

对象存储
(Object-Oriented Storage, OOS)
开发者文档 V6

中国电信股份有限公司
云计算分公司

目录

1 对象存储（OOS）	8
1.1 产品介绍.....	8
1.2 产品优势.....	8
1.3 存储类型.....	8
2 主要概念.....	10
2.1 Account.....	10
2.2 Service.....	10
2.3 Bucket.....	11
2.3.1 Bucket 的命名规范	11
2.3.2 Bucket 的基本操作	11
2.3.3 Bucket 的数据位置和索引位置	12
2.4 Object.....	12
2.5 通过 IPv6 访问 OOS.....	12
2.6 统计.....	14
2.6.1 什么是统计分析.....	14
2.6.2 基本概念.....	14
2.7 操作跟踪.....	15
2.7.1 什么是操作跟踪.....	15
2.7.2 基本概念.....	15
2.8 访问控制.....	16
2.8.1 什么是 IAM.....	16
2.8.2 基本概念.....	16
2.9 Endpoint 列表.....	18
3 安全策略.....	19
3.1 用户签名验证 (V2)	19
3.1.1 Head 中包含签名.....	19
3.1.2 使用查询参数验证.....	25
3.2 用户签名验证 (V4)	27

3.2.1 Signature Version 4 的工作原理.....	27
3.2.2 认证方法.....	27
3.2.3 使用 Authorization 请求头验证.....	27
3.2.4 使用查询参数验证.....	39
3.3 Bucket 权限控制.....	45
3.4 Bucket Policy 安全策略.....	46
3.4.1 介绍.....	46
3.4.2 Bucket Policy 元素.....	47
3.4.3 示例.....	52
3.5 合规保留.....	54
4 HTTP REST 接口（OOS API）	56
4.1 接口请求/响应头.....	56
4.1.1 通用请求头.....	56
4.1.2 通用响应头.....	56
4.2 关于 Service 的操作.....	57
4.2.1 GET Service（List Bucket）	57
4.2.2 GET Regions.....	57
4.3 关于 Bucket 的操作.....	58
4.3.1 PUT Bucket.....	58
4.3.2 GET Bucket location.....	63
4.3.3 GET Bucket acl.....	65
4.3.4 GET Bucket（List Objects）	67
4.3.5 DELETE Bucket.....	70
4.3.6 PUT Bucket Policy.....	71
4.3.7 GET Bucket Policy.....	72
4.3.8 DELETE Bucket Policy.....	73
4.3.9 PUT Bucket WebSite.....	74
4.3.10 GET Bucket WebSite.....	76
4.3.11 DELETE Bucket WebSite.....	78

4.3.12 List Multipart Uploads.....	78
4.3.13 PUT Bucket Logging.....	85
4.3.14 GET Bucket Logging.....	88
4.3.15 HEAD Bucket.....	90
4.3.16 PUT Bucket Lifecycle.....	90
4.3.17 GET Bucket Lifecycle.....	99
4.3.18 DELETE Bucket Lifecycle.....	101
4.3.19 PUT Bucket accelerate.....	102
4.3.20 GET Bucket accelerate.....	104
4.3.21 PUT Bucket CORS.....	105
4.3.22 GET Bucket CORS.....	109
4.3.23 DELETE Bucket CORS.....	111
4.3.24 PUT Bucket object lock configuration.....	111
4.3.25 GET Bucket object lock configuration.....	114
4.3.26 Delete Bucket object lock configuration.....	116
4.4 关于 Object 的操作.....	118
4.4.1 PUT Object.....	118
4.4.2 GET Object.....	120
4.4.3 DELETE Object.....	123
4.4.4 PUT Object - Copy.....	124
4.4.5 Initial Multipart Upload.....	128
4.4.6 Upload Part.....	131
4.4.7 Complete Multipart Upload.....	133
4.4.8 Abort Multipart Upload.....	137
4.4.9 List Part.....	138
4.4.10 Copy Part.....	142
4.4.11 Delete Multiple Object.....	144
4.4.12 断点续传.....	147
4.4.13 POST Object.....	149

4.4.14 OPTIONS Object.....	160
4.4.15 生成共享链接.....	161
4.4.16 HEAD Object.....	162
4.5 关于 AccessKey 的操作.....	164
4.5.1 CreateAccessKey.....	164
4.5.2 DeleteAccessKey.....	164
4.5.3 UpdateAccessKey.....	164
4.5.4 ListAccessKey.....	164
4.5.5 STS 临时授权访问	164
4.6 Backoff 说明.....	166
4.7 错误响应.....	167
4.7.1 REST 错误响应.....	167
4.7.2 错误码列表.....	167
4.8 使用 HttpURLConnection 开发.....	172
5 统计 API.....	176
5.1 统计 API 请求结构	176
5.1.1 公共请求头.....	176
5.1.2 状态响应码.....	177
5.1.3 公共响应头.....	177
5.2 统计 API 概览	178
5.3 统计 API.....	179
5.3.1 GetCapacity.....	179
5.3.2 GetBilledStorageUsage.....	184
5.3.3 GetRestoreCapacity	189
5.3.4 GetDeleteCapacity	193
5.3.5 GetTraffics	198
5.3.6 GetRequests	205
5.3.7 GetReturnCode	213
5.3.8 GetConcurrentConnection	227

5.3.9 GetUsage	231
5.3.10 GetBandwidth	235
5.4 错误码列表.....	240
6 操作跟踪 API	241
6.1 操作跟踪 API 请求结构	241
6.1.1 公共请求头.....	241
6.1.2 状态响应码.....	242
6.1.3 公共响应头.....	242
6.2 操作跟踪 API 概览	243
6.3 操作跟踪 API	244
6.3.1 CreateTrail	244
6.3.2 DeleteTrail	246
6.3.3 DescribeTrails	248
6.3.4 GetTrailStatus	250
6.3.5 PutEventSelectors	252
6.3.6 GetEventSelectors.....	254
6.3.7 UpdateTrail	256
6.3.8 StartLogging	258
6.3.9 StopLogging.....	260
6.3.10 LookupEvents	262
6.4 错误码列表.....	267
7 访问控制（IAM）API	274
7.1 IAM API 请求结构	274
7.1.1 公共参数.....	274
7.1.2 响应结果.....	276
7.2 访问控制 API 概览	277
7.3 用户管理接口.....	280
7.3.1 CreateUser.....	280
7.3.2 GetUser	283

7.3.3 ListUsers	285
7.3.4 DeleteUser.....	288
7.3.5 TagUser.....	290
7.3.6 UntagUser	292
7.3.7 ListUserTags	294
7.3.8 ListGroupsForUser	296
7.3.9 CreateAccessKey	299
7.3.10 ListAccessKeys.....	301
7.3.11 UpdateAccessKey	304
7.3.12 DeleteAccessKey	306
7.3.13 CreateLoginProfile.....	308
7.3.14 GetLoginProfile	311
7.3.15 UpdateLoginProfile.....	313
7.3.16 DeleteLoginProfile.....	315
7.3.17 ChangePassword	317
7.3.18 CreateVirtualMFADevice.....	319
7.3.19 EnableMFADevice	321
7.3.20 ListVirtualMFADevices	323
7.3.21 ListMFADevices.....	326
7.3.22 DeactivateMFADevice	329
7.3.23 DeleteVirtualMFADevice.....	331
7.4 用户组管理接口.....	333
7.4.1 CreateGroup.....	333
7.4.2 GetGroup.....	335
7.4.3 AddUserToGroup	338
7.4.4 RemoveUserFromGroup.....	340
7.4.5 ListGroups.....	342
7.4.6 DeleteGroup.....	345
7.5 权限策略管理接口.....	347

7.5.1 CreatePolicy	347
7.5.2 GetPolicy.....	350
7.5.3 ListPolicies.....	353
7.5.4 ListEntitiesForPolicy	356
7.5.5 DeletePolicy	359
7.5.6 AttachUserPolicy	361
7.5.7 ListAttachedUserPolicies.....	363
7.5.8 DetachUserPolicy	366
7.5.9 AttachGroupPolicy	368
7.5.10 ListAttachedGroupPolicies	370
7.5.11 DetachGroupPolicy	373
7.5.12 UpdateAccountPasswordPolicy	375
7.5.13 GetAccountPasswordPolicy	378
7.5.14 DeleteAccountPasswordPolicy	381
7.6 服务数量查询.....	382
7.6.1 GetAccountSummary.....	382
7.7 错误码列表.....	384
7.8 IAM 策略编写规则	394
7.8.1 Version.....	394
7.8.2 Statement.....	394
7.9 操作权限与 API 对应关系	407

1 对象存储（OOS）

1.1 产品介绍

对象存储（Object-Oriented Storage, OOS）是中国电信为客户提供的一种海量、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。

1.2 产品优势

- 1) 容量几乎没有上限，弹性扩容；
- 2) 自动数据冗余和故障切换，确保高可用；
- 3) 流量按需计费，节省带宽成本；
- 4) 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。如果上载对象时未指定存储类型，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。低频访问存储的对象有最短存储时间，存储时间短于 30 天的对象被提前删除或变更时，会产生一定费用。低频访问存储对象有最小计费大小，即如果对

象大小低于 64KB，会按照 64KB 计算收费，对象大于等于 64KB 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	99.9999999999% (13 个 9)	99.9999999999% (13 个 9)
服务设计的可用性	99.9%	99.9%
对象最小计费大小	按照对象实际大小计算	64KB
最少存储时间	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

支持对象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（对象容器）和 Object（对象）。它们之间的关系如图 1 所示。在使用 OOS 之前，首先需要在 www.ctyun.cn 注册一个账户（Account），注册成功之后，OOS 会为该账户提供服务（Service），在该服务下，用户可以创建 1 个或多个对象容器（Bucket），每个对象容器中可以存储不限数量的对象（Object）。

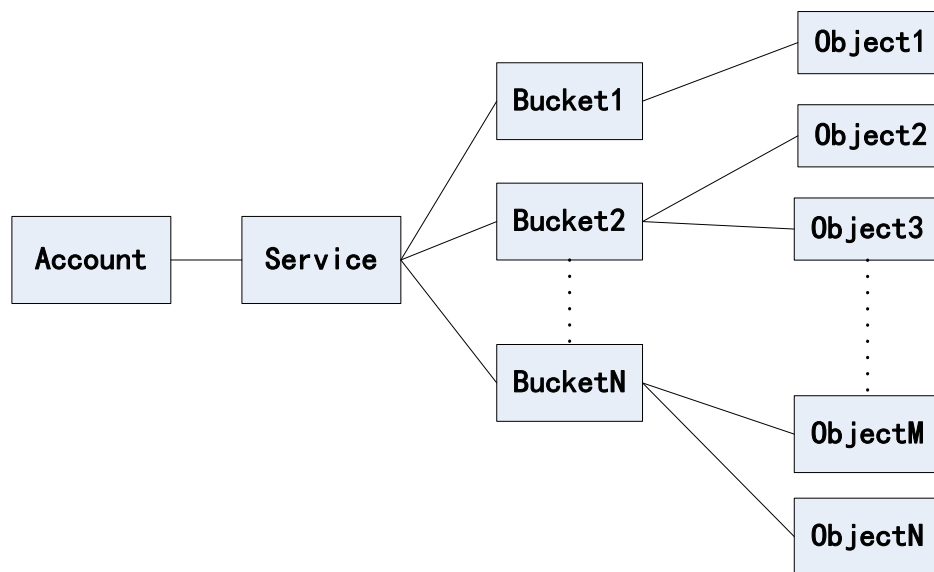


图 1 OOS 主要概念之间的关系

2.1 Account

在使用 OOS 之前，需要在天翼云网站 www.ctyun.cn 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，在**控制台**页面点击**开通 OOS 服务**，提交订单后便可开通。开通成功之后，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 个的对象容器（Bucket）。

2.3 Bucket

Bucket 是存储 Object 的容器。中国电信天翼云对象存储系统的每个 Object 都必须包含在一个 Bucket 中。您可以设置容器的属性，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容器内的所有对象，因此您可以通过灵活的属性设置，来创建不同的容器，完成不同的管理功能。每个用户最多可以建立 10 个 Bucket。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 Bucket。

Bucket 的命名方式如下：

- 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：
 - 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 IP 地址形式（如 192.162.0.1）；
- Bucket 名称不能是一组或多组“数字.数字”的组合；
- Bucket 名称中不能包含双横线（--）、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 Bucket 的基本操作

身份验证通过的用户可以新建 Bucket、删除 Bucket 以及查看 Bucket 的属性。

新建 Bucket 时需要输入 Bucket 的名称，选择操作权限（Bucket 的默认操作权限是 private）。Bucket 一旦创建成功，其名将不可更改。根用户和拥有 Bucket 相应权限的子用户可以更改 Bucket 的操作权限。

删除 Bucket：只有根用户和拥有删除权 Bucket 权限的子用户才能删除 Bucket，且被删除的 Bucket 中不能包含任何对象。

查看属性：查看 Bucket 的属性时，用户可以更改 Bucket 的操作权限。

2.3.3 Bucket 的数据位置和索引位置

在创建 Bucket 时，需要指定数据的数据位置和索引位置，数据位置指的是存放对象数据的位置，索引位置是指存放对象数据索引信息的位置。

在创建 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 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

用户存储在 OOS 上的每个文件都是一个 Object。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 Object。同时用户还可以对 Object 的组织形式进行管理，将 Object 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 的使用情况、指定数据域的使用情况。用户可以根据统计分析数据，采取对应的措施。

2.6.2 基本概念

- **直接流量-互联网流量**：从互联网上传下载对象数据，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）上传下载对象数据，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**：从互联网上传下载对象，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）上传下载对象，并且经过对象存储网络内部调度产生的流量。
- **删除容量**：已删除对象的大小。

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元素等。

操作跟踪功能主要包括：

- **管理事件记录：**用户可以通过操作跟踪功能查看近 180 天内管理事件，包括查看、创建、修改和删除资源的管理事件。
- **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

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。

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。

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

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。

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，新用户无权执行任何 OOS 操作或访问任何 OOS 资源。

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。

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。

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。

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：天翼云 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

OOS 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。对于 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。）

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。

OOS 对象存储具体资源池 Endpoint 如下：

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

3 安全策略

为了保证用户数据的安全，OOS 提供三种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、Policy 安全策略。

