始密码登录后，无需重置密码也可以执行操作。 默认值为 **false**。	否

- **请求示例**

  修改 IAM 用户 test_ch_passwd 控制台的登录密码为 a12345678。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190315T024705Z
Content-Type: application/octet-stream
Content-Length: 68

Action=UpdateLoginProfile&Version=2010-05-08&UserName=test_ch_passwd&Password=a12345678
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:b6dbab65e82e48a3
Content-Type:text/xml;charset=UTF-8
Content-Length:216
Date:Fri, 15 Mar 2019 02:47:07 GMT
Server: CTYUN

<UpdateLoginProfileResponse>
    <ResponseMetadata>
        <RequestId>b6dbab65e82e48a3</RequestId>
    </ResponseMetadata>
</UpdateLoginProfileResponse>
```

7.3.18 DeleteLoginProfile

此操作用来删除指定 IAM 用户控制台的登录密码。执行此操作后，指定用户将不能通过控制台进行 OOS 服务。

- **请求参数**

名称	描述	是否必须
Action	DeleteLoginProfile。	是
Version	请求版本。 **取值**：2010-05-08。默认值为 2010-05-08。	否
UserName	指定需要删除控制台登录密码的 IAM 用户名。 **类型**：字符串 **取值**：1~64 个字符组成，字符只能包含字母、数字或特殊字符，字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	是

- **请求示例**

  删除 IAM 用户 test_ch_passwd 控制台的登录密码。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190315T080950Z
Content-Type: application/octet-stream
Content-Length: 68

Action=DeleteLoginProfile&Version=2010-05-08&UserName=test_ch_passwd
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:576688b832fa43fb
Content-Type:text/xml;charset=UTF-8
Content-Length:216
Date:Fri, 15 Mar 2019 08:09:52 GMT
Server: CTYUN

<DeleteLoginProfileResponse>
    <ResponseMetadata>
        <RequestId>576688b832fa43fb</RequestId>
    </ResponseMetadata>
</DeleteLoginProfileResponse>
```

7.3.19 ChangePassword

此操作用来修改 IAM 用户自身的控制台登录密码。

注意：此操作仅能修改用户自身的控制台登录密码。

- **请求参数**

名称	描述	是否必须
Action	ChangePassword。	是
Version	请求版本。 **取值：**2010-05-08。默认值为 2010-05-08。	否
NewPassword	指定控制台登录新密码。 **类型：**字符串 **取值：**新密码必须符合 OOS 账户的密码策略（如果存在）。默认密码规则为 8-128 位密码，必须是小写字母和数字的组合。	是
OldPassword	目前控制台的登录密码。 **注意：**如果输入的 OldPassword 错误，则修改密码不成功，并返回错误码。	是

- **请求示例**

将用户自身的控制台登录密码由 12345678a 修改为 12345678b。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190202T015602Z
Content-Type: application/octet-stream
Content-Length: 72

Action=ChangePassword&Version=2010-05-08&OldPassword=12345678a&NewPassword=12345678b
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id: 3d19f89e195b4c77
Content-Type: text/xml
Content-Length: 210
Date:Sat, 02 Feb 2019 01:56:03 GMT
Server: CTYUN

<ChangePasswordResponse>
    <ResponseMetadata>
        <RequestId>3d19f89e195b4c77</RequestId>
    </ResponseMetadata>
</ChangePasswordResponse>
```

7.3.20 CreateVirtualMFADevice

此操作用来创建虚拟 MFA 设备。创建虚拟 MFA 后，可以使用 EnableMFADevice 启用虚拟 MFA 设备，并将该虚拟 MFA 设备与指定的 IAM 用户关联。

注意：QR 代码和 Base32 字符串中包含的 MFA 设备密钥，就像您的密码一样，应被妥善保管和处理。

- **请求参数**

名称	描述	是否必须
Action	CreateVirtualMFADevice。	是
Version	请求版本。 **取值：**2010-05-08。默认值为 2010-05-08。	否
VirtualMFADeviceName	虚拟 MFA 设备的名称。 **类型：**字符串 **取值：**1~128 个字符组成，字符只能包含字母、数字或特殊字符，字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	是

- **响应结果**

名称	描述
VirtualMFADevices.Base32StringSeed	MFA 设备密钥。
VirtualMFADevices.QRCodePNG	密钥二维码，使用 Base64 编码。
VirtualMFADevices.SerialNumber	唯一标识 MFA 设备的序列号。

- **请求示例**

创建名为 mfa1 的虚拟 MFA 设备。

```
POST / HTTP/1.1
```

```
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190315T083028Z
Content-Type: application/octet-stream
Content-Length: 74

Action=CreateVirtualMFADevice&Version=2010-05-08&VirtualMFADeviceName=mfa1
```

- 响应示例

```
HTTP/1.1 200 OK
x-amz-request-id:1a7e651e412c4ecc
Content-Type:text/xml;charset=UTF-8
Content-Length:2746
vary:accept-encoding
Date:Fri, 15 Mar 2019 08:30:30 GMT
Server: CTYUN

<CreateVirtualMFADeviceResponse>
    <CreateVirtualMFADeviceResult>
        <VirtualMFADevice>
            <Base32StringSeed>
        OBAZWZBY2WENX22KUJUUZVPKZ7OHGXBO23L52ZXHLVYIBNXSJTKAR7AZ5FH5RQLV
    </Base32StringSeed>
            <QRCodePNG><!-- byte array of png file --></QRCodePNG>
            <SerialNumber>arn:ctyun:iam::10rc2arpn6306:mfa/mfa1</SerialNumber>
        </VirtualMFADevice>
    </CreateVirtualMFADeviceResult>
    <ResponseMetadata>
        <RequestId>1a7e651e412c4ecc</RequestId>
    </ResponseMetadata>
</CreateVirtualMFADeviceResponse>
```

7.3.21 EnableMFADevice

此操作用来启用指定的虚拟 MFA 设备，并将该虚拟 MFA 设备与指定的 IAM 用户关联。

- **请求参数**

名称	描述	是否必须
Action	EnableMFADevice。	是
Version	请求版本。 **取值**：2010-05-08，默认值为 2010-05-08。	否
UserName	指定与虚拟 MFA 关联的 IAM 用户名。 **类型**：字符串 **取值**：1~64 个字符组成，字符只能包含字母、数字或特殊字符，字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	是
AuthenticationCode1	虚拟 MFA 设备发出的验证码，为六位数字验证码。	是
AuthenticationCode2	虚拟 MFA 设备发出的下一个紧邻 AuthenticationCode1 的六位数字验证码。	是
SerialNumber	唯一标识 MFA 设备的序列号。对于虚拟 MFA 设备，序列号是设备 ARN。 **类型**：字符串 **取值**：字符只能包含字母、数字或特殊字符，字母不区分大小写，特殊字符只能是：下划线（_）、右斜线（/）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）、at 符号（@）和冒号（:）。	是

- **请求示例**

  启用序列号为 arn:ctyun:iam::10rc2arpn6306:mfa/mfa1 的虚拟 MFA 设备，并将该虚拟 MFA 设备与 IAM 用户为 test_user 关联。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190322T023910Z
Content-Type: application/octet-stream
Content-Length: 175

Action=EnableMFADevice&Version=2010-05-
08&UserName=test_user&SerialNumber=arn%3Actyun%3Aiam%3A%3A10rc2arpn6306%3Amfa%2Fmfa1&A
uthenticationCode1=048078&AuthenticationCode2=856474
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:55fdaf68adb34476
Content-Type:text/xml;charset=UTF-8
Content-Length:210
Date:Fri, 22 Mar 2019 02:39:11 GMT
Server: CTYUN

<EnableMFADeviceResponse>
    <ResponseMetadata>
        <RequestId>55fdaf68adb34476</RequestId>
    </ResponseMetadata>
</EnableMFADeviceResponse>
```

7.3.22 ListVirtualMFADevices

此操作用来按分配状态列出 OOS 账户中定义的虚拟 MFA 设备。如果未指定分配状态，则操作将返回所有虚拟 MFA 设备的列表。

- **请求参数**

名称	描述	是否必须
Action	ListVirtualMFADevices。	是
Version	请求版本。 **取值**：2010-05-08。默认值为 2010-05-08。	否
AssignmentStatus	指定需要列出的虚拟 MFA 设备状态。 **类型**：字符串 **取值**： ● **Assigned**：已分配给 IAM 用户的虚拟 MFA 设备。 ● **Unassigned**：未分配给 IAM 用户的虚拟 MFA 设备。 ● **Any**：所有虚拟 MFA 设备。 默认值为 **Any**。	否
Marker	分页标识。还有需要返回的虚拟 MFA 设备时，上条响应结果中会返回该参数。查看未显示项时，请求参数中需要携带此参数。 **类型**：字符串 **取值**：与上条响应中返回的结果值相同。	否
MaxItems	设置响应中最多返回的条数。 如果存在超出您指定的返回项，则 **IsTruncated** 响应结果为 **true**，表示还有未返回项。查看未显示的项时，需要携带响应参数 **Marker** 的值。 **类型**：整型	否

	取值：1~1000，默认值为 100。	

- **响应结果**

名称	描述
IsTruncated	虚拟 MFA 设备是否已经都返回： ● **true**：有未返回的虚拟 MFA 设备。 ● **false**：已经返回所有的虚拟 MFA 设备。
Marker	分页标识。当 **IsTruncated** 是 **true** 时，该项存在，其值用于下一次请求的参数 **Marker** 的取值。
VirtualMFADevices.member.EnableDate	虚拟 MFA 设备启用的时间。
VirtualMFADevices.member.SerialNumber	唯一标识虚拟 MFA 设备的序列号。
VirtualMFADevices.member.User.PasswordLastUsed	IAM 用户密码最后使用的时间。
VirtualMFADevices.member.User.UserName	IAM 用户名。
VirtualMFADevices.member.User.UserId	IAM 用户 ID。
VirtualMFADevices.member.User.Arn	IAM 用户的资源名称。
VirtualMFADevices.member.User.CreateDate	IAM 用户被创建的时间。

- **请求示例**

列出所有已经分配的虚拟 MFA 设备。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190322T081652Z
Content-Type: application/octet-stream
Content-Length: 73

Action=ListVirtualMFADevices&Version=2010-05-08&AssignmentStatus=Assigned
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:5c6e03b9c8c043db
Content-Type:text/xml;charset=UTF-8
Content-Length:1599
Date:Fri, 22 Mar 2019 08:16:54 GMT
Server: CTYUN

<ListVirtualMFADevicesResponse>
    <ListVirtualMFADevicesResult>
        <IsTruncated>false</IsTruncated>
        <VirtualMFADevices>
            <member>
              <EnableDate>2019-03-22T08:16:09Z</EnableDate>
              <SerialNumber>arn:ctyun:iam::10rc2arpn6306:mfa/mfa1</SerialNumber>
                <User>
                    <PasswordLastUsed>2019-01-07T05:57:35Z</PasswordLastUsed>
                    <UserName>test_user</UserName>
                    <Arn>arn:ctyun:iam::10rc2arpn6306:user/test_user</Arn>
                    <UserId>dd3cf79eda454f04871a87386792914f</UserId>
                    <CreateDate>2019-01-07T05:53:20Z</CreateDate>
                </User>
            </member>
        </VirtualMFADevices>
    </ListVirtualMFADevicesResult>
    <ResponseMetadata>
        <RequestId>5c6e03b9c8c043db</RequestId>
    </ResponseMetadata>
</ListVirtualMFADevicesResponse>
```

7.3.23 ListMFADevices

此操作用来列出 IAM 用户已绑定的虚拟 MFA 设备。

说明：如果请求中没有指定用户，则会列出请求用户自己名下已绑定的虚拟 MFA 设备。

- **请求参数**

名称	描述	是否必须
Action	ListMFADevices。	是
Version	请求版本。 **取值：**2010-05-08。默认值为 2010-05-08。	否
UserName	指定 IAM 用户名。 **类型：**字符串 **取值：**1~64 个字符组成，字符只能包含字母、数字或特殊字符，字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	否
Marker	分页标识。还有需要返回的虚拟 MFA 设备时，上条响应结果中会返回该参数。查看未显示项时，请求参数中需要携带此参数。 **类型：**字符串 **取值：**与上条响应中返回的结果值相同。	否
MaxItems	设置响应中最多返回的条数。如果存在超出您指定的返回项，则 **IsTruncated** 响应结果为 **true**，表示还有未返回项。查看未显示的项时，需要携带响应参数 **Marker** 的值。 **类型：**整型 **取值：**1~1000，默认值为 100。	否

- **响应结果**

名称	描述

IsTruncated	虚拟 MFA 设备是否已经都返回： ● **true**：有未返回的虚拟 MFA 设备。 ● **false**：已经返回所有的虚拟 MFA 设备。
Marker	分页标识。当 **IsTruncated** 是 **true** 时，该项存在，其值用于下一次请求的参数 **Marker** 的取值。
MFADevices.member.EnableDate	虚拟 MFA 设备启用的时间。
MFADevices.member.SerialNumber	唯一标识虚拟 MFA 设备的序列号。
MFADevices.member.UserName	IAM 用户名。

- **请求示例**

  列出与 IAM 用户 test_ch_passwd 关联的虚拟 MFA 设备，并设置最多返回 10 条结果。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190709T055430Z
Content-Type: application/octet-stream
Content-Length: 76
Connection: Keep-Alive

Action=ListMFADevices&Version=2010-05-08&UserName=test_ch_passwd&MaxItems=10
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-requestId:6f8d2ae413ca4f1a
Content-Type:text/xml;charset=UTF-8
Content-Length:528
Date:Tue, 09 Jul 2019 05:54:33 GMT
Server: CTYUN

<ListMFADevicesResponse>
  <ListMFADevicesResult>
    <IsTruncated>false</IsTruncated>
    <MFADevices>
```

```
      <member>
        <EnableDate>2019-03-22T08:16:09Z</EnableDate>
        <SerialNumber>arn:ctyun:iam::10rc2arpn6306:mfa/mfa1</SerialNumber>
        <UserName>test_ch_passwd</UserName>
      </member>
    </MFADevices>
  </ListMFADevicesResult>
  <ResponseMetadata>
    <RequestId>6f8d2ae413ca4f1a</RequestId>
  </ResponseMetadata>
</ListMFADevicesResponse>
```

7.3.24 DeactivateMFADevice

此操作用来终止使用指定的 MFA 设备，并与用户解除关联。

- **请求参数**

名称	描述	是否必须
Action	DeactivateMFADevice。	是
Version	请求版本。 **取值**：2010-05-08，默认值为 2010-05-08。	否
UserName	指定与虚拟 MFA 解除关联的 IAM 用户名。 **类型**：字符串 **取值**：1~64 个字符组成，字符只能包含字母、数字或特殊字符，字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	是
SerialNumber	唯一标识虚拟 MFA 设备的序列号。虚拟 MFA 设备序列号是设备的 ARN。 **类型**：字符串 **取值**：字符只能包含字母、数字或特殊字符，字母不区分大小写，特殊字符只能是：下划线（_）、右斜线（/）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）、at 符号（@）和冒号（:）。	是

- **请求示例**

解绑 IAM 用户 test_user 的虚拟 MFA 设备。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
```

```
Authorization: SignatureValue
X-Amz-Date: 20190322T024559Z
Content-Type: application/octet-stream
Content-Length: 125

Action=DeactivateMFADevice&Version=2010-05-
08&UserName=test_user&SerialNumber=arn%3Actyun%3Aiam%3A%3A10rc2arpn6306%3Amfa%2Fmfa1
```

- 响应示例

```
HTTP/1.1 200 OK
x-amz-request-id:25999d8abffa4dbe
Content-Type:text/xml;charset=UTF-8
Content-Length:218
Date:Fri, 22 Mar 2019 02:45:59 GMT
Server: CTYUN

<DeactivateMFADeviceResponse>
    <ResponseMetadata>
        <RequestId>25999d8abffa4dbe</RequestId>
    </ResponseMetadata>
</DeactivateMFADeviceResponse>
```

7.3.25 DeleteVirtualMFADevice

此操作用来删除指定的虚拟 MFA 设备。

- **请求参数**

名称	描述	是否必须
Action	DeleteVirtualMFADevice。	是
Version	请求版本。 **取值**：2010-05-08，默认值为 2010-05-08。	否
SerialNumber	唯一标识虚拟 MFA 设备的序列号。虚拟 MFA 设备序列号是设备的 ARN。 **类型**：字符串 **取值**：字符只能包含字母、数字或特殊字符，字母不区分大小写，特殊字符只能是：下划线（_）、右斜线（/）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）、at 符号（@）和冒号（:）。	是

- **请求示例**

删除 SerialNumber 为 arn:ctyun:iam::10rc2arpn6306:mfa/mfa2 的 MFA 设备。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190318T030748Z
Content-type: application/octet-stream
Content-Length: 108

Action=DeleteVirtualMFADevice&Version=2010-05-
08&SerialNumber=arn%3Actyun%3Aiam%3A%3A10rc2arpn6306%3Amfa%2Fmfa2
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:76e6ad99b1294225
```

```
Content-Type:text/xml;charset=UTF-8
Content-Length:224
Date:Mon, 18 Mar 2019 03:07:49 GMT
Server: CTYUN

<DeleteVirtualMFADeviceResponse>
    <ResponseMetadata>
        <RequestId>76e6ad99b1294225</RequestId>
    </ResponseMetadata>
</DeleteVirtualMFADeviceResponse>
```

7.4 用户组管理接口

7.4.1 CreateGroup

此操作用来创建新的 IAM 用户组。

- **请求参数**

名称	描述	是否必须
Action	CreateGroup。	是
Version	请求版本。 **取值**：2010-05-08。默认值为 2010-05-08。	否
GroupName	IAM 用户组名，用户组名在账户中必须唯一。 **类型**：字符串 **取值**：1~128 个字符组成，字符只能包含字母、数字或特殊字符。字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	是

- **响应结果**

名称	描述
Arn	IAM 用户组的资源名称。
GroupName	IAM 用户组的名称。
GroupId	IAM 用户组 ID。
CreateDate	IAM 用户组创建的时间。

- **请求示例**

创建一个组名为 test_group 的 IAM 用户组。

```
POST / HTTP/1.1
Host:oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190130T060830Z
```

```
Content-type: application/octet-stream

Action=CreateGroup&Version=2010-05-08&GroupName=test_group
```

- 响应示例

```
HTTP/1.1 200 OK
x-amz-request-id:f56d2b8cbdf24617
Content-Type: text/xml;charset=UTF-8
Content-Length:510
Date:Wed, 30 Jan 2019 06:08:32 GMT
Server: CTYUN

<CreateGroupResponse>
    <CreateGroupResult>
        <Group>
            <GroupName>test_group</GroupName>
           <Arn>arn:ctyun:iam::10rc2arpn6306:group/test_group</Arn>
            <GroupId>514648bfbc4e423f867a25281642cdfc</GroupId>
            <CreateDate>2019-01-30T06:08:33Z</CreateDate>
        </Group>
    </CreateGroupResult>
    <ResponseMetadata>
        <RequestId>f56d2b8cbdf24617</RequestId>
    </ResponseMetadata>
</CreateGroupResponse>
```

7.4.2 GetGroup

此操作用来获取指定 IAM 用户组及组内 IAM 用户列表。

- **请求参数**

名称	描述	是否必须
Action	GetGroup。	是
Version	请求版本。 **取值**：2010-05-08。默认值为 2010-05-08。	否
GroupName	IAM 用户组名。 **类型**：字符串 **取值**：1~128 个字符组成，字符只能包含字母、数字或特殊字符。字母不区分大小写，特殊字符只能是：下划线_、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	是
Marker	分页标识。还有需要返回的用户时，上条响应结果中会返回该参数。查看未显示项时，请求参数中需要携带此参数。 **类型**：字符串 **取值**：与上条响应中返回的结果值相同。	否
MaxItems	设置响应中最多返回的条数。如果存在超出您指定的返回项，则 IsTruncated 响应结果为 true，表示需要再次发送请求查看未显示的项。查看未显示的项时，需要携带响应参数 Marker 的值。 **类型**：整数类型。 **取值**：1~1000，默认值为 100。	否

- **响应结果**

名称	描述
IsTruncated	用户是否已经都返回：

	● **true**：有未返回的用户。 ● **false**：已经返回所有的用户。
Marker	分页标识。当 **IsTruncated** 是 **true** 时，该项存在，其值用于下一次请求的参数 **Marker** 的取值。
Users.member.PasswordLastUsed	上次使用用户密码登录 OOS 网站时间，采用 ISO 8601 日期时间格式。 **说明：** ● 如果从未使用密码登录，不返回此字段。 ● 如果用户当前没有密码，但过去有密码，则此字段包含最近使用密码的时间。
Users.member.UserName	IAM 用户名。
Users.member.Arn	IAM 用户的资源名称。
Users.member.UserId	IAM 用户 ID。
Users.member.CreateDate	IAM 用户创建时间。
Users.member.JoinDate	IAM 用户加入该用户组的时间。
Group.GroupName	IAM 用户组名。
Group.Arn	IAM 用户组的资源名称。
Group.GroupId	IAM 用户组 ID。
Group.CreateDate	IAM 用户组创建的时间。

- **请求示例**

  获取指定 IAM 用户组 test_group 中的 IAM 用户列表。

```
POST / HTTP/1.1
Host:oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190130T065257Z
Content-type: application/octet-stream
```

```
Action=GetGroup&Version=2010-05-08&GroupName=test_group
```

- 响应示例

```
HTTP/1.1 200 OK
x-amz-request-id:eaa7820a93344169
Content-Type:text/xml;charset=UTF-8
Content-Length:1098
Date:Wed, 30 Jan 2019 06:52:57 GMT
Server: CTYUN

<GetGroupResponse>
    <GetGroupResult>
        <IsTruncated>false</IsTruncated>
        <Users>
           <member>
               <PasswordLastUsed>2019-01-07T05:57:35Z</PasswordLastUsed>
               <UserName>test1</UserName>
               <Arn>arn:ctyun:iam::10rc2arpn6306:user/test1</Arn>
               <UserId>623387e786314d3f973359c9de61a39d</UserId>
               <CreateDate>2019-01-07T05:53:20Z</CreateDate>
               <JoinDate>2019-01-10T05:53:20Z</JoinDate>
           </member>
           <member>
               <UserName>test2</UserName>
               <Arn>arn:ctyun:iam::10rc2arpn6306:user/test2</Arn>
               <UserId>723387e786314d3f973359c9de61a39d</UserId>
               <CreateDate>2015-09-21T07:20:12Z</CreateDate>
               <PasswordLastUsed>2019-07-15T01:18:21Z</PasswordLastUsed>
               <JoinDate>2019-01-10T05:53:20Z</JoinDate>
           </member>
        </Users>
        <Group>
           <GroupName>test_group</GroupName>
           <Arn>arn:ctyun:iam::10rc2arpn6306:group/test</Arn>
            <GroupId>514648bfbc4e423f867a25281642cdfc</GroupId>
           <CreateDate>2019-01-07T05:39:03Z</CreateDate>
        </Group>
    </GetGroupResult>
    <ResponseMetadata>
        <RequestId>eaa7820a93344169</RequestId>
    </ResponseMetadata>
```

```
</GetGroupResponse>
```

7.4.3 AddUserToGroup

此操作用来将指定的 IAM 用户加入到指定的 IAM 用户组，每次只能将一个用户加入到指定用户组。

说明：多次执行此操作将加同一个用户加入同一个组，不会报错。

- **请求参数**

名称	描述	是否必须
Action	AddUserToGroup。	是
Version	请求版本。 **取值：**2010-05-08。默认值为 2010-05-08。	否
UserName	IAM 用户名。 **类型：**字符串 **取值：**1~64 个字符组成，字符只能包含字母、数字或特殊字符，字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	是
GroupName	指定 IAM 用户组名。 **类型：**字符串 **取值：**1~128 个字符组成，字符只能包含字母、数字或特殊字符。字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	是

- **请求示例**

  将用户名为 test_user 的用户加入到 IAM 用户组 test_group 中。

```
POST / HTTP/1.1
Host:oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190130T075254Z
Content-type: application/octet-stream

Action=AddUserToGroup&Version=2010-05-08&UserName=test_user&GroupName=test_group
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:cf637e45e05745b3
Content-Type:text/xml;charset=UTF-8
Content-Length:208
Date:Wed, 30 Jan 2019 07:52:54 GMT
Server: CTYUN

<AddUserToGroupResponse>
    <ResponseMetadata>
        <RequestId>cf637e45e05745b3</RequestId>
    </ResponseMetadata>
</AddUserToGroupResponse>
```

7.4.4 RemoveUserFromGroup

此操作用来将指定用户从指定用户组移除。

说明：如要移除的用户存在，但不在指定的用户组，不会报错，返回 200。如果要移除的用户不存在，会报 404 NoSuchEntity 错误。

- **请求参数**

名称	描述	是否必须
Action	RemoveUserFromGroup。	是
Version	请求版本。 **取值**：2010-05-08。默认值为 2010-05-08。	否
GroupName	IAM 用户组名。 **类型**：字符串 **取值**：1~128 个字符组成，字符只能包含字母、数字或特殊字符。字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	是
UserName	要删除的 IAM 用户名。 **类型**：字符串 **取值**：1~64 个字符组成，字符只能包含字母、数字或特殊字符，字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	是

- **请求示例**

  将 IAM 用户 test_user 从 IAM 用户组 test_group 中删除。

```
POST / HTTP/1.1
Host:oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
```

```
X-Amz-Date: 20190130T080102Z
Content-type: application/octet-stream


Action=RemoveUserFromGroup&Version=2010-05-08&GroupName=test_group&UserName=test_user
```

- 响应示例

```
HTTP/1.1 200 OK
x-amz-request-id:86b7075854d24711
Content-Type:text/xml;charset=UTF-8
Content-Length:218
Date:Wed, 30 Jan 2019 08:01:03 GMT
Server: CTYUN

<RemoveUserFromGroupResponse>
    <ResponseMetadata>
        <RequestId>86b7075854d24711</RequestId>
    </ResponseMetadata>
</RemoveUserFromGroupResponse>
```

7.4.5 ListGroups

此操作用来列出所有的 IAM 用户组。

- **请求参数**

名称	描述	是否必须
Action	ListGroups。	是
Version	请求版本。 **取值**：2010-05-08。默认值为 2010-05-08。	否
GroupName	指定 IAM 用户组名。在此操作中进行用户组模糊匹配查询，例如指定 GroupName 为 test，凡是包含 test 的用户组都会返回。 **类型**：字符串 **取值**：1~128 个字符组成，字符只能包含字母、数字或特殊字符。字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	否
Marker	分页标识。还有需要返回的用户组时，上条响应结果中会返回该参数。查看未显示项时，请求参数中需要携带此参数。 **类型**：字符串 **取值**：与上条响应中返回的结果值相同。	否
MaxItems	设置响应中最多返回的条数。如果存在超出您指定的返回项，则 **IsTruncated** 响应结果为 **true**，表示还有未返回项。查看未显示的项时，需要携带响应参数 **Marker** 的值。 **类型**：整型 **取值**：1~1000，默认值为 100。	否

- **响应结果**

名称	描述
IsTruncated	用户组是否已经都返回：

	● **true**：有未返回的用户组。 ● **false**：已经返回所有的用户组。
Marker	分页标识。当 **IsTruncated** 是 **true** 时，该项存在，其值用于下一次请求的参数 **Marker** 的取值。
Groups.memeber.GroupName	IAM 用户组的名称。
Groups.memeber.Arn	IAM 用户的资源名称。
Groups.memeber.GroupID	IAM 用户组的 ID。
Groups.memeber.CreateDate	IAM 用户组创建的时间。
Groups.memeber.Policies	IAM 用户组关联的策略个数。
Groups.memeber.Users	IAM 用户组中用户的个数。

- **请求示例**

列出所有的用户组，每次最多返回 2 组。

```
POST / HTTP/1.1
Host:oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190130T072355Z
Content-type: application/octet-stream

Action=ListGroups&Version=2010-05-08&MaxItems=2
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:1d1089f15291424a
Content-Type:text/xml;charset=UTF-8
Content-Length:28798
Date:Wed, 30 Jan 2019 07:23:55 GMT
Server: CTYUN

<ListGroupsResponse>
    <ListGroupsResult>
        <IsTruncated>true</IsTruncated>
```

```
        <Groups>
            <member>
                <GroupName>PM</GroupName>
                <Arn>arn:ctyun:iam::10rc2arpn6306:group/PM</Arn>
                <GroupId>514648bfbc4e423f867a25281642cdfc</GroupId>
                <CreateDate>2012-07-30T01:01:37Z</CreateDate>
                <Policies>10</Policies>
                <Users>10</Users>
            </member>
            <member>
                <GroupName>test_group</GroupName>
                <Arn>arn:ctyun:iam::10rc2arpn6306:group/test_group</Arn>
                <GroupId>614648bfbc4e423f867a25281642cdfc</GroupId>
                <CreateDate>2019-01-07T05:39:03Z</CreateDate>
                <Policies>10</Policies>
                <Users>10</Users>
            </member>
        </Groups>
        <Marker>10rc2arpn6306|test_group</Marker>
    </ListGroupsResult>
    <ResponseMetadata>
        <RequestId>1d1089f15291424a</RequestId>
    </ResponseMetadata>
</ListGroupsResponse>
```