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/AccessSecretKey 对称加密的方法来验证某个请求的发送者身份。AccessKeyID 和 AccessSecretKey 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 AccessSecretKey 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 AccessSecretKey，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 Head 中包含签名

用户可以在 HTTP 请求中增加 Authorization（授权）的 Head 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;

CanonicalizedResource = [ "/" + Bucket ] +
<HTTP-Request-URI > +
[<sub-resource>];
CanonicalizedAmzHeaders

```

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"。

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 AccessSecretKey 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:

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

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) 删除请求头和内容之间分隔符两端出现的任何空格。如 'x-amz-meta-name: fred, barney' 转换成: 'x-amz-meta-name: fred, barney'。
- 5) 将所有的头和内容用 ' \n ' 分隔符分隔拼成最后的 CanonicalizedAMZHeader。

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。此时的 CanonicalizedResource 例如： /BucketName/ObjectName?acl
- 在构造 CanonicalizedResource 时，必须包含的子资源包括：acl, lifecycle, location, logging, notification, partNumber, policy, requestPayment, torrent, uploadId, uploads, versionId, versioning, versions 和 website。

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。

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

当发送批量删除对象请求时，delete 参数需要包含在 CanonicalizedResource 中。

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。

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。

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。

示例

1. GET Object

从名为 johnsmith 的 bucket 中 get 对象

请求	StringToSign
GET /photos/puppy.jpg HTTP/1.1 Host: johnsmith.oos-cn.ctyunapi.cn Date: Tue, 27 Mar 2007 19:36:42 +0000 <i>Authorization: AWS</i> <i>7799e793ce4624ee7e5a:xXjDGYUmKxn</i> <i>wqr5KXNPGldn5LbA=</i>	GET\n \n \n Tue, 27 Mar 2007 19:36:42 +0000\n /johnsmith/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。

2. PUT Object

向名为 johnsmith 的 bucket 中上传一个对象。

请求	StringToSign
PUT /photos/puppy.jpg HTTP/1.1 Content-Type: image/jpeg Content-Length: 94328 Host: johnsmith.oos-cn.ctyunapi.cn Date: Tue, 27 Mar 2007 21:15:45 +0000 <i>Authorization: AWS</i> <i>7799e793ce4624ee7e5a:hcicpDDvL9Ss06A</i> <i>kvxqmIWkmOuQ=</i>	PUT\n \n image/jpeg\n Tue, 27 Mar 2007 21:15:45 +0000\n /johnsmith/photos/puppy.jpg

注意：Content-Type 请求头包含在请求中，也包含在 StringToSign 中。但请求中没有 Content-MD5 请求头，所以 StringToSign 中是空行。

3. List Objects

list 名为 johnsmith 的 bucket 的对象。

请求	StringToSign
GET /?prefix=photos&max-keys=50&marker=puppy HTTP/1.1 User-Agent: Mozilla/5.0 Host: johnsmith.oos-cn.ctyunapi.cn Date: Tue, 27 Mar 2007 19:42:41 +0000 <i>Authorization: AWS</i> 7799e793ce4624ee7e5a:jsRt/rhG+Vtp88HrYL 706QhE4w4=	GET\n \n \n Tue, 27 Mar 2007 19:42:41 +0000\n /johnsmith/

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 johnsmith 的 bucket 的访问控制权限配置信息。

请求	StringToSign
GET /?acl HTTP/1.1 Host: johnsmith.oos-cn.ctyunapi.cn Date: Tue, 27 Mar 2007 19:44:46 +0000 <i>Authorization: AWS</i> 7799e793ce4624ee7e5a:thdUi9VAkzhkniLj 96JIrOPGi0g=	GET\n \n \n Tue, 27 Mar 2007 19:44:46 +0000\n /johnsmith/?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 johnsmith 的 bucket 中删除对象。bucket 在 path 中指定，并使用 x-amz-date 请求头。

请求	StringToSign
DELETE /johnsmith/photos/puppy.jpg HT TP/1.1 User-Agent: dotnet Host: oos-cn.ctyunapi.cn Date: Tue, 27 Mar 2007 21:20:27 +0000 x-amz-date: Tue, 27 Mar 2007 21:20:26	DELETE\n \n \n \n x-amz-date:Tue, 27 Mar 2007 21:20:26 +0000\n

+0000 <i>Authorization: AWS</i> 7799e793ce4624ee7e5a:k3nL7gH3+PadhT EVn5Ip83xlYzk=	/johnsmith/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 形式上传对象

下面的例子通过 CNAME 形式上传对象，并使用自定义元数据。

请求	StringToSign
PUT /db-backup.dat.gz HTTP/1.1 User-Agent: curl/7.15.5 Host: static.johnsmith.net:8080 Date: Tue, 27 Mar 2007 21:06:08 +0000 x-amz-acl: public-read content-type: application/x-download Content-MD5: 4gJE4saaMU4BqNR0kLY+lw== X-Amz-Meta-ReviewedBy: joe@johnsmith.net X-Amz-Meta-ReviewedBy: jane@johnsmith.net X-Amz-Meta-FileChecksum: 0x02661779 X-Amz-Meta-ChecksumAlgorithm: crc32 Content-Disposition: attachment; file name=database.dat Content-Encoding: gzip Content-Length: 5913339 <i>Authorization: AWS</i> 7799e793ce4624ee7e5a:C0FI0tU8Ylb9KD TpZqYkZPX91iI=	PUT\n4gJE4saaMU4BqNR0kLY+lw==\napplication/x-download\nTue, 27 Mar 2007 21:06:08 +0000\nx-amz-acl:public-read\nx-amz-meta-checksumalgorithm:crc32\nx-amz-meta-filechecksum:0x02661779\nx-amz-meta-reviewedby: joe@johnsmith.net,jane@johns mith.net\n/static.johnsmith.net/dbbackup. 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: Wed, 28 Mar 2007 01:29:59 +0000 <i>Authorization: AWS</i> <i>7799e793ce4624ee7e5a:Db+gepJSUbZK</i> <i>wpx1FR0DLtEYoZA=</i>	GET\n \n \n Wed, 28 Mar 2007 01:29:59 +0000\n /
---	--

8. 对象名被编码

请求	StringToSign
GET /diction ary/fran%C3%A7ais/pr%C3%a9f%C3%a 8re HT TP/1.1 Host: oos-cn.ctyunapi.cn Date: Wed, 28 Mar 2007 01:49:49 +0000 <i>Authorization: AWS</i> <i>7799e793ce4624ee7e5a:dxhSBHoI6eVSP</i> <i>cXJqEghlUzZMnY=</i>	GET\n \n \n Wed, 28 Mar 2007 01:49:49 +0000\n /diction ary/fran%C3%A7ais/pr%C3%a9f%C3%a 8re

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。

注意：使用预签名 URL 方式，有将您授权的数据在过期时间内暴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/nelson?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

请求字符串参数名称	示例中的值	描述
AWSAccessKeyId	44CF9590006BF252F707	可以登录到对象存储控制 住中心，点击 访问控制—安全凭证—密钥 ，查看到 AccessKeyId 和 AccessSecretKey。
Expires	1141889120	定义里签名过期时间，按 照秒来计算的。服务器端 超过这个时间的请求将被 拒绝。 有效时长最多为 7 天，如 果您配置过期时间大于 7 天，有效期时间仍是 7 天，7 天过期后会返回提 示信息。
Signature	vjbyPxybdZaNmGa%2ByT272 YEAiv4%3D	URL 是按照 HMAC-SHA1 的 StringToSign 的 Base64 编码方式编码。

Expires 这个参数的值是一个 UNIX 时间（自 UTC 时间 1970 年 1 月 1 号开始的秒数，详见 wiki），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。

所有的 OOS 支持的请求和各种 Head 参数，URL 中包含签名的方法和 Head 中包含签名的算法基本一样，主要区别如下：

- 1) 通过 URL 包含签名时，之前的 Date 参数换成 Expires 参数。
- 2) 不支持同时在 URL 和 Head 中包含签名。
- 3) 如果传入的 Signature, Expires, AWSAccessKeyId 出现不止一次，以第一次为准。
- 4) 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

客户端按照如下方式创建签名：

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 SecretKey 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：

1. 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。
2. 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=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f
1024
```

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： <pre><your-access-key-id>/<date>/<region>/<service>/aws4_request</pre> 其中： ● date 格式为 YYYYMMDD。 ● region: ■ 对于 oos api：访问域名为 oos-xx.ctyunapi.cn，region 为 xx。各资源池的详细访问域名 详见 Endpoint 列表； ■ 对于统计 api：访问域名为 oos-cn-mg.ctyunapi.cn，region 为 cn-mg； ■ 对于操作跟踪 api：访问域名为 oos-cn-cloudtrail.ctyunapi.cn，region 为 cn； ■ 对于 iam api：访问域名为 oos-cn-iam.ctyunapi.cn，region 为 cn； ● 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。

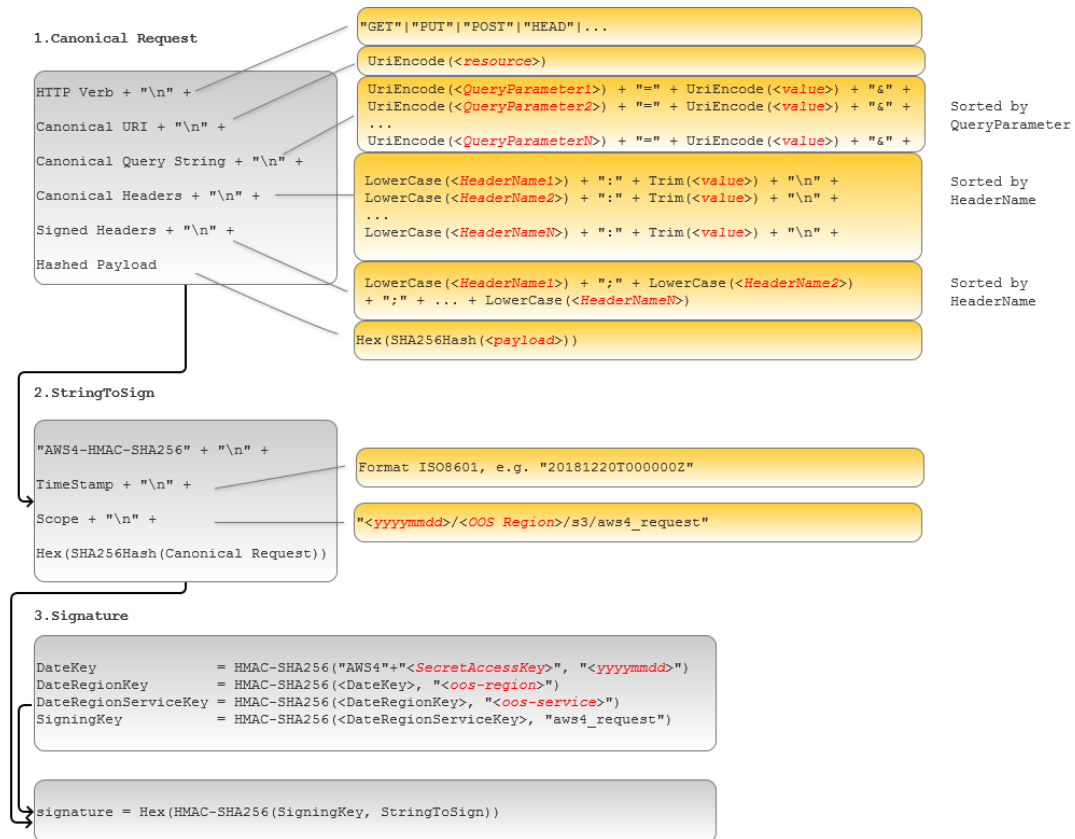
- **无签名负载方式：**在签名计算中不包括负载的 checksum。

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：

OOS 开发者文档



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"。 - 除了对象名之外，对正斜杠字符 '/' 进行编码。例如，如果对象名称为 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/examplebucket/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/examplebucket?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/examplebucket?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。需要按字母顺序对请求头名称进行排序，示例如下：

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

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

```
host:s3.amazonaws.com
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
```

HashedPayload 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 请求上传对象时，用户可以在请求体中提供对象数据。使用 GET 请求检索对象时，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"
```

对于 OOS，服务字符串是 s3。Region 为域名 oos-xx.ctyunapi.cn 中的 xx 部分。

3. 计算签名

使用 secretKey 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>, "<oos-region>") DateRegionServiceKey
= HMAC-SHA256(<DateRegionKey>, "<oos-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
AWSAccessKeyId	2a948fd3f00ba0925806
AWSSecretAccessKey	ef2017c2e5ffa0b1761717ecbca021da16501384

Bucket 名称：examplebucket。

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

1. 示例：GET 对象

从存储桶 examplebucket 中获取对象 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: examplebucket.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:examplebucket.oos-cn.ctyunapi.cn
range:bytes=0-9
x-amz-content-
sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b85
5
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
bca722269a76aadb00dfe5a50fefdbd5712065267e1692cc596cefd2681f5d14
```

- 2) 生成签名密钥

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

3) 计算后的签名

```
be3f55b78165716c51ce37f588048f858fc27f7449d8fe74f887d999e5fc9193
```

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=be3f55b78165716c51ce37f588048f858fc27f7449d8fe74f887d999e5fc9193
```

2. 示例：PUT 对象

在存储桶 examplebucket 中上传对象 test.txt。

```
PUT /examplebucket/test.txt HTTP/1.1  
x-amz-content-sha256:  
7509e5bda0c762d2bac7f90d758b5b2263fa01ccbc542ab5e3df163be08e6ca9  
Authorization:SignatureToBeCalculated  
x-amz-date: 20190220T070722Z  
x-amz-storage-class: STANDARD  
Host: 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  
/examplebucket/test.txt  
  
content-length:12  
host: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
66919f4f7f555dec8599c5894bbd5c104767bbf0180103d751653143f67a8d45
```

2) 生成签名密钥

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

3) 计算后的签名

```
29407b3d2010ab3f86e313302a4d952d8ac0070364cd91ba3b113258a4d36b9b
```

4) Authorization 请求头