7.4.6 DeleteGroup

此操作用来删除指定的 IAM 用户组。

- **请求参数**

名称	描述	是否必须
Action	DeleteGroup。	是
Version	请求版本。 **取值**：2010-05-08。默认值为 2010-05-08。	否
GroupName	IAM 用户组名。 **类型**：字符串 **取值**：1~128 个字符组成，字符只能包含字母、数字或特殊字符。字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	是

- **请求示例**

  删除名为 test_group 的 IAM 用户组。

```
POST / HTTP/1.1
Host:oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190130T064232Z
Content-type: application/octet-stream

Action=DeleteGroup&Version=2010-05-08&GroupName=test_group
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id: ff7f5cc92e9e4d48
Content-Type: text/xml;charset=UTF-8
```

```
Content-Length:202
Date:Wed, 30 Jan 2019 06:42:32 GMT
Server: CTYUN

<DeleteGroupResponse>
    <ResponseMetadata>
        <RequestId>ff7f5cc92e9e4d48</RequestId>
    </ResponseMetadata>
</DeleteGroupResponse>
```

7.5 权限策略管理接口

7.5.1 CreatePolicy

此操作用来为账户创建策略。如果策略名已存在，再创建同一名称的策略，后创建的策略会将已存在的同名策略覆盖。

- **请求参数**

名称	描述	是否必须
Action	CreatePolicy。	是
Version	请求版本。 **取值**：2010-05-08。默认值为 2010-05-08。	否
PolicyName	策略名称，策略名必须唯一。 **类型**：字符串 **取值**：1~128 个字符组成，字符只能包含字母、数字或特殊字符。字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	是
Description	策略描述。 **类型**：字符串 **取值**：长度为 0~1000 的字符串。	否
PolicyDocument	用 JSON 语言描述的策略内容。 PolicyDocument 的赋值需要做 urlencode 处理。 **类型**：字符串 **取值**：长度为 1~131072 的字符串。可以包含以下字符： ● 任何可打印的 ASCII 字符，范围从空格字符（\u0020）到 ASCII 字符范围的结尾。	是

	● Basic Latin 和 Latin-1 Supplement 字符集中的可打印字符（到\ u00FF）。 ● 特殊字符 Tab（\ u0009），换行符（\ u000A）和回车符（\ u000D）。 **说明：**各操作权限对应的具体资源、API 详见**操作权限与 API 对应关系**。	

- **响应结果**

名称	描述
UpdateDate	策略最近一次更新的时间。
Description	策略描述。
PolicyId	策略 ID。
IsAttachable	策略是否可以附加到 IAM 用户或者 IAM 用户组： ● **true**：策略可以附加到 IAM 用户或 IAM 用户组。 ● **false**：策略不能附加到 IAM 用户或 IAM 用户组。
PolicyName	策略名称。
AttachmentCount	策略关联用户和用户组的数量。 如果策略附加到 1 个用户组，用户组有 N 个用户，此处仅计用户组的数量 1，不计组内的用户数。
Arn	策略的资源名称。
CreateDate	策略创建的时间。

- **请求示例**

创建名为 test_policy 的权限策略。urlencode 前的权限策略为：`{"Version":"2012-10-17","Statement":[{"Effect":"Allow","Action":"oos:*","Resource":"arn:ctyun:oos::1pqvmpcd9dmxp:*"},{"Effect":"Deny","Action":"iam:*","Resource":"arn:ctyun:iam::1pqvmpcd9dmxp:*"}]}`

使用时，需要使用 urlencode 工具将权限策略进行编码，编码后的权限策略为：%7B%22Version%22%3A%222012-10-17%22%2C%22Statement%22%3A%5B%7B%22Effect%22%3A%22Allow%22%2C%22Action%22%3A%22oos%3A*%22%2C%22Resource%22%3A%22arn%3Actyun%3Aoos%3A%3A1pqvmpcd9dmxp%3A*%22%7D%2C%7B%22Effect%22%3A%22Deny%22%2C%22Action%22%3A%22iam%3A*%22%2C%22Resource%22%3A%22arn%3Actyun%3Aiam%3A%3A1pqvmpcd9dmxp%3A*%22%7D%5D%7D%0A

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190212T063400Z
Content-type: application/octet-stream
Content-Length: 426

Action=CreatePolicy&Version=2010-05-08&PolicyName=test_policy&PolicyDocument=%7B%22Version%22%3A%222012-10-17%22%2C%22Statement%22%3A%5B%7B%22Effect%22%3A%22Allow%22%2C%22Action%22%3A%22oos%3A*%22%2C%22Resource%22%3A%22arn%3Actyun%3Aoos%3A%3A1pqvmpcd9dmxp%3A*%22%7D%2C%7B%22Effect%22%3A%22Deny%22%2C%22Action%22%3A%22iam%3A*%22%2C%22Resource%22%3A%22arn%3Actyun%3Aiam%3A%3A1pqvmpcd9dmxp%3A*%22%7D%5D%7D%0A&Description=test_des
```

- 响应示例

```
HTTP/1.1 200 OK
x-amz-request-id:a1d9f3579d9045b0
Content-Type:text/xml;charset=UTF-8
Content-Length:777
Date:Tue, 12 Feb 2019 06:34:01 GMT
Server: CTYUN

<CreatePolicyResponse>
    <CreatePolicyResult>
        <Policy>
            <UpdateDate>2019-02-12T06:34:01Z</UpdateDate>
            <Description>test_des</Description>
            <PolicyId>52cc325781954fff9069aec37c9b038a</PolicyId>
            <IsAttachable>true</IsAttachable>
            <PolicyName>test_policy</PolicyName>
            <AttachmentCount>0</AttachmentCount>
```

```
            <Arn>arn:ctyun:iam::1pqvmpcd9dmxp:policy/test_policy</Arn>
            <CreateDate>2019-02-12T06:34:01Z</CreateDate>
        </Policy>
    </CreatePolicyResult>
    <ResponseMetadata>
        <RequestId>a1d9f3579d9045b0</RequestId>
    </ResponseMetadata>
</CreatePolicyResponse>
```

7.5.2 GetPolicy

此操作用来获取策略相关信息。

- **请求参数**

名称	描述	是否必须
Action	GetPolicy。	是
Version	请求版本。 **取值：**2010-05-08。默认值为 2010-05-08。	否
PolicyArn	权限策略的资源名称。 **类型：**字符串 **取值：**符合长度为 20~2048 的字符串。	是

- **响应结果**

名称	描述
UpdateDate	策略最近一次更新的时间。
PolicyId	策略 ID。
IsAttachable	策略是否可以附加到 IAM 用户或者 IAM 用户组： ● **true：**策略可以附加到 IAM 用户或 IAM 用户组。 ● **false：**策略不能附加到 IAM 用户或 IAM 用户组。
PolicyName	策略名称。
Description	策略描述。
AttachmentCount	策略关联用户和用户组的数量。 如果策略附加到 1 个用户组，用户组有 N 个用户，此处仅计用户组的数量 1，不计组内的用户数。
Arn	策略的资源名称。
CreateDate	策略创建的时间。
Document	策略的文档内容。

Scope	策略的类型。

- **请求示例**

查询策略资源名称为 arn:ctyun:iam:: 10rc2arpn6306:policy/test_policy0 的策略相关信息。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190214T051440Z
Content-type: application/octet-stream
Content-Length: 103

Action=GetPolicy&Version=2010-05-
08&PolicyArn=arn%3Actyun%3Aiam%3A%3A10rc2arpn6306%3Apolicy%2Ftest_policy0
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:2f6014be663b4a06
Content-Type:text/xml;charset=UTF-8
Content-Length:789
Date:Thu, 14 Feb 2019 05:14:40 GMT
Server: CTYUN

<GetPolicyResponse>
    <GetPolicyResult>
        <Policy>
            <UpdateDate>2019-02-13T06:59:13Z</UpdateDate>
            <PolicyId>52cc325781954fff9069aec37c9b038a</PolicyId>
            <IsAttachable>true</IsAttachable>
            <PolicyName>test_policy0</PolicyName>
            <Description>test_des</Description>
            <AttachmentCount>2</AttachmentCount>
            <Arn>arn:ctyun:iam::10rc2arpn6306:policy/test_policy0</Arn>
            <CreateDate>2019-02-12T06:56:26Z</CreateDate>
            <Document>%7B%0A%20%20%20%20%22Version%22%3A%20%222012-10-
17%22%2C%0A%20%20%20%20%22Statement%22%3A%20%5B%0A%20%20%20%20%20%20%20%20%7B%0A%20%20
%20%20%20%20%20%20%20%20%20%20%22Sid%22%3A%20%22VisualEditor0%22%2C%0A%20%20%20%20%20%
20%20%20%20%20%20%20%20%22Effect%22%3A%20%22Allow%22%2C%0A%20%20%20%20%20%20%20%20%20%20%
20%20%22Action%22%3A%20%22oos%3AHeadBucket%22%2C%0A%20%20%20%20%20%20%20%20%20%20%20%2
```

```
0%22Resource%22%3A%20%22%2A%22%0A%20%20%20%20%20%20%20%20%7D%0A%20%20%20%20%5D%0A%7D</
Document>
        <Scope>Local</Scope>
        </Policy>
    </GetPolicyResult>
    <ResponseMetadata>
        <RequestId>2f6014be663b4a06</RequestId>
    </ResponseMetadata>
</GetPolicyResponse>
```

7.5.3 ListPolicies

此操作用来列出账户下所有的策略。

- **请求参数**

名称	描述	是否必须
Action	ListPolicies。	是
Version	请求版本。 **取值**：2010-05-08。默认值为 2010-05-08。	否
OnlyAttached	用于标识是否只显示已关联 IAM 用户或 IAM 用户组的策略。 **类型**：字符串 **取值**： ● True：只显示已关联 IAM 用户或 IAM 用户组的策略。 ● false：显示所有关联和未关联 IAM 用户或 IAM 用户组的策略。 默认值为 false。	否
PolicyName	权限策略名称。可以进行模糊匹配查询。 **类型**：字符串 **取值**：1~128 个字符组成，字符只能包含字母、数字或特殊字符。字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	否
Marker	分页标识。还有需要返回的策略时，上条响应结果中会返回该参数。查看未显示项时，请求参数中需要携带此参数。 **类型**：字符串 **取值**：与上条响应中返回的结果值相同。	否

MaxItems	设置响应中最多返回的条数。如果存在超出您指定的返回项，则 **IsTruncated** 响应结果为 **true**，表示还有未返回项。查看未显示的项时，需要携带响应参数 Marker 的值。 **类型**：整数类型 **取值**：1~1000，默认值为 100。	否

- **响应结果**

名称	描述
IsTruncated	策略是否已经都返回： ● **true**：有未返回的策略。 ● **false**：已经返回所有的策略。
Marker	分页标识。当 **IsTruncated** 是 **true** 时，该项存在，其值用于下一次请求的参数 **Marker** 的取值。
Policies.member.UpdateDate	最近一次更新策略的时间。
Policies.member.PolicyId	策略 ID。
Policies.member.IsAttachable	策略是否可以附加到 IAM 用户或者 IAM 用户组： ● **true**：策略可以附加到 IAM 用户或 IAM 用户组。 ● **false**：策略不能附加到 IAM 用户或 IAM 用户组。
Policies.member.PolicyName	策略名称。
Policies.member.AttachmentCount	关联 IAM 用户和 IAM 用户组的个数。
Policies.member.Arn	策略的资源名称。
Policies.member.CreateDate	策略创建的时间。
Policies.member.Scope	策略类型： ● **Local**：用户自定义策略。 ● **OOS**：系统策略。
Policies.member.Description	策略的描述。

- **请求示例**

列出 OOS 账户中策略类型为 Local 的策略，每次返回 1 个策略。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190214T061818Z
Content-type: application/octet-stream
Content-Length: 61

Action=ListPolicies&Version=2010-05-08&MaxItems=1
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:8e5751e20d7e4720
Content-Type:text/xml;charset=UTF-8
Content-Length:2060
Vary:Accept-Encoding
Date:Thu, 14 Feb 2019 06:18:18 GMT
Server: CTYUN

<ListPoliciesResponse>
    <ListPoliciesResult>
        <IsTruncated>true</IsTruncated>
        <Policies>
            <member>
                <UpdateDate>2019-02-12T07:03:43Z</UpdateDate>
                <PolicyId>52cc325781954fff9069aec37c9b038a</PolicyId>
                <IsAttachable>true</IsAttachable>
                <PolicyName>test_policy751</PolicyName>
                <AttachmentCount>0</AttachmentCount>
                <Arn>arn:ctyun:iam::10rc2arpn6306:policy/test_policy751</Arn>
                <CreateDate>2019-02-12T07:03:43Z</CreateDate>
                <Scope>Local</Scope>
                <Description>desc</Description>
            </member>
```

```
        </Policies>
        <Marker>10rc2arpn6306|test1_policy</Marker>
    </ListPoliciesResult>
    <ResponseMetadata>
        <RequestId>8e5751e20d7e4720</RequestId>
    </ResponseMetadata>
</ListPoliciesResponse>
```

7.5.4 ListEntitiesForPolicy

此操作用来列出指定策略所附加的所有 IAM 用户或 IAM 用户组。

- **请求参数**

名称	描述	是否必须
Action	ListEntitiesForPolicy。	是
Version	请求版本。 **取值**：2010-05-08。默认值为 2010-05-08。	否
EntityFilter	指定过滤的实体类型。如果在请求中不写此参数，则列出指定权限策略附加的所有 IAM 用户和 IAM 用户组。 类型：字符串 取值： ● **User**：仅列出指定权限策略附加的 IAM 用户。 ● **Group**：仅列出指定权限策略附加的 IAM 用户组。	否
PolicyArn	权限策略的资源名称。 **类型**：字符串 **取值**：长度为 20~2048 的字符串。	是
Marker	分页标识。还有需要返回的用户或用户组时，上条响应结果中会返回该参数。查看未显示项时，请求参数中需要携带此参数。 **类型**：字符串 **取值**：与上条响应中返回的结果值相同。	否
MaxItems	设置响应中最多返回的条数。如果存在超出您指定的返回项，则 **IsTruncated** 响应结果为 **true**，表示还有未返回项。查看未显示的项时，需要携带响应参数 **Marker** 的值。 **类型**：整型	否

	取值：1~1000，默认值为 100。	

- **响应结果**

名称	描述
IsTruncated	是否已经返回所有的用户或用户组： ● **true：**有未返回的用户或用户组。 ● **false：**已经返回所有的用户和用户组。
Marker	分页标识。当 **IsTruncated** 是 **true** 时，该项存在，其值用于下一次请求的参数 **Marker** 的取值。
PolicyUsers.member.UserName	关联该权限策略的 IAM 用户名。
PolicyUsers.member.UserID	关联该权限策略的 IAM 用户 ID。
PolicyGroups.member.GroupID	关联该权限策略的 IAM 用户组 ID。
PolicyGroups.member.GroupName	关联该权限策略的 IAM 用户组名。

- **请求示例**

  列出策略的资源名称为 arn:ctyun:iam::10rc2arpn6306/policy/test_policy0 的所有 IAM 用户和 IAM 用户组。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190214T060233Z
Content-type: application/octet-stream
Content-Length: 115

Action=ListEntitiesForPolicy&Version=2010-05-
08&PolicyArn=arn%3Actyun%3Aiam%3A%3A10rc2arpn6306%3Apolicy%2Ftest_policy0
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:c65c1442fd4f43b3
```

```
Content-Type:text/xml;charset=UTF-8
Content-Length:635
Date:Thu, 14 Feb 2019 06:02:33 GMT
Server: CTYUN

<ListEntitiesForPolicyResponse>
    <ListEntitiesForPolicyResult>
        <PolicyUsers>
            <member>
                <UserName>test_1</UserName>
                <UserId>dd3cf79eda454f04871a87386792914f</UserId>
            </member>
        </PolicyUsers>
        <PolicyGroups>
            <member>
                <GroupId>514648bfbc4e423f867a25281642cdfc</GroupId>
                <GroupName>test_group</GroupName>
            </member>
        </PolicyGroups>
        <IsTruncated>false</IsTruncated>
    </ListEntitiesForPolicyResult>
    <ResponseMetadata>
        <RequestId>c65c1442fd4f43b3</RequestId>
    </ResponseMetadata>
</ListEntitiesForPolicyResponse>
```

7.5.5 DeletePolicy

此操作用来删除指定的策略。

注意：在删除策略前，要确保该策略没有附加到任何 IAM 用户或 IAM 用户组，可以按照下列步骤进行解除关联 IAM 用户和 IAM 用户组：

1. 使用 ListEntitiesForPolicy 操作查看关联的 IAM 用户和用户组。
2. 使用 DetachUserPolicy 解除策略关联的 IAM 用户，使用 DetachGroupPolicy 解除策略关联的 IAM 用户组。

- **请求参数**

名称	描述	是否必须
Action	DeletePolicy	是
Version	请求版本。 **取值：**2010-05-08。默认值为 2010-05-08。	否
PolicyArn	权限策略的资源名称。 **类型：**字符串 **取值：**长度为 20~2048 的字符串。	是

- **请求示例**

删除权限策略资源名为 arn:ctyun:iam::10rc2arpn6306:policy/test_policy 的权限策略。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190213T062506Z
Content-type: application/octet-stream
Content-Length: 105

Action=DeletePolicy&Version=2010-05-
08&PolicyArn=arn%3Actyun%3Aiam%3A%3A10rc2arpn6306%3Apolicy%2Ftest_policy
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:4d1490ee1fcb491e
Content-Type:text/xml;charset=UTF-8
Content-Length:204
Date:Wed, 13 Feb 2019 06:25:06 GMT
Server: CTYUN

<DeletePolicyResponse>
    <ResponseMetadata>
        <RequestId>4d1490ee1fcb491e</RequestId>
    </ResponseMetadata>
</DeletePolicyResponse>
```

7.5.6 AttachUserPolicy

此操作用来将指定的策略与指定的 IAM 用户关联。

- **请求参数**

名称	描述	是否必须
Action	AttachUserPolicy。	是
Version	请求版本。 **取值**：2010-05-08。默认值为 2010-05-08。	否
PolicyArn	权限策略的资源名称。 **类型**：字符串 **取值**：长度为 20~2048 的字符串。	是
UserName	IAM 用户名。 **类型**：字符串 **取值**：1~64 个字符组成，字符只能包含字母、数字或特殊字符，字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	是

- **请求示例**

  将资源名为 arn:ctyun:iam::10rc2arpn6306:policy/test_policy0 的策略与 IAM 用户 test_user_2 关联。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190213T092112Z
Content-type: application/octet-stream
Content-Length: 131
```

```
Action=AttachUserPolicy&Version=2010-05-
08&UserName=test_user_2&PolicyArn=arn%3Actyun%3Aiam%3A%3A10rc2arpn6306%3Apolicy%2Ftest
_policy0
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:e5a4b957cdb04d42
Content-Type:text/xml;charset=UTF-8
Content-Length:212
Date:Wed, 13 Feb 2019 09:21:12 GMT
Server: CTYUN

<AttachUserPolicyResponse>
    <ResponseMetadata>
        <RequestId>e5a4b957cdb04d42</RequestId>
    </ResponseMetadata>
</AttachUserPolicyResponse>
```

7.5.7 ListAttachedUserPolicies

此操作用来列出与指定用户关联的策略。

- **请求参数**

名称	描述	是否必须
Action	ListAttachedUserPolicies。	是
Version	请求版本。 **取值**：2010-05-08。默认值为 2010-05-08。	否
UserName	IAM 用户名。 **类型**：字符串 **取值**：1~64 个字符组成，字符只能包含字母、数字或特殊字符，字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	是
Marker	分页标识。还有需要返回的策略时，上条响应结果中会返回该参数。查看未显示项时，请求参数中需要携带此参数。 **类型**：字符串 **取值**：与上条响应中返回的结果值相同。	否
MaxItems	设置响应中最多返回的条数。如果存在超出您指定的返回项，则 **IsTruncated** 响应结果为 **true**，表示还有未返回项。查看未显示的项时，需要携带响应参数 **Marker** 的值。 **类型**：整型 **取值**：1~1000，默认值为 100。	否

- **响应结果**

名称	描述
IsTruncated	与用户关联的策略是否都已经返回：

	● **true**：有未返回的与用户关联的策略。 ● **false**：已经返回所有与用户关联的策略。
Marker	分页标识。当 **IsTruncated** 是 **true** 时，该项存在，其值用于下一次请求的参数 **Marker** 的取值。
AttachedPolicies.member.PolicyArn	策略的资源名称。
AttachedPolicies.member.PolicyName	策略名称。
AttachedPolicies.member.Scope	策略类型： ● **Local**：用户自定义策略。 ● **OOS**：系统策略。
AttachedPolicies.member.Description	策略描述。

- **请求示例**

  列出 IAM 用户 test_user_2 关联的策略。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190214T055307Z
Content-type: application/octet-stream
Content-Length: 82

Action=ListAttachedUserPolicies&Version=2010-05-08&UserName=test_user_2&MaxItems=2
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:a0597755ea82423f
Content-Type:text/xml;charset=UTF-8
Content-Length:824
Date:Thu, 14 Feb 2019 05:53:07 GMT
Server: CTYUN

<ListAttachedUserPoliciesResponse>
    <ListAttachedUserPoliciesResult>
```

```
        <IsTruncated>false</IsTruncated>
        <AttachedPolicies>
            <member>
                <PolicyArn>arn:ctyun:iam::10rc2arpn6306:policy/policy9</PolicyArn>
                <PolicyName>policy9</PolicyName>
                <Scope>Local</Scope>
                <Description>test</Description>
            </member>
            <member>
                <PolicyArn>arn:ctyun:iam::10rc2arpn6306:policy/policy8</PolicyArn>
                <PolicyName>policy8</PolicyName>
                <Scope>Local</Scope>
                <Description>test</Description>
            </member>
        </AttachedPolicies>
    </ListAttachedUserPoliciesResult>
    <ResponseMetadata>
        <RequestId>a0597755ea82423f </RequestId>
    </ResponseMetadata>
</ListAttachedUserPoliciesResponse>
```

7.5.8 DetachUserPolicy

此操作用来解除指定用户关联的指定策略。

- **请求参数**

名称	描述	是否必须
Action	DetachUserPolicy。	是
Version	请求版本。 **取值**：2010-05-08。默认值为 2010-05-08。	否
UserName	IAM 用户名。 **类型**：字符串 **取值**：1~64 个字符组成，字符只能包含字母、数字或特殊字符，字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	是
PolicyArn	权限策略的资源名称。 **类型**：字符串 **取值**：长度为 20~2048 的字符串。	是

- **请求示例**

解除 IAM 用户 test_user_2 关联的策略资源名为 arn:ctyun:iam::10rc2arpn6306:policy/test_policy0 的策略。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190214T023845Z
Content-type: application/octet-stream
Content-Length: 131
```

```
Action=DetachUserPolicy&Version=2010-05-
08&UserName=test_user_2&PolicyArn=arn%3Actyun%3Aiam%3A%3A10rc2arpn6306%3Apolicy%2Ftest_policy0
```

- 响应示例

```
HTTP/1.1 200 OK
x-amz-request-id:143dc6f82dae45bd
Content-Type:text/xml;charset=UTF-8
Content-Length:212
Date:Thu, 14 Feb 2019 02:38:45 GMT
Server: CTYUN

<DetachUserPolicyResponse>
    <ResponseMetadata>
        <RequestId>143dc6f82dae45bd</RequestId>
    </ResponseMetadata>
</DetachUserPolicyResponse>
```

7.5.9 AttachGroupPolicy

此操作用来将指定的策略与指定的 IAM 用户组关联。

- **请求参数**

名称	描述	是否必须
Action	AttachGroupPolicy。	是
Version	请求版本。 **取值**：2010-05-08。默认值为 2010-05-08。	否
PolicyArn	权限策略的资源名称。 **类型**：字符串 **取值**：长度为 20~2048 的字符串。	是
GroupName	IAM 用户组名。 **类型**：字符串 **取值**：1~128 个字符组成，字符只能包含字母、数字或特殊字符，字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	是

- **请求示例**

  将资源名为 arn:ctyun:iam::10rc2arpn6306:policy/test_policy0 的策略与 IAM 用户组 test_group1 关联。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190213T070613Z
Content-type: application/octet-stream
Content-Length: 133
```

```
Action=AttachGroupPolicy&Version=2010-05-
08&GroupName=test_group1&PolicyArn=arn%3Actyun%3Aiam%3A%3A10rc2arpn6306%3Apolicy%2Ftes
t_policy0
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:ebe8e92cf0424b3a
Content-Type:text/xml;charset=UTF-8
Content-Length:214
Date:Wed, 13 Feb 2019 07:06:14 GMT
Server: CTYUN

<AttachGroupPolicyResponse>
    <ResponseMetadata>
        <RequestId>ebe8e92cf0424b3a</RequestId>
    </ResponseMetadata>
</AttachGroupPolicyResponse>
```

7.5.10 ListAttachedGroupPolicies

此操作用来列出与指定 IAM 用户组关联的策略。

- **请求参数**

名称	描述	是否必须
Action	ListAttachedGroupPolicies。	是
Version	请求版本。 **取值**：2010-05-08。默认值为 2010-05-08。	否
GroupName	IAM 用户组名。 **类型**：字符串 **取值**：1~128 个字符组成，字符只能包含字母、数字或特殊字符，字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	是
Marker	分页标识。还有需要返回的策略时，上条响应结果中会返回该参数。查看未显示项时，请求参数中需要携带此参数。 **类型**：字符串 **取值**：与上条响应中返回的结果值相同。	否
MaxItems	设置响应中最多返回的条数。如果存在超出您指定的返回项，则 **IsTruncated** 响应结果为 **true**，表示还有未返回项。查看未显示的项时，需要携带响应参数 **Marker** 的值。 **类型**：整型 **取值**：1~1000，默认值为 100。	否

- **响应结果**

名称	描述
IsTruncated	与用户组关联的策略是否已经都返回：

	● **true**：有未返回的与用户组关联的策略。 ● **false**：已经返回所有的与用户组关联的策略。
Marker	分页标识。当 **IsTruncated** 是 **true** 时，该项存在，其值用于下一次请求的参数 **Marker** 的取值。
AttachedPolicies.member.PolicyArn	策略的资源名称。
AttachedPolicies.member.PolicyName	策略名称。
AttachedPolicies.member.Scope	策略类型： ● **Local**：用户自定义策略。 ● **OOS**：系统策略。
AttachedPolicies.member.Description	策略描述。

- **请求示例**

  列出 IAM 用户组 test_group1 关联的策略, 响应中所需的最大项目数为 2。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190214T053115Z
Content-type: application/octet-stream
Content-Length: 84

Action=ListAttachedGroupPolicies&Version=2010-05-08&GroupName=test_group1&MaxItems=2
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:8590bb26ed75455f
Content-Type:text/xml;charset=UTF-8
Content-Length:828
Date:Thu, 14 Feb 2019 05:31:15 GMT
Server: CTYUN

<ListAttachedGroupPoliciesResponse>
    <ListAttachedGroupPoliciesResult>
```

```
        <IsTruncated>true</IsTruncated>
        <AttachedPolicies>
            <member>
                <PolicyArn>arn:ctyun:iam::10rc2arpn6306:policy/policy9</PolicyArn>
                <PolicyName>policy9</PolicyName>
                <Scope>Local</Scope>
                <Description>test_des</Description>
            </member>
            <member>
                <PolicyArn>arn:ctyun:iam::10rc2arpn6306:policy/policy8</PolicyArn>
                <PolicyName>policy8</PolicyName>
                <Scope>Local</Scope>
                <Description>test_des</Description>
            </member>
        </AttachedPolicies>
        <Marker>policy|10rc2arpn6306|test_group|Local|test1_policy</Marker>
    </ListAttachedGroupPoliciesResult>
    <ResponseMetadata>
        <RequestId>8590bb26ed75455f</RequestId>
    </ResponseMetadata>
</ListAttachedGroupPoliciesResponse>
```

7.5.11 DetachGroupPolicy

此操作用来解除指定 IAM 用户组关联的指定策略。

- **请求参数**

名称	描述	是否必须
Action	DetachGroupPolicy。	是
Version	请求版本。 **取值**：2010-05-08。默认值为 2010-05-08。	否
GroupName	IAM 用户组名。 **类型**：字符串 **取值**：1~64 个字符组成，字符只能包含字母、数字或特殊字符，字母不区分大小写，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。	是
PolicyArn	权限策略的资源名称。 **类型**：字符串 **取值**：长度为 20~2048 的字符串。	是

- **请求示例**

  解除 IAM 用户组 test_group1 关联的策略资源名为 arn:ctyun:iam::10rc2arpn6306:policy/test_policy0 的策略。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190214T021410Z
Content-type: application/octet-stream
Content-Length: 133
```

```
Action=DetachGroupPolicy&Version=2010-05-
08&GroupName=test_group1&PolicyArn=arn%3Actyun%3Aiam%3A%3A10rc2arpn6306%3Apolicy%2Ftes
t_policy0
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:22893776fa50431b
Content-Type:text/xml;charset=UTF-8
Content-Length:214
Date:Thu, 14 Feb 2019 02:14:10 GMT
Server: CTYUN