```
Authorization: AWS4-HMAC-SHA256
Credential=2a948fd3f00ba0925806/20190220/cn/s3/aws4_request,
SignedHeaders=content-length;host;x-amz-content-sha256;x-amz-date;x-amz-storage-class,
Signature=29407b3d2010ab3f86e313302a4d952d8ac0070364cd91ba3b113258a4d36b9b
```

3. 示例：列出存储桶中的对象

OOS 开发者文档

列出存储桶 examplebucket 中的对象，prefix 设置为“t”，最多返回 2 个对象。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: examplebucket.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:examplebucket.oos-cn.ctyunapi.cn
x-amz-content-
sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b85
5
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
bc2b6af0cbb17679b2697f7239b02dc21d4b62fc30e197441cf900d35d3b103
```

- 2) 生成签名密钥

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

3) 计算后的签名

```
ce5ef3764d4a34b4e3c81d37b9a310432e5c4bf8bb4722c14877adba882fc559
```

4) Authorization 请求头

```
Authorization: AWS4-HMAC-SHA256
Credential=2a948fd3f00ba0925806/20190220/cn/s3/aws4_request,
SignedHeaders=host;x-amz-content-sha256;x-amz-date,
Signature=ce5ef3764d4a34b4e3c81d37b9a310432e5c4bf8bb4722c14877adba882fc559
```

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）中使用它来下载对象。

预签名 URL 支持 GET、DELETE、PUT、HEAD、POST 请求。

注意：使用预签名 URL 方式，有将您授权的数据在过期时间内暴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/examplebucket/test.txt
?X-Amz-Algorithm=AWS4-HMAC-SHA256 &X-Amz-Credential=<your-access-
key-id>/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 URL 和 Header 中都包含签名信息，以 URL 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=<your-access-key-id>%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	说明
X-Amz-Algorithm	请将此参数值设置为 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： <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。各资源池的详细访问域名详见 Endpoint 列表； ■ 对于统计 api: 访问域名为 oos-cn- mg.ctyunapi.cn, region 为 cn-mg； ■ 对于操作跟踪 api: 访问域名为 oos-cn- cloudtrail.ctyunapi.cn, region 为 cn； ■ 对于 iam api: 访问域名为 oos-cn- iam.ctyunapi.cn, region 为 cn；

	● 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“yyyyMMdT 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”。
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。

OOS 开发者文档



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”。
- 除了对象名之外，对正斜杠字符 ‘/’ 进行编码。例如，如果对象名称为 photos/Jan/sample.jpg，则不对键名称中的正斜杠进行编码。

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，让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 examplebucket 中 test.txt 对象，过期时间设置为 7 天（604800 秒）。以 GET 请求为例：

```
GET
http://oos-cn.ctyunapi.cn/examplebucket/test.txt
?X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=2a948fd3f00ba0925806/20190220/cn/s3/aws4_request
&X-Amz-Date=20190220T095256Z
&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
AWSAccessKeyId	2a948fd3f00ba0925806
AWSSecretAccess Key	ef2017c2e5ffa0b1761717ecbca021da16501384

- 1) StringToSign
 - a. 创建规范请求（以 GET 请求为例）

```
GET
/examplebucket/test.txt
X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Credential=2a948fd3f00ba0925806%2F20190220%2Fcn%2Fs3%2Faws4_request
&X-Amz-Date=20190220T095256Z&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  
20190220T095256Z  
20190220/cn/s3/aws4_request  
023d2e0e5fba779afb9fe621e9413622e7f1aff9ffc7348e55d0805d97cb1571
```

2) 生成签名密钥

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

3) 计算后的签名

```
f566134de06fb3daa22b9649baf82d15d6aa575e146b6ba9aff13a2bde63a1ec
```

4) 预签名 URL

```
http://oos-cn.ctyunapi.cn/examplebucket/test.txt?X-Amz-Algorithm=AWS4-  
HMAC-SHA256&X-Amz-  
Credential=2a948fd3f00ba0925806/20190220/cn/s3/aws4_request&X-Amz-  
Date=20190220T095256Z&X-Amz-Expires=604800&X-Amz-  
SignedHeaders=host&X-Amz-  
Signature=f566134de06fb3daa22b9649baf82d15d6aa575e146b6ba9aff13a2bde63a  
1ec
```

3.3 Bucket 权限控制

对象存储（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（只读）和 public（公有）：

- 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 Object 进行读写操作（包括 Put、Delete 和 Get Object）。其他人无法访问该 Bucket 内的 Object。

- public-read（只读）

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）可以对该 Bucket 中的 Object 进行读操作（Get Object）。

- public-read-write（公有）

任何人（包括匿名访问）都可以对该 Bucket 中的 Object 进行 Put，Get 和 Delete 操作。这些操作可能会造成 Bucket 所有者数据的增加或者丢失，且所有这些操作产生的费用由该 Bucket 的所有者承担，所以请慎用该权限。

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 Bucket Policy 安全策略

3.4.1 介绍

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

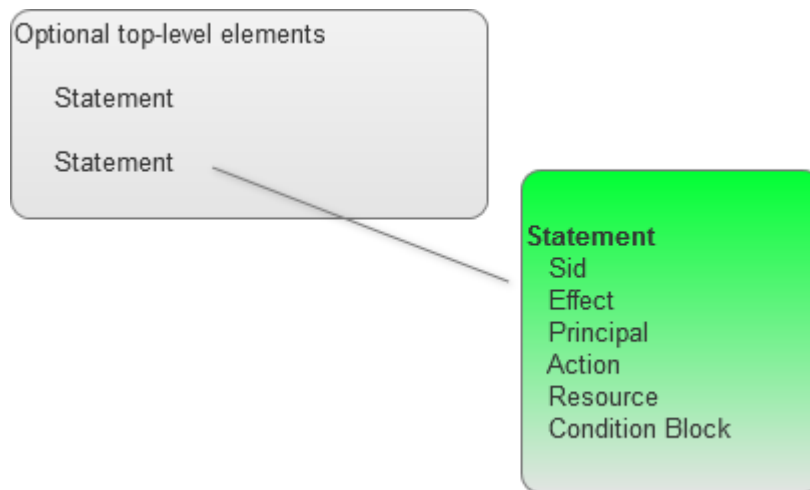
- 允许/拒绝 Bucket 级别的权限；
- 允许/拒绝 Object 级别的权限。

Policy 本身是一个 JSON 字符串，限制在 20KB。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。



3.4.2 Bucket Policy 元素

下面介绍 statements 中的各个元素。

3.4.2.1 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"
```

3.4.2.2 Id

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

3.4.2.3 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 用于指定被允许或被拒绝访问的操作，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 权限列表如下。

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

对象操作的 OOS 权限列表如下。

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"
```

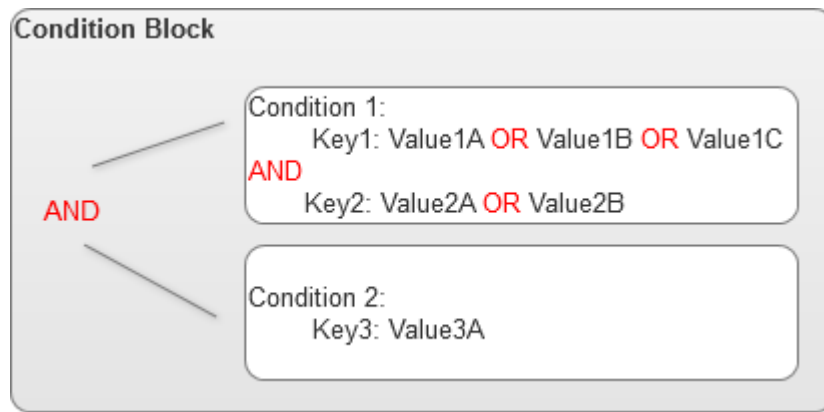
5. Resource

Resource 是指 Policy 覆盖的对象或对象集合。可以使用通配符*和?，必须以 arn:ctyun:oos:::bucketName/开头。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.mysite.com/”开头的请求，可以这样配置：

```

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

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 **SSL** 发出请求。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": { "CTYUN": ["*"] },
      "Action": "oos:PutObject",
      "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。

3.4.3 示例

下面是一个定义 Referer Policy 的例子

```
{
  "Version": "2012-10-17",
  "Id": "http referer policy example",
  "Statement": [
    {
      "Sid": "Allow get requests referred by www.mysite.com , mysite.com
and empty referer",
      "Effect": "Allow",
      "Principal": { "CTYUN": ["*"] },
      "Action": "oos:*",
      "Resource": "arn:ctyun:oos::example-bucket/*",
      "Condition": {
        "StringLike": {
          "ctyun:Referer": [
            "http://www.mysite.com/*",
            "http://mysite.com/*",
            ""
          ]
        }
      }
    }
  ]
}
```

1. 下面是一个定义 IP Policy 的例子

```
{
  "Version": "2012-10-17",
  "Id": "PolicyId1",
  "Statement": [
    {
      "Sid": "IPAllow",
      "Effect": "Allow",
      "Principal": {
        "CTYUN": "*"
      },
      "Action": "oos:*",

```

```

        "Resource": "arn:ctyun:oos:::example-bucket /*",
        "Condition" : {
            "IpAddress" : {
                "ctyun:SourceIp": "192.168.143.0/24"
            },
            "NotIpAddress" : {
                "ctyun:SourceIp": "192.168.143.188/32"
            }
        }
    }
}

```

2. 下面的例子可向匿名用户授予只读权限

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对象数据，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对象时，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:::examplebucket/*"]
        }
    ]
}

```

3.5 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对象进行修改和删除。

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，对象 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对象；
- 处于合规保留期的对象，无法通过调用 API、控制台修改对象的存储类型，只能通过生命周期修改存储类型。
- 处于合规保留期的对象，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对象过了合规保留期后才能生效。

示例：

对象 A1、A2、A3 和 A4 都属于 Bucket A，2020 年 2 月 1 日对 Bucket A 设置了合规保留时长为 30 天。对象 A1、A2 分别设置的了生命周期规则，对象 A3、A4 未设置生命周期规则。

对象	生命周期规则
A1	对象创建 10 天由标准存储转换为低频访问存储
A2	对象创建 10 天后删除
A3	无
A4	无

OOS 开发者文档

对象	对象创建时间	合规保留失效时间	生命周期规则生效能情况
A1	2020-03-01 00:00:00	2020-03-31 00:00:01	2020-03-11 后启动转换为低频访问存储操作
A2	2020-03-01 00:00:00	2020-03-31 00:00:01	待 2020-03-31 后启动删除操作。
A3	2020-01-01 00:00:00	不存在。因为合规保留 2020-02-01 创建的，在 2020-01-31 00:00:01 已经过了 30 天，故合规保留对 A3 不生效。	不涉及
A4	2020-01-29 00:00:00	2020-02-28 00:00:01	不涉及

4 HTTP REST 接口（OOS API）

4.1 接口请求/响应头

4.1.1 通用请求头

以下是 OOS API 的通用请求头。

名称	描述
Authorization	用于身份验证的请求头。
Content-Length	请求体的长度。
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。
Content-Type	描述请求内容的标准 MIME 类型。
Date	请求的日期和时间。当设置了 Authorization 请求头时，必须指定 x-amz-date 或者 Date 请求头。
x-amz-date	请求的日期和时间。当设置了 Authorization 请求头时，必须指定 x-amz-date 或者 Date 请求头。如果两者都指定了，将以 x-amz-date 的值为准。
Host	请求的域名。

4.1.2 通用响应头

以下是 OOS API 的通用响应头。

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	对象的哈希值。ETag 只反映了对象的内容，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。

4.2 关于 Service 的操作

4.2.1 GET Service (List Bucket)

对于做 Get 请求的服务，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。

返回结果

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

4.2.2 GET Regions

获取资源池中的索引位置和数据位置列表。

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
regions	表示要获取索引位置和数据位置的列表。

返回结果

```

HTTP/1.1 200 OK
x-amz-request-id: 236A8905248E5A01
Date: Mon, 03Sep 2017 12:00:00 GMT
Content-Length: 100
Content-Type: application/xml; charset=utf-8
Connection: close
Server: CTYUN

```

```

<BucketRegions>
  <MetadataRegions>
    <Region>ChengDu</Region>
    <Region>ShenYang</Region>
  </MetadataRegions>
  <DataRegions>
    <Region>ChengDu</Region>
    <Region>ShenYang</Region>
  </DataRegions>
</BucketRegions>

```

响应元素

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

4.3 关于 Bucket 的操作

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。

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 [2.3.1 Bucket 的命名规范](#)。

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>
```

请求元素

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	是
MetadataLocationConstraint	设置 Bucket 的索引位置。 类型：容器 父节点：CreateBucketConfiguration	创建 Bucket 的时候必须填写，修改 Bucket

		ACL 或数据索引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：String 有效值： - 父节点为 MetadataLocationConstraint，表示索引位置，有效值为：ChengDu、FuZhou、GuiYang、HangZhou、LaSa、LanZhou、QingDao、ShenYang、ShenZhen、WuHan、WuHu、WuLuMuQi、ZhengZhou - 父节点为 LocationList，表示数据位置，有效值为：ChengDu、GuiYang、LaSa、LanZhou、QingDao、SH2、ShenYang、ShenZhen、SuZhou、WuHan、WuHu、WuLuMuQi、ZhengZhou 默认值：无 父节点：MetadataLocationConstraint 或 Locationlist	否
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

请求参数

名称	描述	是否必须
x-amz-acl	设置 Bucket 的 ACL（Access Control List） 类型：String 有效值：private public-read public-read-write	否

请求示例 1

请求创建一个名叫 picture 的 bucket，索引位置设置为 Beijing，数据位置设置为优先本地，bucket 权限设置为公有。

```
PUT / HTTP/1.1
Host: picture.oos-cn.ctyunapi.cn
Content-Length:200
Content-Type: application/xml;charset=utf-8
Date: Mon, 03 Sep 2012 12:00:00 GMT
Authorization: AWS
7799e793ce4624ee7e5a:9N/W/kmYwFhbqHYFTvQor317qVw=
x-amz-acl: public

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

返回示例 1

```
HTTP/1.1 200 OK
x-amz-request-id: 236A8905248E5A01
Date: Mon, 03Sep 2012 12:00:00 GMT
Location: /picture
Content-Length: 0
Connection: close
Server: CTYUN
```

OOS 开发者文档

请求创建一个名叫 picture 的 bucket，索引位置设置为 Beijing，数据位置设置为北京、上海，调度策略是允许 OOS 自动调度：