<DetachGroupPolicyResponse>
    <ResponseMetadata>
        <RequestId>22893776fa50431b</RequestId>
    </ResponseMetadata>
</DetachGroupPolicyResponse>
```

7.5.12 UpdateAccountPasswordPolicy

此操作用来更新账户的密码规则设置。

- **请求参数**

名称	描述	是否必须
Action	UpdateAccountPasswordPolicy。	是
Version	请求版本。 **取值**：2010-05-08。默认值为 2010-05-08。	否
AllowUsersToChangePassword	是否允许 IAM 用户自己更改控制台的密码。 **类型**：布尔型 **取值**： ● **true**：允许用户自己更改控制台的密码。 ● **false**：不允许用户自己更改控制台的密码。 默认值为 **true**。	否
HardExpiry	控制台密码过期后，下次登录时，是否允许用户在控制台修密码。 **类型**：布尔型 **取值**： ● **true**：控制台密码过期后，用户不能通过控制台登录，显示密码过期，用户不能通过控制台修改密码。 ● **false**：控制台密码过期后，用户下次通过控制台时，直接跳转到修改密码界面。 默认值为 **false**。	否
MaxPasswordAge	IAM 用户密码有效天数。 **类型**：整型。	否

	取值：0~1095,0 表示永不过期，默认值为 0。	
MinimumPasswordLength	控制台登录密码最短的长度。 **类型**：整型 **取值:**8~128。默认值为 8。	否
PasswordReusePrevention	指定 IAM 用户设置新登录密码时，不能与前多少次内的登录密码重复。 **类型**：整型 **取值**：0~24，0 表示允许 IAM 用户设置先前的登录密码为新登录密码，默认取值为 0。先前的密码不包含当前使用的密码，新密码不能设置为当前的密码。	否
RequireLowercaseCharacters	指定控制台登录密码中是否必须包含小写字母（a-z）。 **类型**：布尔型 **取值**： ● **true**：必须包含小写字母。 ● **false**：不强制要求包含小写字母。 默认值为 **true**。	否
RequireNumbers	指定控制台登录密码中是否必须包含数字（0-9）。 **类型**：布尔型 **取值**： ● **true**：必须包含数字。 ● **false**：不强制要求包含数字。 默认值为 **true**。	否

RequireSymbols	指定控制台登录密码中是否必须包含特殊字符：！@ # $ % ^ & * ( ) _ + - = [ ] { } | ' **类型：**布尔型 **取值：** ● **true：**必须包含特殊字符。 ● **false：**不强制要求包含特殊字符。 默认值为 **false**。	否
RequireUppercaseCharacters	指定控制台登录密码中是否必须包含大写字母（A-Z）。 **类型：**布尔型 **取值：** ● **true：**必须包含大写字母。 ● **false：**不强制要求包含大写字母。 默认值为 **false**。	否

- **请求示例**

更新账户的密码规则设置为：

- 必须包含小写字母。
- 允许用户自己修改控制台的登录密码。
- 密码的有效期为 10 天。
- 控制台密码过期后，允许用户可以自己修改登录密码。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190322T031434Z
Content-type: application/octet-stream
Content-Length: 152
```

```
Action=UpdateAccountPasswordPolicy&Version=2010-05-
08&RequireLowercaseCharacters=true&AllowUsersToChangePassword=true&MaxPasswordAge=10&H
ardExpiry=false
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:81435d5089d341d3
Content-Type:text/xml;charset=UTF-8
Content-Length:234
Date:Fri, 22 Mar 2019 03:14:35 GMT
Server: CTYUN

<UpdateAccountPasswordPolicyResponse>
    <ResponseMetadata>
        <RequestId>81435d5089d341d3</RequestId>
    </ResponseMetadata>
</UpdateAccountPasswordPolicyResponse>
```

7.5.13 GetAccountPasswordPolicy

此操作用来获取账户的密码策略。

- **请求参数**

名称	描述	是否必须
Action	GetAccountPasswordPolicy。	是
Version	请求版本。 **取值：**2010-05-08。默认值为 2010-05-08。	否

- **响应结果**

名称	描述
ExpirePasswords	密码是否有过期时间： ● **true：**有过期时间。 ● **false：**无过期时间，即永不过期。
MinimumPasswordLength	控制台登录密码最短的长度。
PasswordReusePrevention	IAM 用户设置新登录密码时，不能与前多少次内的登录密码重复。
RequireSymbols	指定控制台登录密码中是否必须包含特殊字符（! @ # $ % ^ & * ( ) _ + - = [ ] { } | '）： ● **true：**必须包含特殊字符。 ● **false：**不强制要求包含特殊字符。
AllowUsersToChangePassword	是否允许 IAM 用户自己更改控制台的密码： ● **true：**允许用户自己更改控制台的密码。 ● **false：**不允许用户自己更改控制台的密码。
RequireLowercaseCharacters	指定控制台登录密码中是否必须包含小写字母（a-z）：

	● **true**：必须包含小写字母。 ● **false**：不强制要求包含小写字母。
HardExpiry	控制台密码过期后，下次登录时，是否允许用户在控制台修密码： ● **true**：控制台密码过期后，用户不能通过控制台登录，显示密码过期，用户不能通过控制台修改密码。 ● **false**：控制台密码过期后，用户下次通过控制台时，直接跳转到修改密码界面。
RequireNumbers	指定控制台登录密码中是否必须包含数字（0-9）： ● **true**：必须包含数字。 ● **false**：不强制要求包含数字。
RequireUppercaseCharacters	指定控制台登录密码中是否必须包含大写字母（A-Z）： ● **true**：必须包含大写字母。 ● **false**：不强制要求包含大写字母。
MaxPasswordAge	密码的有效天数，0 表示永不过期。

- **请求示例**

  列出账户的密码策略。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190322T031900Z
Content-type: application/octet-stream
Content-Length: 50

Action=GetAccountPasswordPolicy&Version=2010-05-08
```

- 响应示例

```
HTTP/1.1 200 OK
x-amz-request-id:be5351535a7c4304
Content-Type:text/xml;charset=UTF-8
Content-Length:817
Date:Fri, 22 Mar 2019 03:19:00 GMT
Server: CTYUN

<GetAccountPasswordPolicyResponse>
    <GetAccountPasswordPolicyResult>
        <PasswordPolicy>
            <ExpirePasswords>true</ExpirePasswords>
            <MinimumPasswordLength>8</MinimumPasswordLength>
            <RequireSymbols>false</RequireSymbols>
           <PasswordReusePrevention>0</PasswordReusePrevention>
            <AllowUsersToChangePassword>true</AllowUsersToChangePassword>
            <RequireLowercaseCharacters>true</RequireLowercaseCharacters>
            <HardExpiry>false</HardExpiry>
            <RequireNumbers>false</RequireNumbers>
            <RequireUppercaseCharacters>false</RequireUppercaseCharacters>
            <MaxPasswordAge>10</MaxPasswordAge>
        </PasswordPolicy>
    </GetAccountPasswordPolicyResult>
    <ResponseMetadata>
        <RequestId>be5351535a7c4304</RequestId>
    </ResponseMetadata>
</GetAccountPasswordPolicyResponse>
```

7.5.14 DeleteAccountPasswordPolicy

此操作用来将账户的密码规则恢复到默认密码规则。

- **请求参数**

名称	描述	是否必须
Action	DeleteAccountPasswordPolicy。	是
Version	请求版本。 **取值：**2010-05-08。默认值为 2010-05-08。	否

- **请求示例**

将账户的密码规则恢复至默认的密码规则：

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190318T025813Z
Content-type: application/octet-stream
Content-Length: 53

Action=DeleteAccountPasswordPolicy&Version=2010-05-08
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id:c9ac3274a12b425a
Content-Type:text/xml;charset=UTF-8
Content-Length:234
Date:Mon, 18 Mar 2019 02:58:15 GMT
Server: CTYUN

<DeleteAccountPasswordPolicyResponse>
    <ResponseMetadata>
        <RequestId>c9ac3274a12b425a</RequestId>
    </ResponseMetadata>
</DeleteAccountPasswordPolicyResponse>
```

7.5.15 UpdateAccountLoginSecurityPolicy

此操作用来更新 IAM 用户登录安全策略设置。

- **请求参数**

名称	描述	是否必须
Action	UpdateAccountLoginSecurityPolicy。	是
Version	请求版本。 **取值**：2010-05-08。默认值为 2010-05-08。	否
PeriodWithLoginFailures	登录失败次数的限定时间。 如果在限定登录时长内 IAM 用户达到登录失败次数后，会被锁定一段时间，锁定时间结束后，才能重新登录。 **类型**：整型 **取值**：整数形式，[15, 60]，单位是分钟。默认值为 15。	否
LoginFailedTimes	IAM 用户在限定时间内允许连续登录失败的次数。 **类型**：整型 **取值**：整数形式，[5, 10]。默认值为 5。	否
LockoutDuration	IAM 用户被锁定的时间。 **类型**：整型 **取值**：整数形式，[15, 60]，单位是分钟。默认值为 15。	否
AllowSingleUsersSimultaneousLogin	是否允许 IAM 用户同一时刻在多个客户端登录。 **类型**：布尔型	否

	取值： ● **true**：允许 IAM 用户同一时刻在多个客户端登录。 ● **false**：不允许 IAM 用户同一时刻在多个客户端登录。IAM 用户同一时刻在多个客户端登录时，最后一次的登录会保持，之前的登录将被强制下线。 默认值为 **true**。	
SessionDuration	IAM 用户登录控制台后，在无任何操作时，保存会话的时长。 **类型**：整型 **取值**：整数形式，[10, 30]，单位是分钟。 默认值为 30。	否

- **请求示例**

  更新 IAM 用户登录安全策略设置。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
Connection: Keep-Alive
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20211214T024257Z
Content-Type: application/octet-stream
Content-Length: 124

Action=UpdateAccountLoginSecurityPolicy&Version=2010-05-
08&PeriodWithLoginFailures=20&AllowSingleUsersSimultaneousLogin=true
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id: 9f1f4d6e6d1d4ab8
Date: Tue, 14 Dec 2021 02:42:57 GMT
```

```
Content-Type: text/xml;charset=UTF-8
Content-Length: 161
Server: CTYUN


<UpdateAccountLoginSecurityPolicyResponse>
    <ResponseMetadata>
        <RequestId>9f1f4d6e6d1d4ab8</RequestId>
    </ResponseMetadata>
</UpdateAccountLoginSecurityPolicyResponse>
```

7.5.16 GetAccountLoginSecurityPolicy

此操作用来获取 IAM 用户登录安全策略。

- **请求参数**

名称	描述	是否必须
Action	GetAccountLoginSecurityPolicy。	是
Version	请求版本。 **取值：**2010-05-08。默认值为 2010-05-08。	否

- **响应结果**

名称	描述
LoginSecurityPolicy.PeriodWithLoginFailures	登录失败次数的限定时间。单位为分钟。
LoginSecurityPolicy.LoginFailedTimes	IAM 用户在限定时间内允许连续登录失败的次数。
LoginSecurityPolicy.LockoutDuration	IAM 用户被锁定的时间。单位为分钟。
LoginSecurityPolicy.AllowSingleUsersSimultaneousLogin	是否允许 IAM 用户同一时刻在多个客户端登录： ● **true**：允许 IAM 用户同一时刻在多个客户端登录。 ● **false**：不允许 IAM 用户同一时刻在多个客户端登录。IAM 用户同一时刻在多个客户端登录时，最后一次的登录会保持，之前的登录将被强制下线。

LoginSecurityPolicy.SessionDuration	IAM 用户登录控制台后，在无任何操作时，保存会话的时长。单位为分钟。

- **请求示例**

  获取 IAM 用户的登录安全策略。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
Connection: Keep-Alive
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20211214T023208Z
Content-Type: application/octet-stream
Content-Length: 55

Action=GetAccountLoginSecurityPolicy&Version=2010-05-08
```

- **响应示例**

```
HTTP/1.1 200 OK

x-amz-request-id: 3d49e20c3e544ec3
Date: Tue, 14 Dec 2021 02:32:08 GMT
Content-Type: text/xml;charset=UTF-8
Content-Length: 513
Server: CTYUN

<GetAccountLoginSecurityPolicyResponse>
    <GetAccountLoginSecurityPolicyResult>
        <LoginSecurityPolicy>
            <PeriodWithLoginFailures>15</PeriodWithLoginFailures>
            <LoginFailedTimes>5</LoginFailedTimes>
            <LockoutDuration>15</LockoutDuration>
            <AllowSingleUsersSimultaneousLogin>true</AllowSingleUsersSimultaneousLogin>
            <SessionDuration>30</SessionDuration>
        </LoginSecurityPolicy>
    </GetAccountLoginSecurityPolicyResult>
    <ResponseMetadata>
        <RequestId>3d49e20c3e544ec3</RequestId>
```

```
    </ResponseMetadata>
</GetAccountLoginSecurityPolicyResponse>
```

7.5.17 DeleteAccountLoginSecurityPolicy

此操作用来将 IAM 用户登录安全策略恢复为默认值。默认值为：

- PeriodWithLoginFailures：15 分钟。
- LoginFailedTimes：5 次。
- LockoutDuration：15 分钟。
- AllowSingleUsersSimultaneousLogin：true。
- SessionDuration：30 分钟

- **请求参数**

名称	描述	是否必须
Action	DeleteAccountLoginSecurityPolicy。	是
Version	请求版本。 **取值**：2010-05-08。默认值为 2010-05-08。	否

- **请求示例**

将 IAM 用户登录安全策略恢复为默认值：

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
Connection: Keep-Alive
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20211214T024410Z
Content-Type: application/octet-stream
Content-Length: 58

Action=DeleteAccountLoginSecurityPolicy&Version=2010-05-08
```

- **响应示例**

```
HTTP/1.1 200 OK
x-amz-request-id: 8f0b0ec76d984417
Date: Tue, 14 Dec 2021 02:44:12 GMT
Content-Type: text/xml;charset=UTF-8
Content-Length: 161
```