```
PUT / HTTP/1.1
Host: picture.oos-cn.ctyunapi.cn
Content-Length:200
Content-Type: application/xml; charset=utf-8
Date: Mon, 03 Sep 2012 12:00:00 GMT
Authorization: AWS
7799e793ce4624ee7e5a:9N/W/kmYwFhbqHYFTvQor317qVw=

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

请求示例 2

修改数据位置为就近。

```
PUT / HTTP/1.1
Host: ctyuntest0929.oos-cn.ctyunapi.cn
Authorization: AWS
0c6c84764e3b8eddf928:oqF7NDGpgL33NRJJXFH60Kyjyhk=
Date: Sun, 29 Sep 2019 07:16:11 GMT
Content-Type: application/xml;charset=utf-8
Content-Length: 170

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

返回示例 2

```

HTTP/1.1 200 OK
Location: /ctyuntest0929
Date: Sun, 29 Sep 2019 07:16:32 GMT
x-amz-request-id: 7088d28e62cc4002
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。

请求语法

```

GET /?location HTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Content-Length: length
Date: date
Authorization: signatureValue

```

响应元素

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

OOS 开发者文档

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：String 有效值： - 父节点为 MetadataLocationConstraint，表示索引位置，有效值为：ChengDu、FuZhou、GuiYang、HangZhou、LaSa、LanZhou、QingDao、ShenYang、ShenZhen、WuHan、WuHu、WuLuMuQi、ZhengZhou - 父节点为 LocationList，表示数据位置，有效值为：ChengDu、GuiYang、LaSa、LanZhou、QingDao、SH2、ShenYang、ShenZhen、SuZhou、WuHan、WuHu、WuLuMuQi、ZhengZhou 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

请求示例

获取一个名叫 picture 的 bucket，索引位置为 Beijing，数据位置为优先本地：

```
GET /?location HTTP/1.1
Host: picture.oos-cn.ctyunapi.cn
Content-Type: application/xml;charset=utf-8
Content-Length:200
Date: Mon, 03 Sep 2012 12:00:00 GMT
Authorization: AWS
7799e793ce4624ee7e5a:9N/W/kmYwFhbqHYFTvQor317qVw=
```

返回示例

```
HTTP/1.1 200 OK
```

```

x-amz-request-id: 236A8905248E5A01
Date: Mon, 03Sep 2012 12:00:00 GMT
Content-Type: application/xml;charset=utf-8
Content-Length: 200
Connection: close
Server: CTYUN

<BucketConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <MetadataLocationConstraint>
    <Location>BeiJing</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

名称	描述
Owner	存储 Bucket 所属账户信息的容器。
ID	Bucket 所属账户的 ID 信息。
DisplayName	Bucket 所属账户的显示名称。
Grant	存储 Permission 和 Grantee 的容器。
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。 有效值： ● READ（只读） ● FULL_CONTROL（公有） ● 空（私有）

请求示例

请求一个指定 bucket 的 ACL 信息

```
GET ?acl HTTP/1.1
Host: bucket.oos-cn.ctyunapi.cn
Date: Mon, 03Sep 2012 22:32:00 GMT
Content-Type: application/xml; charset=utf-8
Authorization: AWS
7799e793ce4624ee7e5a:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

返回示例

```
HTTP/1.1 200 OK
Date: Mon, 03 Sep 2012 22:32:00 GMT
Last-Modified: Sat, 01 Sep 2012 12:00:00 GMT
Content-Type: application/xml; charset=utf-8
x-amz-request-id: b1ee40b578b4632
Content-Length: 380
Server: CTYUN

<?xml version="1.0" encoding="UTF-8"?>
<AccessControlPolicy>
  <Owner>
    <ID>ctyuntest@oos.cn</ID>
    <DisplayName>CustomersName</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>READ</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。

只有根用户和拥有 GET Bucket 权限的子用户才能执行此操作。

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：String	否
marker	指明在 bucket 中 list object 的起始位置，Amazon S3 中 list object 按照阿拉伯字母顺序。	否
max-keys	设置返回结果中最多显示的数量，如果查询结果超过最大数量，另一个变量是否翻页会标记为 true<IsTruncated>True<IsTruncated>,用来返回剩余的查询结果。	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

返回元素

名称	描述
Contents	每个 Object 返回的元数据。
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 结束的左右字符串组合，比如 prefix 是 note/,同时 delimiter 是斜杠 (/)，结果中的 note/summer/july 将返回 note/summer/,其余结果将按照 maxkey 要求返回。

Delimiter	将 prefix 和第一次出现 delimiter 的所有查询结果滚动的存入 CommonPrefix 组中。
ETag	通过 MD5 的方式计算出标签，ETag 主要用来反映对象内容的信息发生了改变，并不反映元数据的变化。
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。
Key	对象的关键字。
LastModified	记录的对象最后一次被修改的日期和时间。
Marker	标记从哪个位置开始罗列出 bucket 中的 object。
Name	Bucket 的名称。
Prefix	关键字以特定的前缀开始。
Size	记录对象 object 的大小。
StorageClass	存储类型： ● STANDARD：标准存储； ● STANDARD_IA：低频访问存储； ● REDUCED_REDUNDANCY：低冗余存储。
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，以便获取后续对象。

请求示例

```
GET / HTTP/1.1
Host: test0304.oos-cn.ctyunapi.cn
x-amz-content-sha256: UNSIGNED-PAYLOAD
Authorization: AWS4-HMAC-SHA256 Credential=58c5d7df73fdbcb78fe92/20200304/cn-180622/s3/aws4_request, SignedHeaders=content-type;host;x-amz-content-sha256;x-amz-date, Signature=aef077644b2ea50fa69f9787ab025be99ae41c9923411ccc2db2a2ff725f0ffb
X-Amz-Date: 20200304T070632Z
Content-Type: application/xml; charset=utf-8
```

返回示例

```
HTTP/1.1 200 OK
Date: Wed, 04 Mar 2020 07:06:40 GMT
x-amz-request-id: b13be3abf4db4af33eaaeb5bdb5b7bd7c7e82727476787a7c
Content-Length: 509
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>test0304</Name>
<Prefix></Prefix>
<Marker></Marker>
<MaxKeys>1000</MaxKeys>
<IsTruncated>>false</IsTruncated>
<Contents>
  <Key>1.png</Key>
  <LastModified>2020-03-04T07:04:48.135Z</LastModified>
  <ETag>&quot;01fa001a4c65b834696c5e0049fd5004&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：

doc/Sample. pdf

doc/Ant/sample. doc

doc/Feb/sample2. doc

doc/Feb/sample3. doc

doc/Feb/sample4. doc

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

```

GET /?prefix=doc/&marker=doc/F&max-keys=40&delimiter=/ HTTP/1.1
Host: doc.oos-cn.ctyunapi.cn
Date: Mon, 03 Sep 2012 12:00:00 GMT
Authorization: AWS 81ebc16ddc8d2af82c90:thdUi9VAkzhkniLj96JlrOPGi0g

```

返回的结果集为：

```

<ListBucketResult xmlns="http://oos-cn.ctyunapi.cn/doc/2012-09-03/">
  <Name>doc</Name>
  <Prefix>doc/</Prefix>
  <Marker>doc/F</Marker>

```

```

<MaxKeys>40</MaxKeys>
<Delimiter>/</Delimiter>
<IsTruncated>>false</IsTruncated>
<Contents>
  <Key>sample.pdf</Key>
  <LastModified>2012-09-01T01:56:20.000Z</LastModified>
  <ETag>&quot;bf1d737a4d46a19f3bcd6905cc8b902&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 doc 的 Bucket:

```

DELETE / HTTP/1.1
Host: doc.oos-cn.ctyunapi.cn
Date: Mon, 03Sep 2012 12:00:00 GMT

```

```
Authorization: AWS
7799e793ce4624ee7e5a:xQE0diMbLRepdf3YB+FIEXAMPLE=
Content-Type: application/xml;charset=utf-8
```

返回示例

```
HTTP/1.1 204 No Content
x-amz-request-id: 32FE2CEB32F5EE25
Date: Mon, 03Sep 2012 12:00:00 GMT
Connection: close
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。

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。

请求示例

```
PUT /?policy HTTP/1.1
Host: doc.oos-cn.ctyunapi.cn
Date: Mon, 03Sep 2012 12:00:00 GMT
Authorization: AWS
7799e793ce4624ee7e5a:xQE0diMbLRepdf3YB+FIEXAMPLE=

{
```

```
"Version": "2012-10-17",
"Id": "aaaa-bbbb-cccc-dddd",
"Statement": [
  {
    "Effect": "Allow",
    "Sid": "1",
    "Principal": {
      "CTYUN": "*"
    },
    "Action": ["oos:*"],
    "Resource": "arn:ctyun:oos::bucket/*",
  }
]
```

返回示例

```
HTTP/1.1 200OK
x-amz-request-id: 32FE2CEB32F5EE25
Date: Mon, 03Sep 2012 12:00:00 GMT
Connection: close
Server: CTYUN
```

4.3.7 GET Bucket Policy

在 GET 操作的 url 中加上 policy，可以获得指定 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: doc.oos-cn.ctyunapi.cn
Date: Mon, 03Sep 2012 12:00:00 GMT
Authorization: AWS
7799e793ce4624ee7e5a:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

返回示例

```
HTTP/1.1 200 OK
x-amz-request-id: 32FE2CEB32F5EE25
Date: Mon, 03Sep 2012 12:00:00 GMT
Connection: close
Server: CTYUN

{
  "Version": "2012-10-17",
  "Id": "aaaa-bbbb-cccc-dddd",
  "Statement": [
    {
      "Effect": "Allow",
      "Sid": "1",
      "Principal": {
        "CTYUN": "*"
      },
      "Action": ["oos:*"],
      "Resource": "arn:ctyun:oos::bucket/*",
    }
  ]
}
```

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: doc.oos-cn.ctyunapi.cn
Date: Mon, 03Sep 2012 12:00:00 GMT
Authorization: AWS
7799e793ce4624ee7e5a:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

返回示例

```
HTTP/1.1 204 No Content
x-amz-request-id: 32FE2CEB32F5EE25
Date: Mon, 03Sep 2012 12:00:00 GMT
Connection: close
Server: CTYUN
```

4.3.9 PUT Bucket WebSite

在 PUT 操作的 url 中加上 website，可以设置 website 配置。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，否则会返回 403 AccessDenied 错误。

WebSite 功能可以让用户将静态网站存放到 OOS 上。对于已经设置了 WebSite 的 Bucket，当用户访问 `http://bucketName.oos-website-cn.oos-cn.ctyunapi.cn` 时，会跳转到用户指定的主页，当出现 4XX 错误时，会跳转到用户指定的出错页面。

如果想通过自有域名的形式（例如 `http://yourdomain.com/login.html`）而非通过第三方域名的形式（例如 `http://yourdomain.com.oos-cn.ctyunapi.cn/login.html`）访问，可以创建一个名为“`yourdomain.com`”的 bucket，并在域名管理系统中将“`yourdomain.com`”增加一个别名记录“`oos-cn.ctyunapi.cn`”。

请求语法

```
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>
</WebsiteConfiguration>
```

请求元素

名称	描述	是否必须
WebsiteConfiguration	请求的容器。	是
IndexDocument	Suffix 元素的容器。	是
Suffix	在请求 <code>website endpoint</code> 上的路径时，Suffix 会被加在请求的后面。例如，如果 suffix 是 <code>Index.html</code> ，而你请求的是 <code>bucket/images/</code> ，那么返回的响应是名为 <code>images/index.html</code> 的 Object。	是
ErrorDocument	Key 的容器。	否
Key	如果出现 4XX 错误，会返回指定的 Object。	否

请求示例

```
PUT /?website HTTP/1.1
Host: doc.oos-cn.ctyunapi.cn
```

```

Date: Mon, 03Sep 2012 12:00:00 GMT
Authorization: AWS
7799e793ce4624ee7e5a:xQE0diMbLRepdf3YB+FIEXAMPLE=

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

```

返回示例

```

HTTP/1.1 200OK
x-amz-request-id: 32FE2CEB32F5EE25
Date: Mon, 03Sep 2012 12:00:00 GMT
Connection: close
Server: CTYUN

```

4.3.10 GET Bucket WebSite

在 GET 操作的 url 中加上 website，可以获得指定 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。

说明：响应体 XML 中所有字符串类型的返回值，都经过转义处理，用户使用前需要反转义，转义表如下：

转义前	转义后
\t		
\n	

\r	
“	"

OOS 开发者文档

&	&
<	<
>	>

请求语法

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

请求示例

```
GET /?website HTTP/1.1
Host: doc.oos-cn.ctyunapi.cn
Date: Mon, 03Sep 2012 12:00:00 GMT
Authorization: AWS 7799e793ce4624ee7e5a:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

返回示例

```
HTTP/1.1 200 OK
x-amz-request-id: 32FE2CEB32F5EE25
Date: Mon, 03Sep 2012 12:00:00 GMT
Connection: close
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>404.html</Key>
  </ErrorDocument>
```

```
</WebsiteConfiguration>
```

4.3.11 DELETE Bucket WebSite

在 DELETE 操作的 url 中加上 website，可以删除指定 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 website，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: doc.oos-cn.ctyunapi.cn
Date: Mon, 03Sep 2012 12:00:00 GMT
Authorization: AWS 7799e793ce4624ee7e5a:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

返回示例