```
Server: CTYUN


<DeleteAccountLoginSecurityPolicyResponse>
    <ResponseMetadata>
        <RequestId>8f0b0ec76d984417</RequestId>
    </ResponseMetadata>
</DeleteAccountLoginSecurityPolicyResponse>
```

7.6 服务数量查询

7.6.1 GetAccountSummary

此操作用来获取账户中的实体数量和服务限制信息。

- **请求参数**

名称	描述	是否必须
Action	GetAccountSummary。	是
Version	请求版本。 **取值：**2010-05-08。默认值为 2010-05-08。	否

- **响应结果**

名称	描述
Policies	账户中已有的策略个数。
GroupsPerUserQuota	IAM 用户加入到 IAM 用户组的数量限制。
AttachedPoliciesPerUserQuota	每个 IAM 用户关联的策略数量限制。
Users	账户中已有的 IAM 用户数。
PoliciesQuota	账户中的策略数量限制。
AccessKeysPerUserQuota	每个 IAM 用户的 AccessKey 数量限制。
AttachedPoliciesPerGroupQuota	每个 IAM 用户组关联的策略数量限制。
Groups	账户中已有的用户组数量。
UsersQuota	账户中的 IAM 用户数量限制。
MFADevices	账户中已有的虚拟 MFA 设备总数。
MFADevicesInUse	账户中已使用的虚拟 MFA 设备数量。
AccountAccessKeysPresent	根用户已有的 AccessKey 数量。
GroupsQuota	账户中的 IAM 用户组数量限制。

- **请求示例**

  列出账户中的实体数量和服务限制。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: SignatureValue
X-Amz-Date: 20190415T025603Z
Content-type: application/octet-stream
Content-Length: 43

Action=GetAccountSummary&Version=2010-05-08
```

- 响应示例

```
HTTP/1.1 200 OK
x-amz-request-id:1d1089f15291424a
Content-Type:text/xml;charset=UTF-8
Content-Length:3297
vary:accept-encoding
Date:Mon, 15 Apr 2019 02:56:03 GMT
Server: CTYUN

<GetAccountSummaryResponse>
    <GetAccountSummaryResult>
        <SummaryMap>
            <entry><key>Policies</key><value>150</value></entry>
            <entry><key>GroupsPerUserQuota</key><value>10</value></entry>
            <entry><key>AttachedPoliciesPerUserQuota</key><value>10</value></entry>
            <entry><key>Users</key><value>392</value></entry>
            <entry><key>PoliciesQuota</key><value>150</value></entry>
            <entry><key>AccessKeysPerUserQuota</key><value>2</value></entry>
            <entry><key>AttachedPoliciesPerGroupQuota</key><value>10</value></entry>
            <entry><key>Groups</key><value>29</value></entry>
            <entry><key>UsersQuota</key><value>500</value></entry>
            <entry><key>MFADevices</key><value>8</value></entry>
            <entry><key>MFADevicesInUse</key><value>1</value></entry>
            <entry><key>AccountAccessKeysPresent</key><value>1</value></entry>
            <entry><key>GroupsQuota</key><value>30</value></entry>
        </SummaryMap>
    </GetAccountSummaryResult>
    <ResponseMetadata>
        <RequestId>1d1089f15291424a</RequestId>
    </ResponseMetadata>
</GetAccountSummaryResponse>
```

7.7 错误码列表

响应码	错误码	错误信息	描述
400	AuthorizationHeaderMalformed	The authorization header is malformed.	签名请求头不合法。
400	AuthorizationQueryParametersError	X-Amz-Expires must be less than a week (in seconds); that is, the given X-Amz-Expires must be less than 604800 seconds	签名请求参数不合法。
400	MalformedInput	Invalid Argument.	参数不正确。
400	MalformedInput	*N* is not a valid index.	标签的 N 非法。
400	MalformedPolicyDocument	The policy must contain a valid version string	创建策略时，版本不正确。
400	MalformedPolicyDocument	Missing required field Effect	创建策略时，策略没有 Effect。
400	MalformedPolicyDocument	Invalid effect: *effect*	创建策略时，Effect 不正确。
400	MalformedPolicyDocument	Missing required field Action.	创建策略时，Action 缺失。
400	MalformedPolicyDocument	Required field Action cannot be empty.	创建策略时，有 Action，但是 Action 是空.
400	MalformedPolicyDocument	Statement/policy already has instance of Action	创建策略时，Action 和 NotAction 都有时，报错。
400	MalformedPolicyDocument	Missing required field Resource.	创建策略时，Resource 字段缺失。
400	MalformedPolicyDocument	Required field Resource cannot be e mpty.	创建策略时，有 Resouce，但是 Resource 是空。
400	MalformedPolicyDocument	Statement/policy already has instance of Resource.	创建策略时，Resouce 和 NotResource 都有
400	MalformedPolicyDocument	Invalid Condition type : *condition*	创建策略时，条件运算符或者条件键不正确。
400	MalformedPolicyDocument	Syntax errors in policy	创建策略时，policy 不是 json 格式。

400	InvalidAction	Could not find operation *APIName* for version *version*.	Version 错误。
400	InvalidArgument	InvalidDurationSeconds.	DurationSecond 无效。
400	InvalidInput	Arn *policyArn* is not valid.	策略的 Arn 不合法。
400	InvalidInput	Duplicate tag keys found. Please note that Tag keys are case insensitive.	设置了重复的标签 key。
400	PasswordPolicyViolation	Password does not conform to the account password policy.	更改密码时，密码不符合密码策略。
400	PasswordPolicyViolation	Policy constraint violation with password reuse prevention during password change.	该密码和历史密码重复。
400	ValidationError	1 validation errors detected: Value '*groupName*' at 'groupName' failed to satisfy constraint: Member must have length less than or equal to 128	用户组名称长度超过限制。
400	ValidationError	1 validation error detected: Value 'InActive' at 'status' failed to satisfy constraint: Member must satisfy enum value set: [Active, Inactive]	AccessKey 的状态设置的不对。
400	ValidationError	1 validation errors detected: Value null at 'groupName' failed to satisfy constraint: Member must not be null	用户组名为空。
400	ValidationError	1 validation error detected: Value '*userName*' at 'userName' failed to satisfy constraint: Member must have length less than or equal to 64	userName 长度超过限制。
400	ValidationError	The specified value for groupName is invalid. It must contain only alphanumeric characters and/or the following: +=,.@_-	创建用户组时，名称不正确，包含非法字符。
400	ValidationError	Value at 'oldPassword' failed to satisfy constraint: Member must satisfy regular expression pattern: [\u0009\u000A\u000D\u0020-\u00FF]+	更改密码时，密码是空字符串。

400	ValidationError	The specified value for authenticationCode1/ authenticationCode2 is invalid. It must be a six-digit decimal number	设置的 MFA code 格式不正确。
400	ValidationError	2 validation errors detected: Value '1' at 'minimumPasswordLength' failed to satisfy constraint: Member must have value greater than or equal to 6; Value '2000' at 'maxPasswordAge' failed to satisfy constraint: Member must have value less than or equal to 1095	设置密码策略时，策略选项的值不合法。
400	ValidationError	1 validation error detected: Value null at 'tags' failed to satisfy constraint: Member must not be null	添加标签时，没有设置标签。
400	ValidationError	1 validation error detected: Value null at 'tags.1.member.key' failed to satisfy constraint: Member must not be null	添加标签时，没有设置标签的 Key。
400	ValidationError	1 validation error detected: Value null at 'tags.1.member.value' failed to satisfy constraint: Member must not be null	添加标签时，没有设置标签的 value。
400	ValidationError	4 validation errors detected: Value null at 'tags.1.member.value' failed to satisfy constraint: Member must not be null; Value null at 'tags.1.member.key' failed to satisfy constraint: Member must not be null; Value null at 'tags.2.member.value' failed to satisfy constraint: Member must not be null; Value null at 'tags.2.member.key' failed to satisfy constraint: Member must not be null	设置标签时，如果 N 未从 1 开始（例如从 3 开始），会提示 1 到 N-1 的标签没有设置。
400	ValidationError	2 validation errors detected: Value 'test_value?' at 'tags.1.member.value' failed to satisfy constraint: Member	创建用户时，标签名称或值非法。

		must satisfy regular expression pattern: [\p{L}\p{Z}\p{N}_.:/=+\-@]*; Value 'test_key?' at 'tags.1.member.key' failed to satisfy constraint: Member must satisfy regular expression pattern: [\p{L}\p{Z}\p{N}_.:/=+\-@]+	
400	ValidationError	1 validation error detected: Value '*key*' at 'tags.1.member.key' failed to satisfy constraint: Member must have length less than or equal to 128	标签名称长度超长。
400	ValidationError	1 validation error detected: Value '*value*' at 'tags.1.member.value' failed to satisfy constraint: Member must have length less than or equal to 256	标签值长度超长。
400	ValidationError	The specified value for 'serialNumber' is invalid.	MFA 设备的序列号不正确。
400	ValidationError	1 validation error detected: Value '*value*' at 'assignmentStatus' failed to satisfy constraint: Member must satisfy enum value set: [Unassigned, Any, Assigned]	ListVirtualMFADevices Result 时，assignmentStatus 输入不对。
400	ValidationError	The specified value for 'marker' is invalid. It must contain only printable ASCII characters	list user 时，marker 参数非法，不是可打印的 ASCII 码。
400	ValidationError	The specified value for 'userName' is invalid. It must contain only alphanumeric characters and/or the following: +=,.@_-	创建用户时，名称不正确，使用了非法字符。
400	ValidationError	1 validation error detected: Value null at 'policyDocument' failed to satisfy constraint: Member must not be null	创建策略时，policyDocument 为空。
400	ValidationError	1 validation error detected: Value '*entityFilter*' at 'entityFilter' failed to satisfy constraint: Member must	ListEntitiesForPolicy 时，EntityFilter 不正

		satisfy enum value set: [User, Group]	确，不是 User 或 Group.
400	ValidationError	The specified value for 'policyName' is invalid. It must contain only alphanumeric characters and/or the following: +=,.@_-	名称不正确，包含非法字符。
400	ValidationError	1 validation error（s） detected: Value '1' at ' PeriodWithLoginFailures ' failed to satisfy constraint: Member must have value greater than or equal to 15.	限定时间长度不能小于 15。
400	ValidationError	1 validation error detected: Value '*value*' at ' PeriodWithLoginFailures ' failed to satisfy constraint f: Member must have value less than or equal to 60。	限定时间长度不能大于 60。
400	ValidationError	1 validation error detected: Value '*value*' at ' LoginFailedTimes ' failed to satisfy constraint: Member must have value greater than or equal to 5。	登录失败次数不能小于 5。
400	ValidationError	1 validation error detected: Value '*value*' at ' LoginFailedTimes ' failed to satisfy constraint f: Member must have value less than or equal to 10。	登录失败次数不能大于 10。
400	ValidationError	1 validation error detected: Value '*value*' at ' LockoutDuration ' failed to satisfy constraint f: Member must have value less than or equal to 60。	锁定时间长度不能大于 60。
400	ValidationError	1 validation error detected: Value '*value*' at ' SessionDuration ' failed to satisfy constraint: Member must have value greater than or equal to 10。	登录 Session 时间长度不能小于 10。

400	ValidationError	1 validation error detected: Value '*value*' at ' SessionDuration ' failed to satisfy constraint f: Member must have value less than or equal to 30。	登录 Session 时间长度不能大于 30。
403	AccessDenied	The old password was incorrect.	更改密码时，旧密码不正确。
403	AccessDenied	The secret token is expired。	临时访问凭证已过期。
403	AccessDenied	Only IAM Users can change their own password.	root 用户调用 ChangePassword 方法。
403	AccessDenied	Policy is outside your own account.	arn 中的账户 id 和不是请求者的账户 id。
403	AccessDenied	User: *user* is not authorized to perform: *action* on resource: *resource*.	没有权限访问此资源。
403	InvalidAccessKeyId	The AccessKeyId is invalid.	密钥无效。
403	InvalidAuthenticationCode	Authentication code for device is not valid.	MFA 身份认证码不正确。
403	SecurityTokenNotSupported	This interface does not support security token.	当前接口暂不支持通过临时访问凭证权限进行调用。
404	NoSuchEntity	Login Profile for User *userName* cannot be found.	用户无密码。
404	NoSuchEntity	MFA Device invalid for user.	DeactivateMFA 设备时，MFA 设备未绑定到用户。
404	NoSuchEntity	Login Profile for User *userName* cannot be found.	调用 UpdateLoginProfile 时，被修改用户没有密码。
404	NoSuchEntity	Policy arn:ctyun:iam::*policy* does not exist or is not attachable.	策略不存在。
404	NoSuchEntity	VirtualMFADevice with serial number *mfaArn* does not exist.	MFA 设备不存在。
404	NoSuchEntity	The Access Key with id *id* cannot be found.	AccessKey 不存在。
404	NoSuchEntity	The group with name *groupName* cannot be found.	用户组不存在。

404	NoSuchEntity	The user with name *userName* cannot be found.	用户不存在。
404	NoSuchEntity	The Password Policy with domain name *accountId* cannot be found.	删除密码策略时，没有用户自己创建的密码策略。
409	DeleteConflict	Cannot delete entity, must remove users from group first.	删除用户组之前，要先删除用户组中的用户。
409	DeleteConflict	Cannot delete entity, must detach all policies first.	删除用户组之前，要先删除用户组上的策略。
409	DeleteConflict	Cannot delete entity, must remove users from group first.	用户属于某个用户组时，不允许删除。
409	DeleteConflict	Cannot delete entity, must delete login profile first.	用户有登录密码时，不允许删除。
409	DeleteConflict	Cannot delete entity, must detach all policies first.	用户分配了策略时，不允许删除。
409	DeleteConflict	Cannot delete entity, must delete access keys first.	用户有 AccessKey/SecretAccessKey 时，不允许删除。
409	DeleteConflict	Cannot delete a policy attached to entities.	policy 有被附加到用户或用户组，不能删除。
409	DeleteConflict	MFA VirtualDevice in use. Must deactivate first.	MFA 设备正在被使用，不能删除。
409	DeleteConflict	Cannot delete entity, must deactivate MFA device first.	DeleteUser 时，必须先解绑 MFA 设备。
409	EntityAlreadyExists	MFA Device entity at the same path and name already exists.	MFA 设备已经存在。
409	EntityAlreadyExists	MFA Device is already in use.	EnableMFADevice 时，MFA 设备已被启用。
409	EntityAlreadyExists	Group with name *groupName* already exists.	用户组已经存在。
409	EntityAlreadyExists	Login Profile for user *userName* already exists.	用户的密码已经存在。
409	EntityAlreadyExists	User with name *userName* already exists.	试图创建的用户名已经存在。
409	LimitExceeded	Cannot exceed quota for AccessKeysPerAccount: 2.	根用户数量密钥超过限制。

409	LimitExceeded	Cannot exceed quota for AccessKeysPerUser: 2 .	AK 数量超过限制
409	LimitExceeded	Cannot exceed quota for GroupsPerAccount: 30.	用户组的数量超过了最大限制。
409	LimitExceeded	Cannot exceed quota for PoliciesPerAccount: 150.	策略数量超过限制。
409	LimitExceeded	Cannot exceed quota for PoliciesPerGroup: 10.	用户组关联的策略数量超过限制。
409	LimitExceeded	Cannot exceed quota for UsersPerAccount: 500.	用户数量超过限制。
409	LimitExceeded	Cannot exceed quota for PoliciesPerGroup: 10.	用户组关联的策略数量超过限制。
409	LimitExceeded	Cannot exceed quota for PoliciesPerUser: 10.	用户关联的策略数量超过限制。
409	LimitExceeded	Cannot exceed quota limit for MFADevicesPerUser.	为用户分配的 MFA 超过上限。
409	LimitExceeded	The number of tags has reached the maximum limit.	添加标签时，标签数量超过限制。
500	InternalError	We encountered an internal error. Please try again.	发生内部错误。