```
HTTP/1.1 200OK
x-amz-request-id: 32FE2CEB32F5EE25
Date: Mon, 03Sep 2012 12:00:00 GMT
Connection: close
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 对象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: Signature
```

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：String	否
max-uploads	设置返回的分片上传过程的最大数目，范围从 1 到 1000，1000 是响应体中能返回的分片上传信息的最大值。 类型：Integer 默认值：1000	否
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	否

	传过程都会被包括，假设这些分片上传过程包含了比 <i>upload-id-marker</i> 指定值更大的 ID。 类型：String	
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）。	否
upload-id-marker	与 <i>key-marker</i> 参数一起，指定列表操作从什么位置后面开始。如果 <i>key-marker</i> 没有被指定， <i>upload-id-marker</i> 参数也会被忽略。否则，任何 key 值等于或大于 <i>key-marker</i> 的上传过程都会被包含在列表中，只要 ID 大于 <i>upload-id-marker</i> 指定的值。	否

返回元素

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, Commonfixes, IsTruncated 父节点：无
Bucket	分片上传对应的容器名称。 类型：String 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：String 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：String 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：String 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：String 父节点：ListMultipartUploadsResult
MaxUploads	响应中包含的上传过程的最大数目。 类型：String 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
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 子节点：Key, UploadId, InitiatorOwner, StorageClass, Initiated 父节点：ListMultipartUploadsResult
Key	分片上传过程起始位置对象的 Key 值。 类型：Integer 父节点：Upload
UploadId	分片上传过程的 ID 号。 类型：Integer 父节点：Upload
Initiator	指定执行此次分片上传过程的用户账户。 子节点：ID, DisplayName 类型：容器 父节点：Upload
ID	OOS 账户的 ID。 类型：String 父节点：Initiator
DisplayName	OOS 账户的账户名。 类型：String 父节点：Initiator
StorageClass	对象的存储类型（STANDARD 或 REDUCED_REDUDANCY） 类型：String 父节点：Upload
Initiated	分片上传过程启动的时间 类型：Date 父节点：Upload

ListMultipartUploadsResult.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 开头的对象。 类型：String 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：String 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 结束的左右字符串组合，比如 prefix 是 note/,同时 delimiter 是斜杠 (/)，结果中的 note/summer/july 将返回 note/summer/,其余结果将按照 maxkey 要求返回。 类型：String 父节点：ListMultipartUploadsResult
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：String 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: AWS 7799e793ce4624ee7e5a.com:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

返回示例

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-movie2.m2ts (NextKeyMarker 的值) 和 *upload-id-marker=*YW55IGlkZWEd2h5IGVsdmluZydzIHVwbG9hZCBmYWlsZWQ (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-movie.m2ts) 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: 656c76696e6727732072657175657374
Date: Mon, 1 Nov 2010 20:34:56 GMT
Content-Length: 1330
Connection: close
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-movie.m2ts</NextKeyMarker>
  <NextUploadIdMarker>YW55IGlkZWEd2h5IGVsdmluZydzIHVwbG9hZCBmYWlsZWQ</NextUploadIdMarker>
  <MaxUploads>3</MaxUploads>
  <IsTruncated>true</IsTruncated>
  <Upload>
    <Key>my-divisor</Key>
    <UploadId>XMgbGlrZSBlbHZpbmcncyBub3QgaGF2aW5nIG11Y2ggbHVjaw</UploadId>
    <Initiator>
      <ID>mailaddress@email.com</ID>
      <DisplayName>user1-11111a31-17b5-4fb7-9df5-b11111f13de</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>VXBsb2FkIElEIGZvcjBlbHZpbmcncyBteS1tb3ZpZS5tMnRzIHVwbG9hZ
A</UploadId>
    <Initiator>
        <ID>mailaddress@email.com</ID>
        <DisplayName>InitiatorDisplayName</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>YW55IGlkZWVmd2h5IGVsdmluZydzIHVwbG9hZCBmYWlsZWQ</Up
loadId>
    <Initiator>
        <ID>mailaddress@email.com</ID>
        <DisplayName>user1-22222a31-17b5-4fb7-9df5-
b22222f13de</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

在 PUT 操作的 url 中加上 logging，可以进行添加/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。	是
LoggingEnabled	日志信息的容器，当启动日志时，需要包含这个元素。	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。	否
TargetPrefix	生成的 log 文件将以此为前缀命名。	否

以下是启动日志的示例：

请求示例

PUT /?logging HTTP/1.1

Host: doc.oos-cn.ctyunapi.cn

```
Content-Type: application/xml; charset=utf-8
Date: Mon, 03Sep 2012 12:00:00 GMT
Authorization: AWS 7799e793ce4624ee7e5a:xQE0diMbLRepdf3YB+FIEXAMPLE=

<?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>
```

返回示例

```
HTTP/1.1 200OK
x-amz-request-id: 32FE2CEB32F5EE25
Date: Mon, 03Sep 2012 12:00:00 GMT
Connection: close
Server: CTYUN
```

以下是取消日志的示例：

请求示例

```
PUT /?logging HTTP/1.1
Host: doc.oos-cn.ctyunapi.cn
Date: Mon, 03Sep 2012 12:00:00 GMT
Authorization: AWS 7799e793ce4624ee7e5a:xQE0diMbLRepdf3YB+FIEXAMPLE=

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

返回示例

```
HTTP/1.1 200OK
x-amz-request-id: 32FE2CEB32F5EE25
Date: Mon, 03Sep 2012 12:00:00 GMT
Connection: close
Server: CTYUN
```

日志名称

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

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。唯一字符串部分没有实际含义, 可以被忽略。

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 Object 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@email.com	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/Aug/2012:22:34:02 +0000]	请求到达的时间, 格式是 [%d/%B/%Y:%H:%M:%S %z]
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@email.com - 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@email.com。 - 如果子用户, 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对象，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
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“-”

4.3.14 GET Bucket Logging

在 GET 操作的 url 中加上 logging，可以获得指定 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
```

响应元素

名称	描述
BucketLoggingStatus	响应的容器
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
TargetBucket	保存 log 的 bucket，OOS 会向此 bucket 存储日志。
TargetPrefix	生成的 log 文件将以此为前缀命名

请求示例

```
GET /?logging HTTP/1.1
Host: doc.oos-cn.ctyunapi.cn
Date: Mon, 03Sep 2012 12:00:00 GMT
Authorization:AWS
7799e793ce4624ee7e5a:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

返回示例

以下是设置了日志的响应示例

```
HTTP/1.1 200OK
x-amz-request-id: 32FE2CEB32F5EE25
Date: Mon, 03Sep 2012 12:00:00 GMT
Connection: close
Server: CTYUN

<?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>
```

以下是没有设置日志时的响应示例

```
HTTP/1.1 200OK
x-amz-request-id: 32FE2CEB32F5EE25
Date: Mon, 03Sep 2012 12:00:00 GMT
Connection: close
Server: CTYUN

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

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: doc.oos-cn.ctyunapi.cn
Date: Mon, 03Sep 2012 12:00:00 GMT
Authorization: AWS 7799e793ce4624ee7e5a:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

响应示例

```
HTTP/1.1 200OK
x-amz-request-id: 32FE2CEB32F5EE25
Date: Mon, 03Sep 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。

生命周期是指对象从更新开始到被删除/转换存储类型的时间。

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对象。当对象的生命周期到期时，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。但对象到期被删除后，用户将不需要为到期的对象付费。

OOS 删除到期对象后，会在 Bucket log 中记录一条日志，操作项是 "OOS.EXPIRE.OBJECT"。

注意：如果对象的生命周期规则设置的是到期后删除，对象到期后将被永久删除，无法恢复。

- 将与生命周期规则匹配的对象由标准存储转换为低频访问存储。OOS 转换存储类型为低频访问存储后，会在 access logs 中记录一条日志，操作项是 "OOS.TRANSITION_SIA.OBJECT"。

注意：

- 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 已存在相同名称的生命周期规则，则执行此操作将覆盖原有规则。
- 每个 Bucket 最多创建 100 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对象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 的到期删除对象的生命周期规则。
- 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对象和后续新创建的对象。例如，用户今天增加了一个生命周期，指定某些前缀的对象 30 天后过期，那么 OOS 将会将满足条件的 30 天前创建的对象都加入到待删除队列中。

OOS 通过将对象的创建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，一个对象于 GMT 2016 年 1 月 15 日 10:30 创建，生命周期为 3 天，那么对象的到期时间是 GMT 2016 年 1 月 19 日 00:00。当重写一个对象时，OOS 将以最后更新时间为准，来重新计算该对象的到期时间。

可以通过 GET Object、HEAD Object 查询对象的到期时间。

请求语法

PUT /?lifecycle HTTP/1.1 Host: <i>BucketName</i> .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>
  </Rule>
</LifecycleConfiguration>

```

请求头

名称	描述	是否必需
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

请求元素

名称	描述	是否必需
LifecycleConfiguration	生命周期规则容器，最多包含 100 个规则。 类型：容器 子节点：Rule 父节点：无	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 父节点：Rule 取值：1~255 个字符	否
Prefix	指定使用生命周期规则的对象前缀。只有匹配对象前缀的对象才能被该规则影响，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 父节点：Rule	是

OOS 开发者文档

	取值：0~1024 个字符	
Status	指定生命周期规则的状态。 类型：字符串 父节点：Rule 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。	是
Expiration	指定生命周期规则的过期时间容器。 类型：容器 子节点：Days 或 Date 父节点：Rule	Expiration 和 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 子节点：Days 或 Date、StorageClass 父节点：Rule	Expiration 和 Transition 至少包含一个
Days	指定生命周期规则在匹配对象创建多少天后生效。 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：正整数 父节点：Expiration 或 Transition 取值：大于 0 的正整数	Days 和 Date 二选一
Date	指定生命周期规则生效日期，OOS 对在此日期之前创建的对象执行生命周期规则。在该生效时间，生成时间早于此时间的对象将被认为是过期对象。 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：String 父节点：Expiration、Transition	Days 和 Date 二选一
StorageClass	指定对象转换的存储类型。 类型：字符串 父节点：Transition 取值： STANDARD_IA：低频访问存储。	Transition 已设置时，必选。

请求示例

示例 1：单一规则

下面的生命周期只包含一个规则，这个规则指定所有以 logs/ 为前缀的对象将在创建后的 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，删除过期的对象。

```
PUT /?lifecycle HTTP/1.1
Host: doc.oos-cn.ctyunapi.cn
Date: Tue, 13 Dec 2011 17:54:50 GMT
Content-MD5: 8dYiLewFWZyGgV2Q5FNI4W==
Authorization: AWS 7799e793ce4624ee7e5a:z6mvnXscCWad60vdmB9xZVVZn46=
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>
```

返回示例

```
HTTP/1.1 200OK
x-amz-request-id: 32FE2CEB32F5EE25
Date: Tue, 13 Dec 2011 19:14:41 GMT
Connection: close
Server: CTYUN
```

示例 2：设置存储类型转换

指定存储对在创建 10 天后转换为低频访问存储，创建 20 天后删除。

请求示例

```
PUT /?lifecycle HTTP/1.1
Host: ctyuntest0625.oos-cn.ctyunapi.cn
Authorization: AWS
AKIAIM6KGTNKZKF4ETLA:fWuioDnZ3Cc1cXPWu8x5MEWglLg=
Date: Mon, 29 Jul 2019 05:08:31 GMT
Content-Type: application/xml
Content-MD5: ouPJihB832ZHNLZNyl2Ccw==
Content-Length: 175

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

返回示例

```
HTTP/1.1 200OK
x-amz-request-id: 32FE2CEB32F5EE25
Date: Mon, 29 Jul 2019 05:10:33 GMT
Connection: close
Server: CTYUN
```

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/ 为前缀的对象将在创建后的 30 天过期。第二个规则指定所有以 documents/ 为前缀的对象将在创建后的 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: doc.oos-cn.ctyunapi.cn
Date: Tue, 13 Dec 2011 17:54:50 GMT
Content-MD5: 8dYiLewFWZyGgV2Q5FNI4W==
Authorization: AWS
7799e793ce4624ee7e5a:z6mvnXscCWad60vdmB9xZVVZn46=
Content-Length: 413

<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>
```

返回示例

```
HTTP/1.1 200OK
x-amz-request-id: 32FE2CEB32F5EE25
Date: Tue, 13 Dec 2011 19:14:41 GMT
Connection: close
Server: CTYUN
```

示例 4：定义所有对象的生命周期规则

下面规则指定所有对象将在创建后的 3650 天被删除。

注意：当指定空的前缀时，规则将对 bucket 内的所有对象生效。OOS 将会删除满足此规则的所有对象。

请求示例

```
PUT /?lifecycle HTTP/1.1
Host: doc.oos-cn.ctyunapi.cn
Date: Tue, 13 Dec 2011 17:54:50 GMT
Content-MD5: 8dYiLewFWZyGgV2Q5FNI4W==
Authorization: AWS
7799e793ce4624ee7e5a:z6mvnXscCWad60vdmB9xZVVZn46=
Content-Length: 226

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

返回示例

```
HTTP/1.1 200OK
x-amz-request-id: BDC4B83DF5096BBE
Date: Tue, 13 Dec 2011 19:14:41 GMT
Connection: close
Server: CTYUN
```

示例 5：避免设置冲突的生命周期规则

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 为前缀的对象在 30 天后过期。而另一个规则指定所有以 documents/2011 为前缀的对象在 365 天过期。在这种情况下，OOS 将返回错误信息。

```
PUT /?lifecycle HTTP/1.1
Host: doc.oos-cn.ctyunapi.cn
Date: Tue, 13 Dec 2011 17:54:50 GMT
Authorization: AWS
7799e793ce4624ee7e5a:z6mvnXscCWad60vdmB9xZVVZn46=
Content-MD5: 8dYiLewFWZyGgV2Q5FNI4W==
Content-Length: 226

<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

```

响应元素

名称	描述
LifecycleConfiguration	生命周期规则容器。 类型：容器 子节点：Rule 父节点：无
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	使用规则的对象前缀。 类型：字符串 父节点：Rule
Status	生命周期规则的状态。 类型：字符串 父节点：Rule

	取值： ● Enabled: 生命周期规则生效； ● Disabled: 生命周期规则不生效。
Expiration	描述过期动作的容器。 类型：容器 子节点：Days 或 Date 父节点：Rule
Transition	生命周期规则的转换存储类型。 类型：容器 子节点：StorageClass、Days 或 Date 父节点：Rule
StorageClass	对象转换的存储类型。 类型：字符串 父节点：Transition 取值：STANDARD_IA: 低频访问存储。
Days	生命周期规则在匹配对象创建多少天后生效。 类型：整数 父节点：Expiration 或 Transition
Date	生命周期规则生效日期，OOS 对在此日期之前创建的对象执行生命周期规则。 类型：String 父节点：Expiration 或 Transition

错误信息

错误码	描述	HTTP 响应码
NoSuchLifecycleConfiguration	The lifecycle configuration does not exist.	404 Not Found

请求示例

从指定 Bucket 中获取生命周期规则的配置信息，OOS 将生命周期的配置返回在响应体中。下面的例子显示所有以 logs 开头的对象将在创建后的 30 天到期。

```
GET /?lifecycle HTTP/1.1
Host: doc.oos-cn.ctyunapi.cn
Date: Tue, 13 Dec 2011 17:54:50 GMT
Authorization: AWS 7799e793ce4624ee7e5a:z6mvnXscCWad60vdmB9xZVVZn46=
```

返回示例

```

HTTP/1.1 200OK
x-amz-request-id: BDC4B83DF5096BBE
Date: Tue, 13 Dec 2011 19:14:41 GMT
Connection: close
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>

```

4.3.18 DELETE Bucket Lifecycle

此接口用于删除配置的 Bucket 生命周期，OOS 将会删除指定 Bucket 的所有生命周期配置规则。用户的对象将永远不会到期，OOS 也不会再自动删除对象。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: doc.oos-cn.ctyunapi.cn
Date: Wed, 14 Dec 2011 05:37:16 GMT
Authorization: AWS 7799e793ce4624ee7e5a:k3nL7gH3+PadhTEVn5EXAMPLE
```

返回示例

```
HTTP/1.1 204No Content
x-amz-request-id: BDC4B83DF5096BBE
Date: Tue, 13 Dec 2011 19:14:41 GMT
Connection: close
Server: CTYUN
```

4.3.19 PUT Bucket accelerate

在 PUT 操作的 url 中加上 accelerate，可以进行添加或修改 CDN IP 白名单的操作。如果 Bucket 已经配置了 CDN 加速，此操作会替换原有配置。只有根用户和 PUT Bucket accelerate 的子用户才能执行此操作，否则会返回 403 AccessDenied 错误。一个 Bucket 最多配置 5 个 IP 白名单地址段。

请求语法

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

<AccelerateConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status>transfer acceleration state</Status>
  <IPWhiteLists>
    <IP>IP1</IP>
    <IP>IP2</IP>
  </IPWhiteLists>
</AccelerateConfiguration>
```

请求元素

名称	描述	是否必须
AccelerateConfiguration	配置 CDN 加速的容器。 类型：容器	是
Status	设置 bucket 的 CDN 加速状态。 类型：枚举 有效值：Enabled Suspended 父节点：AccelerateConfiguration	是
IPWhiteLists	配置 IP 白名单的容器。 类型：容器	否
IP	CDN 服务提供商的 IP 白名单，IP 值需符合 RFC 4632 中描述的 CIDR 表示法。	否

请求示例

启动 bucket 的 CDN 加速功能，设置 IP 白名单列表包含：36.111.88.0/24 和 114.80.1.136。

```
PUT /? accelerate HTTP/1.1
Host: doc.oos-cn.ctyunapi.cn
Date: Mon, 03Sep 2012 12:00:00 GMT
Authorization: AWS fad0e782cd5132563e38:xQE0diMbLRepdf3YB+FIEXAMPLE=

< AccelerateConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status> Enabled</Status>
  <IPWhiteLists>
    <IP>36.111.88.0/24</IP>
    <IP>114.80.1.136</IP>
  </IPWhiteLists>
</AccelerateConfiguration>
```

返回示例

```
HTTP/1.1 200OK
x-amz-request-id: 32FE2CEB32F5EE25
Date: Mon, 03Sep 2012 12:00:00 GMT
Connection: close
Server: CTYUN
```

4.3.20 GET Bucket accelerate

在 GET 操作的 url 中加上 accelerate，可以获得指定 Bucket 的 CDN 配置信息。只有根用户和拥有 GET Bucket accelerate 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 没有配置过 CDN 加速，那么将不会返回状态信息。

请求语法

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

请求示例

```
GET /?accelerate HTTP/1.1
Host: doc.oos-cn.ctyunapi.cn
Date: Mon, 03Sep 2012 12:00:00 GMT
Authorization: AWS fad0e782cd5132563e38:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

返回示例

```
HTTP/1.1 200 OK
x-amz-request-id: 32FE2CEB32F5EE25
Date: Mon, 03Sep 2012 12:00:00 GMT
Connection: close
Server: CTYUN

< AccelerateConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status> Enabled</Status>
  <IPWhiteLists>
    <IP>36.111.88.0/24</IP>
    <IP>114.80.1.136</IP>
  </IPWhiteLists>
</AccelerateConfiguration>
```

如果 bucket 没有配置过 CDN 加速，将返回以下信息：

```
<AccelerateConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
```

4.3.21 PUT Bucket CORS

跨域资源共享 (Cross-Origin Resource Sharing, 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, 是浏览器出于安全考虑而设置的一个限制, 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 CORS, 客户可以构建丰富的客户端 Web 应用程序, 同时可以选择性地允许跨域访问 OOS 资源。

以下是有关使用 CORS 的示例场景:

- 场景 1: 比如用户的网站 `www.example.com`, 后端使用了 OOS。在 web 应用中提供了使用 JavaScript 实现的上传对象功能, 但是在该 web 应用中, 只能向 `www.example.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.example.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.example.com` 中转。
- 场景 2: 假设用户在名为 `website` 的 bucket 中托管网站, 网站的 endpoint 是 `http://website.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 `website.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 64KB。

请求语法

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

```

<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>

```

请求元素

名称	描述	是否必需
CORSConfiguration	最多包含100个CORSRules元素的容器。 类型：容器	是
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	是
ID	规则的唯一标示。最长255个字符。 类型：String 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)	是

	父节点: CORSRule	
AllowedOrigin	允许跨域的源。最多包含一个 * 通配符。比如: <code>http://*.example.com</code>。 用户也可以只指定 * 表示允许所有源跨域访问。 类型: String 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, <code><AllowedHeader>x-amz-*</AllowedHeader></code> 类型: String 父节点: CORSRule	否
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: Integer (秒) 父节点: CORSRule	否
ExposeHeader	指定客户应用程序 (例如, JavaScript XMLHttpRequest对象) 能够访问的响应头。 类型: String 父节点: CORSRule	否

请求示例

第一个规则允许来自 `https://www.example1.com` 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://www.example2.com`。

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: examplebucket.oos-cn.ctyunapi.cn
x-amz-date: Tue, 21 Aug 2012 17:54:50 GMT
Content-MD5: 8dYiLewFWZyGgV2Q5FNI4W==
Authorization: AWS fad0e782cd5132563e38:xQE0diMbLRepdf3YB+FIEXAMPLE=
Content-Length: 445

<CORSConfiguration>
  <CORSRule>
    <AllowedOrigin>http://www.example1.com</AllowedOrigin>

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

    <AllowedHeader>*</AllowedHeader>
  </CORSRule>
  <CORSRule>
    <AllowedOrigin>http://www.example2.com</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: BDC4B83DF5096BBE
```

Date: Mon, 21 Aug 2017 17:54:50 GMT

Server:CTYUN

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。

4.3.22 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: authorization string
```

响应元素

名称	描述
CORSConfiguration	最多包含100个CORSRules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：String 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 (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: <code>http://*.example.com</code>。 用户也可以只指定 * 表示允许所有源跨域访问。 类型: String 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, <code><AllowedHeader>x-amz-*</AllowedHeader></code> 类型: String 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: Integer (秒) 父节点: CORSRule
ExposeHeader	指定客户应用程序 (例如, JavaScript XMLHttpRequest对象) 能够访问的响应头。 类型: String 父节点: CORSRule

请求示例

获取 Bucket 的跨域配置信息。

```
GET /?cors HTTP/1.1
Host: examplebucket.oos-cn.ctyunapi.cn
Date: Wed, 13 Dec 2017 19:14:42 GMT
Authorization:AWS fad0e782cd5132563e38:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

响应示例

```
HTTP/1.1 200 OK
x-amz-request-id: 0CF038E9BCF63097
Date: Wed, 13 Dec 2017 19:14:42 GMT
Server: CTYUN
Content-Length: 280

<CORSConfiguration>
```

```

<CORSRule>
  <AllowedOrigin>http://www.example.com</AllowedOrigin>
  <AllowedMethod>GET</AllowedMethod>
  <MaxAgeSeconds>3000</MaxAgeSec>
  <ExposeHeader>x-amz-server-side-encryption</ExposeHeader>
</CORSRule>
</CORSConfiguration>

```

4.3.23 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: authorization string

```

请求示例

```

DELETE /?cors HTTP/1.1
Host: examplebucket.oos-cn.ctyunapi.cn
Date: Tue, 13 Dec 2011 19:14:42 GMT
Authorization: AWS fad0e782cd5132563e38:xQE0diMbLRepdf3YB+FIEXAMPLE=

```

响应示例

```

HTTP/1.1 204 No Content
x-amz-request-id: 0CF038E9BCF63097
Date: Tue, 13 Dec 2011 19:14:42 GMT
Server: CTYUN

```

4.3.24 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 中所有对象生效。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对象进行修改和删除。

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，对象 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对象；
- 处于合规保留期的对象，无法通过调用 API、控制台修改对象的存储类型，只能通过生命周期修改存储类型。
- 处于合规保留期的对象，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对象过了合规保留期后才能生效。

请求语法

```
PUT /?object-lock HTTP/1.1
Host: bucket-name.oos-cn.ctyunapi.cn
Date: date
Authorization: authorization-string
Content-MD5:md5
Content-Length:length

<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	是否必需
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

请求元素

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 父节点：ObjectLockConfiguration 有效值： ● Enabled：开启合规保留； ● Disabled：不开启合规保留。	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。 类型：容器 父节点：Rule 子节点：Mode、Days 或 Years 二选一	是
Mode	合规保留模式。 类型：枚举 父节点：DefaultRetention 有效值：COMPLIANCE：合规保留。	是
Days	合规保留的天数。 类型：整型 父节点：DefaultRetention 取值：整数形式，1~36500。	条件 Days 和 Years 必须二选一

OOS 开发者文档

	说明： 年与天的换算关系：1 年等于 365 天。	
Years	合规保留的年数。 类型：整型 父节点：DefaultRetention 取值：整数形式，1~100。 说明： 年与天的换算关系：1 年等于 365 天。	条件 Days 和 Years 必 须二选一

请求示例

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

```
<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: 0431A9A8190CF829
Date: Tue, 12 May 2020 03:24:38 GMT
Server: CTYUN
```

4.3.25 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: bucket-name.oos-cn.ctyunapi.cn
Date: date
Authorization: authorization-string
```

响应元素

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。 类型：枚举 父节点：ObjectLockConfiguration 取值： ● Enabled：开启合规保留； ● Disabled：不开启合规保留。
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。 类型：枚举 父节点：DefaultRetention 取值：COMPLIANCE：合规保留。
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: test.oos-cn.ctyunapi.cn
Authorization: authorization
Date: 20200514T020802Z
```

响应示例

```
HTTP/1.1 200 OK
x-amz-request-id: F90E807C4F345033
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>
```

4.3.26 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: bucket-name.oos-cn.ctyunapi.cn
Date: date
Authorization: authorization-string
```

请求示例

```
DELETE /?object-lock HTTP/1.1
Host: test.oos-cn.ctyunapi.cn
```

OOS 开发者文档

Authorization:authorization X-Amz-Date: 20200514T020802Z

响应示例

HTTP/1.1 200 OK x-amz-request-id: F90E807C4F345033 Date: Thu, 14 May 2020 02:08:15 GMT Content-Length:50 Server: CTYUN
--

4.4 关于 Object 的操作

4.4.1 PUT Object

Put 操作用来向指定 bucket 中添加一个对象，要求发送请求者对该 Bucket 有写权限，用户必须添加完整的对象。

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。	否
Content-Disposition	指出对象的描述性的信息。	否
Content-Encoding	指出对象所使用的编码格式。	否
Content-Length	指定对象的大小，单位为字节。	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
Content-Type	标准的 MIME 类型用来描述内容格式。 默认值：application/octet-stream。	否
Expires	对象不再被缓存的时间。 类型：String	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，它将会和对象一起被存储并返回。PUT 请求头大小限制为 8KB。在 PUT 请求头中，用户定义的元数据大小限制为 2KB。	否
x-amz-limit	对象上传限制的速率。 格式为：x-amz-limit:rate=xxx，取值为大于 128 的正整数，单位是 KiB/s。	否
x-amz-storage-class	数据的存储类型。 类型：String 取值：	否

	● STANDARD: 标准存储 ● REDUCED_REDUNDANCY: 低冗余存储 ● STANDARD_IA: 低频访问存储 默认值: STANDARD	
x-ctyun-data-location	设置Bucket的数据位置。 类型: key-value形式。 有效值: 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, 可取值为 ChengDu、GuiYang、LaSa、LanZhou、QingDao、SH2、ShenYang、ShenZhen、SuZhou、WuHan、WuHu、WuLuMuQi、ZhengZhou。 ● 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

请求示例

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

```
PUT /my-image.jpg HTTP/1.1
Host: myBucket.oos-cn.ctyunapi.cn
Date: Mon, 03Sep 2012 17:50:00 GMT
Authorization: AWS 7799e793ce4624ee7e5a:xQE0diMbLRepdf3YB+FIEXAMPLE=
Content-Type: image/ipeg
Content-Length: 11434
[11434 bytes of object data]
```

返回示例