7.8 IAM 策略编写规则

7.8.1 Version

Version 策略元素用在策略之中，用于定义策略语言的版本，包含在所有策略中的 Statement 元素之前。

目前 OOS IAM 在用的策略版本为：2012-10-17，兼容 AWS 最新策略版本。

如果未包含 Version 元素，则此值默认为 2012-10-17。

7.8.2 Statement

Statement 为策略的主要元素，该元素为必填项。Statement 中可含一条单独的 JSON 语句，也可包含由多条语句组成的 JSON 语句块。每条单独的语句块必须使用大括号{}括起来。每个 JSON 语句块中包括下列元素：Sid（非必填）、Effect（必填）、Action 或 NotAction（二选一）、Resource 或 NotResource（二选一）、Condition（非必填）。

Statement 语句的结构如下：

```
“Statement”: [ {…}, {…}, {…}, …]
```

例如下例为多个 JSON 语句块组成的示例：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": " AllowGroupToManageTrail",
            "Effect": "Allow",
            "Action": "cloudtrail:*",
            "Resource": "*"
        },
        {
            "Sid": " AllowGroupToSeeBucket",
            "Effect": "Allow",
            "Action": [
                "oos:GetObject",
                "oos:ListBucket"
            ],
            "Resource": [
                "arn:ctyun:oos::10rc2arpn6306:trailbucket",
```

```
            "arn:ctyun:oos::10rc2arpn6306:trailbucket/*"
        ]
      }
   ]
}
```

7.8.2.1 Sid

Sid 是针对策略语句提供的可选标识符，用户可以为声明数组中的每份声明指定 Sid 值，Sid 值是策略文件 ID 的子 ID。在 IAM 中，Sid 值在 JSON 策略中必须唯一。

7.8.2.2 Effect

Effect 元素是必需具备的元素，用于指定声明所产生的结果是“允许”还是“显式拒绝”。Effect 的有效值为 Allow 和 Deny。在默认情况下，将拒绝访问资源。如要允许访问资源，必须将 Effect 元素设置为 Allow。

7.8.2.3 Action

Action 元素描述将允许或拒绝的指定操作。每个服务有对应的任务操作，用户可以使用相应服务来执行所描述的任务。目前提供的服务有：oos（对象存储）、cloudtrail（操作跟踪）、statistics（统计）和 iam。具体每种服务包括的操作详见**操作列表**。

Action 元素的语法结构为：”Action”: “服务:具体操作”。其中具体操作也可以用通配符（*）表示某类操作。

- 示例 1：OOS：获取文件操作。

```
"Action": "oos:GetObject"
```

- 示例 2：IAM：创建 IAM 用户。

```
"Action": "iam:CreateUser"
```

- 示例 3：使用通配符（*）表示执行 OOS 的所有服务。

```
"Action": "oos:*"
```

- 示例 4：使用通配符（*）表示执行 IAM 服务中包含 AccessKey 的操作。

```
"Action": "iam:*AccessKey*"
```

7.8.2.4 NotAction

NotAction 元素描述与指定操作列表之外的所有内容显式匹配。使用 NotAction 时只列出不应匹配的一些操作。使用 NotAction 时：

- 如果使用 Allow 效果，则允许未列出的所有适用操作或服务。
- 如果使用 Deny 效果，则拒绝此类未列出的操作或服务。如果想允许某个已列出的操作，则必须显式允许此操作。

- 示例 1：除删除存储桶操作外，允许用户执行 OOS 其他所有操作。

```
{
    "Version": "2012-10-17",
    "Statement": [{
         "Effect": "Allow",
         "NotAction": "oos:DeleteBucket",
         "Resource": "arn:ctyun:oos::10rc2arpn6306:*",
      }]
  }
```

- 示例 2：允许用户执行除 IAM 服务外的所有操作。

```
{
    "Version": "2012-10-17",
    "Statement": [{
       "Effect": "Allow",
       "NotAction": "iam:*",
       "Resource": "*",
    }]
  }
```

- 示例 3：拒绝除 oos、cloudtrail 和 statistics 之外的服务。但并不是允许 oos、cloudtrail 和 statistics 服务的操作，如果允许 oos、cloudtrail 和 statistics 中的某个操作，需要再写新的策略进行显式允许。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Deny",
        "NotAction": [
            "oos:*",
            "cloudtrail:*",
            "statistics:*",
        ],
        "Resource": "*",
    }]
}
```

7.8.2.5 Resource

Resource 元素指定执行策略的资源，可以指定一个或多个文件。

格式可以为：

- “Resource”: “arn:ctyun:*service*::*accountid*:*resource*”
- “Resource”: “arn:ctyun:*service*::*accountid*:*resourcetype*/*resource*”

其中：

- *service*：服务名。
- *accountid*：账户 ID。
- *resource*：具体资源。在指定资源时，可以使用通配符，其中*表示字符的任意组合，？表示任何单个字符。

说明：

- 在 resource 最后部分添加**策略变量“${ctyun:username}”**指定占位符。当策略执行时，策略变量将被替换为请求本身的用户名。
- 在 resource 最后部分添加**策略变量“${ctyun:AccessKey}”**指定占位符。当策略执行时，策略变量将被替换为请求本身的 AccessKeyID。

如下列举例，将含有策略变量的策略附加给多个用户，当用户 A 发起请求时，username 将替换为 A 的用户名。当用户 B 发起请求时，username 将替换为 B 的用户名。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "oos:GetObject",
        "oos:PutObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:ctyun:oos::123456789012:mybucket/${ctyun:username}/*"]
    }
  ]
}
```

- *resourcetype*：资源类型。

7.8.2.6 NotResource

NotResource 元素指除指定资源列表之外的所有内容显式匹配的策略元素。使用 NotResource 时，只列出不应匹配的一些资源，而不是包括将匹配的资源列表。使用 NotResource 时应注意，在此元素中指定的资源是受限的资源，即：

- 如果使用 Allow，则将允许未列出的所有资源，包括所有其他服务中的资源。
- 如果使用 Deny，则拒绝所有未列出资源。

7.8.2.7 Condition

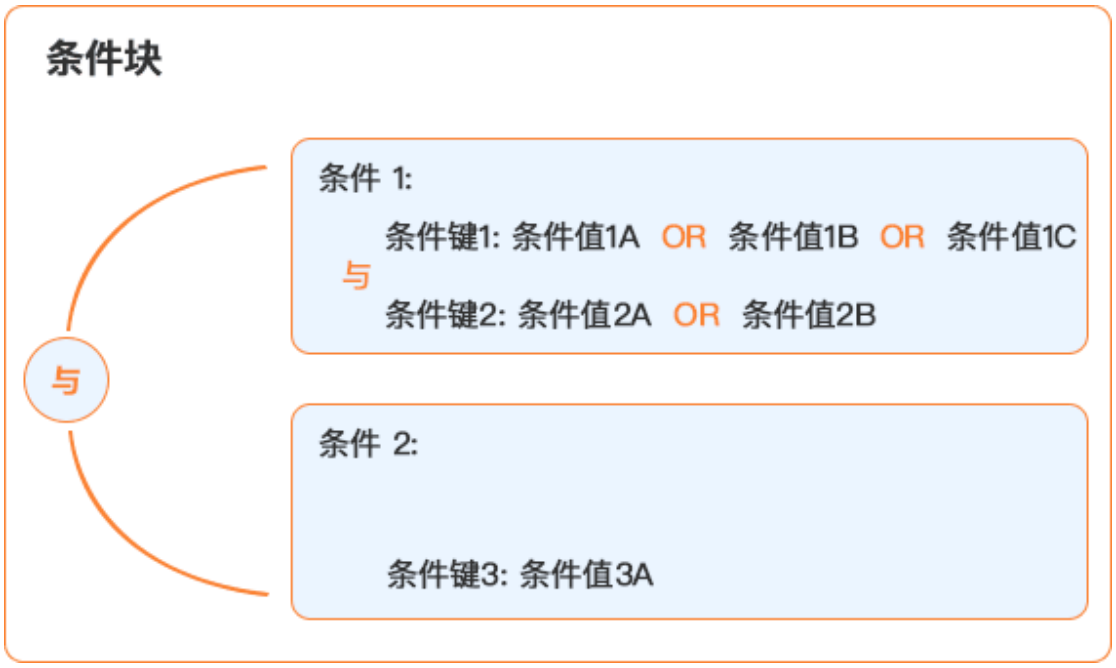
Condition 元素描述允许用户指定策略生效的条件。在 Condition 元素中，用户可构建表达式，并使用条件运算符将策略中的条件与请求值相匹配。

Condition 元素可以由多个条件组成。条件包括：条件运算符、条件键和条件值组成，一个条件键可以对应多个条件值。

Condition 的语法结构如下：

```
“Condition”: {“条件运算符 A”: {“条件键 A”:[“条件值 A1”, “条件值 A2”，…]}, “条件运算符 B”: {“条件键 B”:[“条件值 B1”, “条件值 B2”，…] } }
```

说明：条件键不区分大小写。如果条件值是时间，将需要设置的时间转换为 UTC+0 时区的时间。



若存在多个条件，各个条件之间的约束如下：

- 存在多个条件运算符，采用逻辑 AND 评估这些条件。
- 若一个条件键对应多个条件值，采用逻辑 OR 评估这些条件值。
- 必须满足所有条件运算符才能做出允许或者拒绝。如果多个条件中的任何一个不满足，那么策略不生效。

条件键、运算符、条件值见下表：

条件键	运算符	条件值
ctyun:CurrentTime	● **DateEquals**：匹配指定日期。 ● **DateNotEquals**：不等于指定日期。 ● **DateLessThan**：早于指定日期。 ● **DateLessThanEquals**：早于或等于指定日期。 ● **DateGreaterThan**：晚于指定日期。 ● **DateGreaterThanEquals**：晚于或等于指定日期。	格式为：yyyy-MM-dd'T'HH:mm:ss'Z'。例如：2019-12-18T09:00:00Z。 **DateEquals** 和 **DateNotEquals** 精确到天，其他精确到秒。 **注意**：将需要设置的时间转换为 UTC+0 时间。
ctyun:SourceIp	● **IpAddress**：与指定 IP 地址或范围匹配。 ● **NotIpAddress**：除指定 IP 地址或范围外的所有 IP 地址匹配。	● IPv4：点分十进制格式 ● IPv6：32 位 16 进制数，格式为 X:X:X:X:X:X:X:X。 如果指定地址范围，IP 地址后加掩码表示，如 192.163.1.5/3。
ctyun:userid	● **StringEquals**：精准匹配指定的值，区分大小写。 ● **StringNotEquals**：与指定的值不匹配，区分大小写。 ● **StringEqualsIgnoreCase**：与指定的值精准匹配，不区分大小写。	包含数字和小写字母的 32 个字符。 运算符为 **StringLike** 和 **StringNotLike**，可以包含通配符。

	● **StringNotEqualsIgnoreCase**：与指定的值不匹配，不区分大小写。 ● **StringLike**：与指定的值精准匹配。或通过填充通配符，与指定的值相似，可以包括多字符匹配的通配符(*) 或单字符匹配的通配符 (?)。区分大小写。 ● **StringNotLike**：与指定的值不匹配，区分大小写的无效匹配。或通过填充通配符，与指定的值也不匹配。	
ctyun:username	● **StringEquals**：精准匹配指定的值，区分大小写。 ● **StringNotEquals**：与指定的值不匹配，区分大小写。 ● **StringEqualsIgnoreCase**：与指定的值精准匹配，不区分大小写。 ● **StringNotEqualsIgnoreCase**：与指定的值不匹配，不区分大小写。 ● **StringLike**：与指定的值精准匹配。或通过填充通配符，与指定的值相似，可以包括多字符匹配的通配符(*)或单字符匹配的通配符 (?)。区分大小写。 ● **StringNotLike**：与指定的值不匹配，区分大小写的无效匹配。或通过填充通配符，与指定的值也不匹配。	1~64 个字符组成，字符只能包含字母、数字或特殊字符，特殊字符只能是：下划线（_）、中划线（-）、逗号（,）、句点（.）、加号（+）、等号（=）和 at 符号（@）。 **说明：**运算符为 **StringLike** 和 **StringNotLike**，可以包含通配符。
ctyun:UserAgent	● **StringEquals**：精准匹配指定的值，区分大小写。	字符串，可以包含特殊字符。

	• **StringNotEquals**：与指定的值不匹配，区分大小写。 • **StringEqualsIgnoreCase**：与指定的值精准匹配，不区分大小写。 • **StringNotEqualsIgnoreCase**：与指定的值不匹配，不区分大小写。 • **StringLike**：与指定的值精准匹配。或通过填充通配符，与指定的值相似，可以包括多字符匹配的通配符 (*) 或单字符匹配的通配符 (?)。区分大小写。 • **StringNotLike**：与指定的值不匹配，区分大小写的无效匹配。或通过填充通配符，与指定的值也不匹配。	
ctyun:Referer	• **StringEquals**：精准匹配指定的值，区分大小写。 • **StringNotEquals**：与指定的值不匹配，区分大小写。 • **StringEqualsIgnoreCase**：与指定的值精准匹配，不区分大小写。 • **StringNotEqualsIgnoreCase**：与指定的值不匹配，不区分大小写。 • **StringLike**：与指定的值精准匹配。或通过填充通配符，与指定的值相似，可以包括多字符匹配的通配符 (*)或单字符匹配的通配符 (?)。区分大小写。 • **StringNotLike**：与指定的值不匹配，	字符串，可以包含特殊字符。

	区分大小写的无效匹配。或通过填充通配符，与指定的值也不匹配。	
ctyun:SecureTransport	**Bool**：布尔匹配。	● **true** ● **false**
ctyun:MultiFactorAuthPresent	**Bool**：布尔匹配。 **说明**：只有 IAM 服务支持此条件键。	● **true** ● **false**
ctyun:MultiFactorAuthAge	● **NumericEquals**：与指定的值相同。 ● **NumericNotEquals**：与指定的值不同，否定匹配。 ● **NumericLessThan**：小于指定的值。 ● **NumericLessThanEquals**：小于等于指定的值。 ● **NumericGreaterThan**：大于指定的值。 ● **NumericGreaterThanEquals**：大于等于指定的值。 **说明**：只有 IAM 服务支持此条件键。	整型。
oos:prefix	● **StringEquals**：精准匹配指定的值，区分大小写。 ● **StringNotEquals**：与指定的值不匹配，区分大小写。 ● **StringEqualsIgnoreCase**：与指定的值精准匹配，不区分大小写。 ● **StringNotEqualsIgnoreCase**：与指定的值不匹配，不区分大小写。 ● **StringLike**：与指定的值精准匹配。或通过填充通配符，与指定的值相似，可以包括多字符匹配的通配符	字符串形式。 说明：本条件键仅对 ListBucket 生效。

	(*) 或单字符匹配的通配符 (?)。区分大小写。 ● **StringNotLike**：与指定的值不匹配，区分大小写。值可以在字符串中的任何位置包括多字符匹配的通配符 (*) 或单字符匹配的通配符 (?)。	
oos:x-amz-acl	● **StringEquals**：精准匹配指定的值，区分大小写。 ● **StringNotEquals**：与指定的值不匹配，区分大小写。 ● **StringEqualsIgnoreCase**：与指定的值精准匹配，不区分大小写。 ● **StringNotEqualsIgnoreCase**：与指定的值不匹配，不区分大小写。 ● **StringLike**：与指定的值精准匹配。或通过填充通配符，与指定的值相似，可以包括多字符匹配的通配符 (*) 或单字符匹配的通配符 (?)。区分大小写。 ● **StringNotLike**：与指定的值不匹配，区分大小写。值可以在字符串中的任何位置包括多字符匹配的通配符 (*) 或单字符匹配的通配符 (?)。	字符串形式。 取值为： ● private：私有 ● public-read：公共读 ● public-read-write：公共读写 说明：创建 Bucket 时，通过使用此条件键可以控制存储桶 ACL 的类型，本条件键仅对 PutBucket 生效。

说明：

■ 在 Condition 元素中添加**策略变量“${ctyun:username}”**指定占位符。当策略执行时，策略变量将被替换为请求本身的用户名。

- 在 Condition 最后部分添加**策略变量“${ctyun:AccessKey}”**指定占位符。当策略执行时，策略变量将被替换为请求本身的 AccessKeyID。

示例：将含有策略变量的策略附加给多个用户，当用户 A 发起请求时，条件键 oos:prefix 将根据用户 A 的 username 进行判断。当用户 B 发起请求时，条件键 oos:prefix 将根据用户 B 的 username 进行判断。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": ["oos:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:ctyun:oos::123456789012:mybucket"],
      "Condition": {"StringLike": {"oos:prefix": ["${ctyun:username}/*"]}}
    }
  ]
}
```