```
HTTP/1.1 100 Continue

HTTP/1.1 200 OK
x-amz-request-id: 0A49CE4060975EAC
Date: Mon, 03 Sep 2012 17:50:00 GMT
ETag: "1b2cf535f27731c974343645a3985328"
Content-Length: 0
Connection: close
Server: CTYUN
```

副本模式

标准存储和低频访问存储使用的动态冗余模式的方式存储数据，默认情况下，OOS 将采用动态冗余模式的方式来存储数据。但对于那些不太重要，可以重复生成的数据，用户可以在请求头中设置 `x-amz-storage-class` 为 `REDUCED_REDUNDANCY`，选择减少冗余策略来降低成本。

4.4.2 GET Object

GET 操作用来检索在 OOS 中的对象信息，执行 GET 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。	否
response-content-language	设置返回头中的 Content-Language。	否
response-cache-control	设置返回头中的 Cache-Control	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.encode(URLEncoder.encode("文件.txt","UTF-8"), " UTF-8")	否
response-content-encoding	设置返回头中的 Content-Encoding	否
response-expires	设置返回头中的 Expires	否
x-amz-limitrate	对象下载限制的速率。取值为大于 0 的正整数，单位是 KiB/s。 注意：x-amz-limitrate 和 x-amz-limit 只能二选一。	否
x-amz-limit	对象下载限制的速率。 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 128 的正整数，单位是 KiB/s； ● concurrency 为并发连接数。取值为大于 0 的正整数。 注意：x-amz-limit 和 x-amz-limitrate 只能二选一。	否

请求头格式

名称	描述	是否必须
Range	下载指定范围内大小的一个对象	否
If-Modified-Since	只返回一个在指定时间点后被修改的对象，否则返回 304 错误	否

If-Unmodified-Since	返回一个在指定时间点后未被修改的对象，否则返回 412 错误	否
If-Match	当对象的 ETag 与指定值一致时，返回此对象。否则返回 412 错误	否
If-None-Match	当对象的 ETag 与指定值不一致时，返回此对象。否则返回 304 错误	否
x-amz-limitrate	对象下载限制的速率。取值为大于 0 的正整数，单位是 KiB/s。 注意：x-amz-limitrate 和 x-amz-limit 只能二选一。	否
x-amz-limit	对象下载限制的速率。 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 128 的正整数，单位是 KiB/s； ● concurrency 为并发连接数。取值为大于 0 的正整数。 注意：x-amz-limit 和 x-amz-limitrate 只能二选一。	否

响应头

变量	描述
x-amz-expiration	如果对象被配置了到期时间，那么 OOS 返回此响应头。这个响应头包含键值对 <i>expiry-date</i> 和 <i>rule-id</i> 。 <i>rule-id</i> 的值是 URL 编码的。
x-ctyun-metadata-location	获取对象的索引位置。 类型：枚举 取值：ChengDu、FuZhou、GuiYang、HangZhou、LaSa、LanZhou、QingDao、ShenYang、ShenZhen、WuHan、WuHu、WuLuMuQi、ZhengZhou
x-ctyun-data-location	获取 Bucket 的数据位置。 类型：枚举 取值：ChengDu、GuiYang、LaSa、LanZhou、QingDao、SH2、ShenYang、ShenZhen、SuZhou、WuHan、WuHu、WuLuMuQi、ZhengZhou
x-amz-storage-class	对象的存储类型，如果存储类型为 STANDARD，则不返回此参数。 类型：String 取值：

- REDUCED_REDUNDANCY: 低冗余存储
- STANDARD_IA: 低频访问存储

请求示例

下面示例中返回对象 my-image.jpg

```
GET /my-image.jpg HTTP/1.1
Host: bucket.oos-cn.ctyunapi.cn
Date: Mon, 03Sep 2012 22:32:00 GMT
Authorization: AWS 7799e793ce4624ee7e5a:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

返回示例

```
HTTP/1.1 200 OK
x-amz-request-id: 318BC8BC148832E5
Date: Mon, 03 Sep 2012 22:32:00 GMT
Last-Modified: Sat, 01 Sep 2012 17:50:00 GMT
ETag: "fba9dede5f27731c9771645a39863328"
Content-Length: 434234
Content-Type: image/ipeg
Connection: close
Server: CTYUN

[434234 bytes of object data]
```

4.4.3 DELETE Object

Delete 操作移除指定的对象，要求用户要对对象所在的 bucket 拥有写权限。

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

下一个例子中删除一个叫 my-image.jpg 的图片对象

```
DELETE /my-image.jpg HTTP/1.1
Host: bucket.oos-cn.ctyunapi.cn
Date: Mon, 03 Sep 2012 17:50:00 GMT
Authorization: AWS 7799e793ce4624ee7e5a:xQE0diMbLRepdf3YB+FIEXAMPLE=
Content-Type: application/xml; charset=utf-8
```

返回示例

```
HTTP/1.1 204 NoContent
x-amz-request-id: 0A49CE4060975EAC
Date: Mon, 03Sep 2012 17:50:00 GMT
Content-Length: 0
Connection: close
Server: CTYUN
```

4.4.4 PUT Object - Copy

此操作用来创建一个存储在 OOS 里的对象拷贝。类似于执行一个 GET，然后再执行一次 PUT。要执行拷贝请求，用户需要对源对象有读权限，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

使用者可以通过以下请求头实现对应操作。

名称	描述	是否必需
x-amz-copy-source	源 Bucket 和对象的名称，用斜杠(/)分割。 类型： 字符串	是
x-amz-metadata-directive	指明元数据是源对象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）外的其他元数据保持不变，拷贝源对象的元数据； 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，源对象和目的对象相同，则必须携带 x-amz-storage-class，否则不能拷贝，返回 400 错误码。	否
x-amz-copy-source-if-match	只有当源对象的 Etag 与给定 Etag 匹配时，才能执行对象拷贝操作，否则返回 412HTTP 状态码错误。 类型： 字符串	否
x-amz-copy-source-if-none-match	只有当源对象的 Etag 与给定 Etag 不匹配时，才能执行对象拷贝操作，否则返回 412HTTP 状态码错误。 类型： 字符串	否
x-amz-copy-source-if-unmodified-since	只有源对象在指定时间点之后没有修改，才执行对象拷贝操作，否则返回 412 错误。	否

	类型：符合 https://tools.ietf.org/html/rfc7232 规定格式的 HTTP 时间字符串。	
x-amz-copy-source-if-modified-since	只有源对象在指定时间点之后修改过，才执行对象拷贝操作，否则返回 412 错误。 类型：符合 https://tools.ietf.org/html/rfc7232 规定格式的 HTTP 时间字符串。	否
x-amz-storage-class	目标对象的存储类型。 类型：String 取值： ● STANDARD：标准存储 ● REDUCED_REDUNDANCY：低冗余存储 ● 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。 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，可取值为 ChengDu、GuiYang、LaSa、LanZhou、QingDao、SH2、ShenYang、ShenZhen、SuZhou、WuHan、WuHu、WuLuMuQi、ZhengZhou。	否

	● 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

返回元素

名称	描述
CopyObjectResult	包含所有返回元素的容器。
ETag	返回新对象的 ETag。ETag 只反映对象内容发生了改变, 元数据未改变。
LastModified	返回对象最后一次修改的日期。

请求示例

将存储桶 **bucket-sample** 中的对象 **my-image.jpg** 复制一份到本存储桶, 新对象命名为 **my-second-image.jpg**。

```
PUT /my-second-image.jpg HTTP/1.1
Host: bucket-sample.oos-cn.ctyunapi.cn
Date: Mon, 03Sep 2012 22:32:00 GMT
x-amz-copy-source: /bucket-sample/my-image.jpg
Authorization: AWS 7799e793ce4624ee7e5a:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

返回示例

```
HTTP/1.1 200 OK
x-amz-request-id: 318BC8BC148832E5
Date: Mon, 03Sep 2012 22:32:00 GMT
Connection: close
Server: CTYUN
```

```
<CopyObjectResult>
  <LastModified>2012-09-01T22:32:00</LastModified>
  <ETag>9b2cf535f27731c974343645a3985328</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对象。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	是否必需
Cache-Control	可以用来指定请求或响应中的缓存操作。 类型：String 默认值：None	否
Content-Disposition	指定对象的描述性信息。 类型：String 默认值：None	否
Content-Encoding	指定对象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：String 默认值：None	否
Content-Type	用来描述对象数据格式的标准 MIME 类型。 类型：String 默认值：application/octet-stream	否

	限制：仅 MIME 类型	
Expires	对象不再被缓存的时间。 类型：String	否
x-amz-meta-	任何以 x-amz-meta-为前缀的头都被当作用户元数据，它和对象一起存储，当用户获取该对象的时候作为响应的一部分被返回。	否
x-amz-storage-class	对象的存储类型，针对那些在成功完成分片上传后被创建的对象。 类型：String 取值： ● STANDARD：标准存储 ● REDUCED_REDUNDANCY：低冗余存储 ● 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。 类型：key-value形式。 有效值： 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，可取值为 ChengDu、GuiYang、LaSa、LanZhou、QingDao、SH2、ShenYang、ShenZhen、SuZhou、WuHan、WuHu、WuLuMuQi、ZhengZhou。 ● 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。	
--	---	--

返回元素

名称	描述
InitiateMultipartUploadResult	包含所有返回元素的容器。 类型：容器 子节点：Bucket, Key, UploadId 父节点：无
Bucket	分片上传对应的 Bucket 的名称。 类型：String 父节点：InitiateMultipartUploadResult
Key	分片上传对应的对象名称。 类型：String 父节点：InitiateMultipartUploadResult
UploadId	分片上传 ID。 类型：String 父节点：InitiateMultipartUploadResult

请求示例

示例中执行的操作是初始化一个名为“example-object”对象的分片上传操作。

```
POST /example-object?uploads HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
Date: Mon, 1 Nov 2010 20:34:56 GMT
Authorization: AWS 7799e793ce4624ee7e5a:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

返回示例

```
HTTP/1.1 200 OK
x-amz-request-id: 656c76696e6727732072657175657374
Date: Mon, 1 Nov 2010 20:34:56 GMT
Content-Length: 197
Connection: close
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>VXBsb2FkIElEIGZvciA2aWWpbmcncyBteS1tb3ZpZS5tMnRzIHVwbG
9hZA</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对象中的位置，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
```

请求参数

无

请求头格式

使用者可以通过一下请求头实现对应操作。

名称	描述	是否必须
Content-Length	该分片的大小，单位为字节。 类型：Integer 默认值：None	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：String 默认值：None	否
x-amz-limit	对象上传限制的速率。 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 128 的正整数，单位是 KiB/s； ● 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：String 默认值：None 有效值：100-continue	否

返回元素

无

出错响应

响应代码	描述	HTTP 状态码
NoSuchUpload	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。	404 Not Found

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=VCVsb2FkIElEIGZvciBlbZZpbm
cncyBteS1tb3ZpZS5tMnRzIHVwbG9hZR 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: AWS 7799e793ce4624ee7e5a:xQE0diMbLLRepdf3YB+FIEXAMPLE=

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

返回示例

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: 656c76696e6727732072657175657374
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对象。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: Signature

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

请求参数

无

请求头格式

无

请求元素

名称	描述	是否必须
CompleteMultipartUpload	请求的容器。 父节点：无 类型：容器 子节点：1 个或多个 Part 元素	是
Part	一个片段的容器。	是

	父节点: CompleteMultipartUpload 类型: 容器 子节点: PartNumber, ETag	
PartNumber	标识片段的分片号。 父节点: Part 类型: Integer	是
ETag	片段上传完成时返回的 Etag 内容。 父节点: Part 类型: String	是

返回元素

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 父节点: 无
Location	新创建对象的 URL 地址。 类型: URI 父节点: CompleteMultipartUploadResult
Bucket	分片上传对应的对象容器。 类型: String 父节点: CompleteMultipartUploadResult
Key	新创建的对象 Key。 类型: String 父节点: CompleteMultipartUploadResult
ETag	ETag 用来标识新创建的对象数据。 类型: String 父节点: CompleteMultipartUploadResult

出错响应

响应代码	描述	HTTP 状态码
InvalidPart	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 Bad Request
InvalidPartOrder	片段列表不以升序排列, 片段列表必须根据分片号按顺序排列	400 Bad Request
NoSuchUpload	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	404 Not Found
InvalidPartSize	如果小于 5M 的分片为 2 片或者 2 片以上, 会终止合并分片上传。	400 Bad Request

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=AAAsb2FkIElEIGZvcjBlbHZpbmcncyWeeS1tb3ZpZS5tMnRzIR
RwbG9hZA HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
Date: Mon, 1 Nov 2010 20:34:56 GMT
Content-Length: 391
Authorization: AWS 7799e793ce4624ee7e5a:xQE0diMbLRepdf3YB+FIEXAMPLE=

<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>
```

返回示例

下面的响应中表示一个对象被拼接成功。

```
HTTP/1.1 200 OK
x-amz-id-2: Uuag1LuByRx9e6j5Onimru9pO4ZVKnJ2Qz7/C1NPcfTWAAtRPfTaOFg==
x-amz-request-id: 656c76696e6727732072657175657374
Date: Mon, 1 Nov 2010 20:34:56 GMT
Connection: close
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: Signature
```

出错响应

响应代码	描述	HTTP 状态码
NoSuchUpload	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	404 Not Found

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=VXBsb2FkIElEIGZvciBlbHZpbmcncyBteS1tb3ZpZS5tMnRzIHVwbG9hZ HTTP/1.1
Host: example-bucket.oos-cn.ctyunapi.cn
Date: Mon, 1 Nov 2010 20:34:56 GMT
Authorization: AWS 7799e793ce4624ee7e5a:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

返回示例

```

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

```

4.4.9 List Part

该操作用于列出一个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: Signature

```

请求参数

此次 Get 请求通过使用下表中的参数来返回一个 `uploadId` 中的 `parts` 集合。

名称	描述	是否必须
<code>uploadId</code>	该 ID 用来标识一个分片上传过程。 类型：String 默认值：None	是
<code>max-parts</code>	设置响应体中返回的片段的最大数目。 类型：String 默认值：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：String 默认值：None	否
--------------------	--	---

返回元素

名称	描述
ListPartsResult	包含整个响应的容器。 类型：容器 子节点：Bucket, Key, UploadId, Initiator, Owner, StorageClass, PartNumberMarker, NextPartNumberMarker, MaxParts, IsTruncated, Part 父节点：无
Bucket	分片上传对应的 bucket 名称。 类型：String 父节点：ListPartsResult
Key	新创建的对象的关键字。 类型：String 父节点：ListPartsResult
UploadId	分片上传 ID。 类型：String 父节点：ListPartsResult
Initiator	指定执行此次分片上传过程的用户账户。 子节点：ID, DisplayName 类型：容器 父节点：ListPartsResult
ID	OOS 账户的 ID。 类型：String 父节点：Initiator
DisplayName	OOS 账户的账户名。 类型：String 父节点：Initiator
StorageClass	对象的存储类型（STANDARD 或 REDUCED_REDUNDANCY）。 类型：String 父节点：ListPartsResult
PartNumberMarker	列表起始位置的片段的分片号。 类型：Integer 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 <i>part-number-marker</i> 参数的值。 类型：Integer

	父节点: ListPartsResult
MaxParts	响应中片段的最大数目。 类型: Integer 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。 子节点: PartNumber, LastModified, ETag, Size 类型: String 父节点: ListPartsResult
PartNumber	标识片段的分片号。 类型: Integer 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: String 父节点: Part
Size	片段的数据大小 类型: Integer 父节点: Part

出错响应

响应代码	描述	HTTP 状态码
InvalidArgument	max-parts 范围必须为 0~1000。	400 Bad Request
NoSuchUpload	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。	404 Not Found

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=XXBsb2FkIElEIGZvciBlbHZpbmcncyVcdS1tb3ZpZS5tMnRzEEEw
bG9hZA&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: AWS 7799e793ce4624ee7e5a:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

返回示例