- **…IfExists 条件运算符**

IfExists：如果请求的内容中存在关键字，则依照策略所述的条件来处理关键字。如果该关键字不存在，则条件元素的计算结果将为 true。

目前仅 Bool 型和数字类型的运算符支持使用 IfExists 条件运算符，表达形式：*运算符* IfExists，例如 BoolIfExists、NumericEqualsIfExists。对于…IfExists 的使用见示例 1 和示例 2。

示例 1

- 拒绝没有使用 MFA 认证的控制台请求，不拒绝使用 MFA 认证的控制台请求和使用密钥的 API 请求。但如果允许使用 MFA 认证的控制台请求和使用密钥的 API 请求，需要再写显性允许语句。

```
"Effect" : "Deny",
"Condition" : { "Bool" : { "ctyun:MultiFactorAuthPresent" : false } }
```

- 拒绝没有使用 MFA 认证的控制台请求及使用密钥的 API 请求，不拒绝 MFA 认证的控制台请求。但如果允许 MFA 认证的控制台请求，需要再写显性允许语句。

```
"Effect" : "Deny",
"Condition" : { "BoolIfExists" : { "ctyun:MultiFactorAuthPresent" : false } }
```

示例 2

- 允许使用 MFA 认证在 1800 秒内的请求及使用密钥的 API 请求。

```
"Effect" : "Allow",
"Condition" : { " NumericLessThanEqualsIfExists" : { "ctyun:MultiFactorAuthAge " :
1800 } }
```

- 允许使用 MFA 认证在 1800 秒内的请求，但不允许 MFA 认证在 1800 秒以上及没有使用 MFA 的请求（包括 API 请求）。

```
"Effect" : "Allow",
"Condition" : { " NumericLessThanEquals" : { "ctyun:MultiFactorAuthAge " : 1800 } }
```

7.8.2.8 策略变量

在编写策略时，如果不能确定 Resource、NotResource 或 Condition 元素中的精确值，可以使用策略变量作为占位符。目前仅支持变量“**${ctyun:username}**”、“**${ctyun:AccessKey}**”。当策略执行时，策略变量将被替换为请求本身的用户名或者 AccessKeyID。

示例 1：将含有策略变量的策略附加给多个用户，当用户 A 发起请求时，username 将替换为 A 的用户名。当用户 B 发起请求时，username 将替换为 B 的用户名。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "oos:GetObject",
        "oos:PutObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:ctyun:oos::123456789012:mybucket/${ctyun:username}/*"]
    }
  ]
}
```

示例 2：将含有策略变量的策略附加给多个用户，当用户 A 发起请求时，条件键 oos:prefix 将根据用户 A 的 username 进行判断。当用户 B 发起请求时，条件键 oos:prefix 将根据用户 B 的 username 进行判断。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": ["oos:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:ctyun:oos::123456789012:mybucket"],
      "Condition": {"StringLike": {"oos:prefix": ["${ctyun:username}/*"]}}
    }
  ]
}
```

7.9 操作权限与 API 对应关系

说明：下列表格中“**涉及资源**”列表示操作权限对应的资源（resource），括号内为生效示例。当资源范围为*时，表示将所有资源都赋予策略中的 Action。建议您在分配资源时尽量不使用*，以避免分配过多的资源。

表1 OOS 的操作权限与 API 对应关系

操作权限		涉及资源	API
Bucket 列表	ListBucket	Bucket（*bucketname* 或*）	GET Bucket（List Objects）、HEAD Bucket
	ListAllMyBucket	所有（*）	GET Service
	GetRegions	所有（*）	GET Regions
Bucket 读取	ListBucketMultipartUploads	Bucket（*bucketname* 或*）	List Multipart Uploads
	GetBucketAcl	Bucket（*bucketname* 或*）	GET Bucket acl
	GetBucketLocation	Bucket（*bucketname* 或*）	GET Bucket location
	GetBucketPolicy	Bucket（*bucketname* 或*）	GET Bucket policy
	GetLifecycleConfiguration	Bucket（*bucketname* 或*）	GET Bucket lifecycle
	GetBucketWebsite	Bucket（*bucketname* 或*）	GET Bucket website
	GetBucketCORS	Bucket（*bucketname* 或*）	GET Bucket cors
	GetBucketLogging	Bucket（*bucketname* 或*）	GET Bucket logging
	GetBucketObjectLockConfiguration	Bucket（*bucketname* 或*）	GET Bucket Object Lock
	GetBucketInventoryConfiguration	Bucket（*bucketname* 或*）	GET Bucket Inventory Configuration、List Bucket Inventory Configuration
Bucket 写入	PutBucket	Bucket（*bucketname* 或*）	PUT Bucket
	DeleteBucket	Bucket（*bucketname* 或*）	DELETE Bucket
	DeleteMultipleObjects	Bucket（*bucketname* 或*）	DELETE Multiple Objects
	PutLifecycleConfiguration	Bucket（*bucketname* 或*）	PUT Bucket lifecycle、DELETE Bucket lifecycle
	PutBucketWebsite	Bucket（*bucketname* 或*）	PUT Bucket website

	DeleteBucketWebsite	Bucket（*bucketname* 或*）	DELETE Bucket website
	PutBucketCORS	Bucket（*bucketname* 或*）	PUT Bucket cors、DELETE Bucket cors
	PutBucketLogging	Bucket（*bucketname* 或*）	PUT Bucket logging
	PutBucketObjectLockConfiguration	Bucket（*bucketname* 或*）	PUT Bucket Object Lock
	DeleteBucketObjectLockConfiguration	Bucket（*bucketname* 或*）	DELETE Bucket Object Lock
	PutBucketInventoryConfiguration	Bucket（*bucketname* 或*）	PUT Bucket Inventory Configuration、DELETE Bucket Inventory Configuration
Bucket 权限	PutBucketPolicy	Bucket（*bucketname* 或*）	PUT Bucket policy
	DeleteBucketPolicy	Bucket（*bucketname* 或*）	DELETE Bucket policy
Object 读取	ListMultipartUploadParts	Object（*bucketname*/*objectname*、*bucketname*/*或*）	List Parts
	GetObject	Object（*bucketname*/*objectname*、*bucketname*/*或*）	GET Object、HEAD Object
Object 写入	PutObject	Object（*bucketname*/*objectname*、*bucketname*/*或*）	PUT Object、PUT Object-Copy、POST Object、Initiate Mulitipart Upload、Upload Part、Compelete Multipart Upload、Upload Part －Copy

	DeleteObject	Object（*bucketname/objectname*、*bucketname*/*或*）	DELETE Object
	AbortMultipartUpload	Object（*bucketname/objectname*、*bucketname*/*或*）	Abort Multipart Upload

表2 统计的操作权限与 API 对应关系

操作权限	**涉及资源**	**API**
GetAccountStatistcsSummary	所有（*）	GET Capacity、GET DeleteCapacity、GET Traffics、GET AvailableBandwidth、GET Requests、GET RetarnCode、GET ConcurrentConnection、GET Usage、GET AvailBW、GET Bandwidth、Get Connection

表3 操作跟踪的操作权限与 API 对应关系

操作权限		**涉及资源**	**API**
列表	DescribeTrails	trail（trail/*或*）	DescribeTrails
	LookupEvents	trail（trail/*或*）	LookupEvents
读取	GetEventSelectors	trail（trail/*trailname*、trail/*或*）	GetEventSelectors
	GetTrailStatus	trail（trail/*trailname*、trail/*或*）	GetTrailStatus
写入	PutEventSelectors	trail（trail/*trailname*、trail/*或*）	PutEventSelectors
	StopLogging	trail（trail/*trailname*、trail/*或*）	StopLogging
	CreateTrail	trail（trail/*trailname*、trail/*或*）	CreateTrail
	UpdateTrail	trail（trail/*trailname*、trail/*或*）	UpdateTrail
	DeleteTrail	trail（trail/*trailname*、trail/*或*）	DeleteTrail
	StartLogging	trail（trail/*trailname*、trail/*或*）	StartLogging

表4 IAM 的操作权限与 API 对应关系

操作权限		涉及资源	API
列表	GetAccountSummary	所有（*）	GetAccountSummary
	GetLoginProfile	user（user/*username*、user/*或*）	GetLoginProfile
	ListAccessKeys	user（user/*username*、user/*或*）	ListAccessKeys
	ListUsers	user（user/*或*）	ListUsers
	ListUserTags	user（user/*username*、user/*或*）	ListUserTags
	ListGroups	group（group/*或*）	ListGroups
	ListGroupsForUser	user（user/*username*、user/*或*）	ListGroupsForUser
	ListPolicies	policy（policy/*或*）	ListPolicies
	ListAttachedGroupPolicies	group（group/*groupname*、group/*或*）	ListAttachedGroupPolicies
	ListAttachedUserPolicies	user（user/*username*、user/*或*）	ListAttachedUserPolicies
	ListEntitiesForPolicy	policy（policy/*policyname*、policy/*或*）	ListEntitiesForPolicy

	ListMFADevices	user（user/*username*、user/*或*）	ListMFADevices
	ListVirtualMFADevices	mfa（mfa/*或*）	ListVirtualMFADevices
读取	GetUser	user（user/*username*、user/*或*）	GetUser
	GetAccessKeyLastUsed	user（user/*username*、user/*或*）	GetAccessKeyLastUsed
	GetGroup	group（group/*groupname*、group/*或*）	GetGroup
	GetPolicy	policy（policy/*policyname*、policy/*或*）	GetPolicy
	GetAccountPasswordPolicy	所有（*）	GetAccountPasswordPolicy
	GetAccountLoginSecurityPolicy	所有（*）	GetAccountLoginSecurityPolicy
写入	CreateAccessKey	user（user/*username*、user/*或*）	CreateAccessKey
	DeleteAccessKey	user（user/*username*、user/*或*）	DeleteAccessKey
	UpdateAccessKey	user（user/*username*、user/*或*）	UpdateAccessKey

	CreateUser	user（user/*username*、user/*或*）	CreateUser
	DeleteUser	user（user/*username*、user/*或*）	DeleteUser
	TagUser	user（user/*username*、user/*或*）	TagUser
	UntagUser	user（user/*username*、user/*或*）	UntagUser
	CreateGroup	group（group/*groupname*、group/*或*）	CreateGroup
	DeleteGroup	group（group/*groupname*、group/*或*）	DeleteGroup
	AddUserToGroup	group（group/*groupname*、group/*或*）	AddUserToGroup
	RemoveUserFromGroup	group（group/*groupname*、group/*或*）	RemoveUserFromGroup
	ChangePassword	user（user/*username*、user/*或*）	ChangePassword
	UpdateAccountPasswordPolicy	所有（*）	UpdateAccountPasswordPolicy

	DeleteAccountPasswordPolicy	所有（*）	DeleteAccountPasswordPolicy
	UpdateAccountLoginSecurityPolicy	所有（*）	UpdateAccountLoginSecurityPolicy
	DeleteAccountLoginSecurityPolicy	所有（*）	DeleteAccountLoginSecurityPolicy
	CreateVirtualMFADevice	mfa（mfa/*mfaname*、mfa/*或*）	CreateVirtualMFADevice
	DeactivateMFADevice	user（user/*username*、user/*或*）	DeactivateMFADevice
	DeleteVirtualMFADevice	mfa（mfa/*mfaname*、mfa/*或*）	DeleteVirtualMFADevice
	EnableMFADevice	user（user/*username*、user/*或*）	EnableMFADevice
	CreateLoginProfile	user（user/*username*、user/*或*）	CreateLoginProfile
	DeleteLoginProfile	user（user/*username*、user/*或*）	DeleteLoginProfile
	UpdateLoginProfile	user（user/*username*、user/*或*）	UpdateLoginProfile
权限	CreatePolicy	policy（policy/*policyname*、policy/*或*）	CreatePolicy
	DeletePolicy	policy（policy/*policyname*、policy/*或*）	DeletePolicy

	AttachUserPolicy	user（user/*username*、user/*或*）	AttachUserPolicy
	DetachUserPolicy	user（user/*username*、user/*或*）	DetachUserPolicy
	AttachGroupPolicy	group（group/*groupname*、group/*或*）	AttachGroupPolicy
	DetachGroupPolicy	group（group/*groupname*、group/*或*）	DetachGroupPolicy