```
HTTP/1.1 200 OK
x-amz-request-id: 656c76696e6727732072657175657374
Date: Mon, 1 Nov 2010 20:34:56 GMT
Content-Length: 985
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>
  <Key>example-object</Key>
  <UploadId>XXBsb2FkIElEIGZvciBlbHZpbmcncyVcdS1tb3ZpZS5tMnRzEEEwbG9hZ
A</UploadId>
  <Initiator>
    <ID>mailaddress@email.com</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 为新对象分片号，UploadId 为新对象的分片上传 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: Signature

```

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。	是

	类型: String 默认值: None	
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。 类型: String 默认值: None	否
x-amz-copy-source-if-match	● 如果对象的实体 Etag 与给定 Etag 匹配, 执行拷贝对象的操作; ● 如果对象的实体 Etag 与给定 Etag 不匹配, 请求返回 412HTTP 状态码错误。	否
x-amz-copy-source-if-none-match	● 如果对象实体 Etag 和指定实体 Etag 不同, 执行拷贝操作; ● 如果对象实体 Etag 和指定实体 Etag 相同, 返回 412 错误。	否
x-amz-copy-source-if-unmodified-since	● 如果对象在指定时间点之后没有修改过, 执行拷贝操作; ● 如果对象在指定时间点之后修改过, 否则返回 412 错误。	否
x-amz-copy-source-if-modified-since	● 如果对象在指定时间点之后被修改过, 执行拷贝操作; ● 如果对象在指定时间点之后没有被修改过, 返回 412 错误。	否

返回元素

名称	描述
CopyPartResult	包含整个响应的容器。 类型: 容器 父节点: 无
ETag	新分片的 ETag。 类型: String 父节点: CopyPartResult
LastModified	分片的最后修改时间。 类型: String 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=VCVsb2FkIElEIGZvciBlbZZpbm
cncyBteS1tb3ZpZS5tMnRzIHVwbG9hZR 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: AWS 7799e793ce4624ee7e5a:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

返回示例

```
HTTP/1.1 200 OK
x-amz-request-id: 996c76696e6727732072657175657374
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 Object

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。

说明：在删除不存在的对象时，也会返回删除成功。

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 /?deleteHTTP/1.1
Host: BucketName.oos-cn.ctyunapi.cn
Content-Length: Size
Content-MD5: MD5
Authorization: Signature
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
<i>Content-MD5</i>	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：String 默认值：无	是
<i>Content-Length</i>	请求体的长度，参考 RFC2616。 类型：String 默认值：无	是

请求元素

名称	描述	是否必须
<i>Delete</i>	包含整个请求的容器。 类型：容器	是

	父节点：无	
<i>Quiet</i>	使用简明信息模式来返回响应，有效值为： <code>true</code> 和 <code>false</code> ，默认为 <code>false</code> 。 类型： <code>Boolean</code> 父节点： <i>Delete</i>	否
<i>Object</i>	包含被删除 <code>object</code> 的容器。 类型：容器 父节点： <i>Delete</i>	是
<i>Key</i>	被删除 <code>object</code> 名。 类型： <code>String</code> 父节点： <i>Object</i>	是

响应元素

名称	描述
<i>DeleteResult</i>	包含整个响应的容器。 类型：容器 父节点：无
<i>Deleted</i>	成功删除的容器，包含成功删除的 <code>object</code> 。 类型：容器 父节点： <i>DeleteResult</i>
<i>Key</i>	尝试删除的 <code>object</code> 名。 类型： <code>String</code> 父节点： <i>Deleted</i> ，或 <i>Error</i>
<i>Error</i>	删除失败的容器，包含删除失败的 <code>object</code> 信息。 类型：容器 父节点： <i>DeleteResult</i>
<i>Code</i>	删除失败的状态码。 类型： <code>String</code> 父节点： <i>Error</i> 值： <code>AccessDenied</code> , <code>InternalError</code>
<i>Message</i>	错误的描述。 类型： <code>String</code> 父节点： <i>Error</i>

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.ciyunapi.cn
Date: Mon, 11 Apr 2011 20:34:56 GMT
Content-MD5: p5/WA/oEr30qrEEL21PAqw==
Authorization: AWS 7799e793ce4624ee7e5a:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

```
Content-Length: 125
Connection: Keep-Alive
```

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

返回示例

```
HTTP/1.1 200 OK
x-amz-request-id: 996c76696e6727732072657175657374
Date: Mon, 1 Nov 2010 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
<i>EnableCheckpoint</i>	是否开启断点续传功能。 默认：关闭
<i>PartSize</i>	每个分段的大小 <i>partSize</i> ，若 <i>partSize</i> 小于 5MB，则会将 <i>partSize</i> 调整至 5MB。 默认：5MB
<i>UploadFile</i>	上传的本地文件。
<i>checkpointFile</i>	记录本地分片上传结果的文件，默认为上传的本地文件同路径下 <i>uploadFile.ucp</i> 。
<i>objectMetadata</i>	文件的元数据。
<i>ProgressListener</i>	上传状态监听器。
<i>TaskNum</i>	分片上传并发线程数，默认为 1，最多为 1000。

用 *checkpoint* 文件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。*checkpoint* 文件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 = "bkt-hhhhhtest";
        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 5MB，若 partSize 小于 5MB，除了最后一个分片以外，会将 partSize 调整至 5MB */
        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 表单将对象上传到指定的 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，将对象上传到指

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 不存储部分对象：如果收到成功的响应，那么对象就是存储成功了。

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对象的 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，用户可以在上传对象到 OOS 时计算对象的 MD5 值，并与 OOS 在响应中返回的 ETag 进行比较。ETag 是对象内容的 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、签名、文件、策略和带 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	是否必需
<i>AWSAccessKeyId</i>	根用户或拥有权限的子用户的访问密钥 ID，该用户将授予匿名用户对满足策略中一组约束的请求的访问权限。如果请求包含 Policy ，对于 V2 签名，则此字段为必填字段。 类型：String 默认值：无	条件
<i>Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires</i>	特定于 REST 的请求头。有关更多信息，请参阅 PUT Object。 类型：String 默认值：无	否
<i>file</i>	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 默认值：无	是
<i>key</i>	上传对象的名称。 可以使用 <code>\${filename}</code> 变量来使用用户提供的文件名。例如，如果用户 Betty 上传的文件名为 lolcatz.jpg，字段值指定为 <code>/user/betty/\${filename}</code> ，那么保存的对象名称将会是 <code>/user/betty/lolcatz.jpg</code> 。 类型：String 默认值：无	是
<i>policy</i>	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。 Policy 的设置参见 Bucket Policy 字段的构造。 类型：String 默认值：无	条件
<i>signature</i>	使用 AWSAccessKeyId 对应的密钥对 policy 的签名。如果请求包含 Policy，对于 V2 签名，则此字段为必填字段。 <code>signature = Base64(HMAC-SHA1(YourSecretKey,policy))</code> ，具体计算方法请查看 Signature 字段的构造步骤。	条件
<i>X-Amz-Algorithm</i>	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。 默认值：无	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： <pre><your-access-key-id>/<date>/<region>/<service>/aws4_request</pre> 其中： ● date 格式为 YYYYMMDD。 ● region: 访问域名，对象存储域名详见 Endpoint 列表； ● service: 取值 s3。 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。	条件
X-Amz-Signature	认证签名，此签名必须与 OOS 计算的签名相匹配；否则，OOS 拒绝该请求。 如果请求包含 Policy，对于 V4 签名，则此字段为必填字段。	条件
success_action_redirect,redirect	上传成功后客户端重定向到的 URL。OOS 将 Bucket、对象名和 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：String 默认值：无 注意：redirect 将在可能会移除，建议使用 success_action_redirect	否

<i>success_action_status</i>	如果没有指定 <code>success_action_redirect</code>，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。 ● 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 对象。 ● 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 <code>success_action_status</code> 设置为 201。	否
<i>x-amz-storage-class</i>	数据的存储类型。 类型: String 取值: ● STANDARD: 标准存储 ● REDUCED_REDUNDANCY: 低冗余存储 ● STANDARD_IA: 低频访问存储 默认值为 STANDARD。	否
<i>x-amz-meta-*</i>	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，它将会和对象一起被存储并返回。更多信息请参考 PUT Object 类型: String 默认值: 无	否
<i>x-amz-website-redirect-location</i>	如果 Bucket 配置为网站，重定向对这个对象的请求到相同 Bucket 的另外一个对象或者到一个其他的 URL。OOS 会保存这个值到对象的 metadata。对象的 metadata 信息请参考 Object Key 和 Metadata 部分。 下面这个例子表示请求头设置重定向到相同 Bucket 的另一个对象(anotherPage.html)。 <code>x-amz-website-redirect-location:</code> /anotherPage.html 重定向到其他网站的例子: <code>x-amz-website-redirect-location:</code> http://www.example.com/ 注意: 这个值必须以 "/"、http://或 https://开头，且长度不能超过 2KB。	否

x-ctyun-data-location	设置Bucket的数据位置。 类型：key-value形式。 有效值： 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，可取值为 ChengDu、GuiYang、LaSa、LanZhou、QingDao、SH2、ShenYang、ShenZhen、SuZhou、WuHan、WuHu、WuLuMuQi、ZhengZhou。 ● 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
-----------------------	--	---

响应格式

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	如果对象设置了过期时间（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：String
success_action_redirect, redirect	上传成功重定向的 URL。 类型：String

响应体

名称	描述
<i>Bucket</i>	存储 Object 的 Bucket 名称。 类型：字符串 父元素：PostResponse
<i>ETag</i>	ETag 是对象内容经过 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 仅反映对象内容的变化，不包括元数据。 类型：String 父元素：PostResponse
<i>Key</i>	对象名称。 类型：String 父元素：PostResponse
<i>Location</i>	对象的 URL。 类型：String 父元素：PostResponse

请求示例

```
POST / HTTP/1.1
Content-Length: 4
Host: BucketName.oos-cn.ctyunapi.cn
Date: Wed, 01 Mar 2009 12:00:00 GMT
Content-Type: text/plain
Expect: the 100-continue HTTP status code
ObjectContent
```

响应示例

下面的代码演示了一个简单的响应头。

```
HTTP/1.1 200 OK
x-amz-request-id: 0A49CE4060975EAC
Date: Wed, 12 Oct 2009 17:50:00 GMT
ETag: "1b2cf535f27731c974343645a3985328"
Content-Length: 0
Connection: close
Server: OOS
```

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

Post Policy 是使用 UTF-8 和 Base64 编码的 JSON 文档，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。根据您的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

```
{
  "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {
      "bucket": "johnsmith"
    },
    [
      "starts-with",
      "$key",
      "user/eric/"
    ]
  ]
}
```

策略文档由过期(expiration)和条件(conditions)两部分构成。

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对象的内容。您在表单中指定的每个表单字段（AWSAccessKeyId、签名、文件、策略和带 x-ignore-前缀的字段名称除外）必须包含在条件列表中。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，如果您将对象名称字段设置为 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
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的最小和最大大小。 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	已上传的密钥的名称。 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。
其他以 x-amz-meta-为前缀的字段名称	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 <code>bucket</code> 必须设置为 <code>BucketName</code> ： <pre>{"bucket": "BucketName"}</pre> 也可写为： <pre>["eq", "bucket": "BucketName"]</pre>
Starts With	如果值必须从某个特定的值开始，请使用 <code>starts-with</code> 。本示例指示密钥必须从 <code>user/betty</code> 开始： <pre>["starts-with", "\$key", "user/betty/"]</pre>
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 <code>starts-with</code> 和一个空值。本示例允许任何 <code>success_action_redirect</code> ： <pre>["starts-with", "\$success_action_redirect", ""]</pre>
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 MB 的文件大小： <pre>["content-length-range", 1048579, 10485760]</pre>

字符串转义

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

Signature 字段的构造步骤

步骤	说明
1	使用 UTF-8 对策略进行编码。

2	使用 Base64 对这些 UTF-8 字节进行编码。
3	使用 HMAC SHA-1，通过您的秘密访问密钥签署策略。
4	使用 Base64 对 SHA-1 签名进行编码。

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
```

请求头格式

名称	描述	是否必需
Origin	标示访问OOS的跨域请求的源。 例如： http://www.example.com 。 类型: String	是
Access-Control-Request-Method	标示在实际请求中，将被用到的 HTTP 方法。 类型: String	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: String	否

响应头格式

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: String
Access-Control-Max-Age	预检请求可以被缓存的时间，单位是秒。 类型: String
Access-Control-Allow-Methods	用户请求中发送的 HTTP 方法。如果此方法不被允许，那么 OOS 不会返回此响应头。

	类型: String
Access-Control-Allow-Headers	在实际请求中, 浏览器可以发送的 HTTP 请求头列表, 以逗号分隔。如果没有任何请求头被允许, OOS 不会返回此响应头, 也不会返回任何以 Access-Control 开头的响应头。 类型: String
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: String

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

http://www.example.com 向名为 examplebucket 的 bucket, 发送 PUT 请求。

```
OPTIONS /exampleobject HTTP/1.1
Host: examplebucket.oos-cn.ctyunapi.cn
Origin: http://www.example.com
Access-Control-Request-Method: PUT
```

响应示例

```
HTTP/1.1 200 OK
x-amz-request-id: BDC4B83DF5096BBE
Date: Wed, 21 Aug 2012 23:09:55 GMT
Access-Control-Allow-Origin: http://www.example.com
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

对于私有或只读 Bucket, 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(String bucketName, String key, Date expiration) 方法, 可以生成共享链接 URL。参数分别是 Bucket 名称, Object 名称和过期时间, 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 void test_generatePresignedUrl_limit() throws Exception{
    oosClient.setEndpoint(OOS_DOMAIN);
    GeneratePresignedUrlRequest GRrequest =
        new GeneratePresignedUrlRequest(Sbucket, Subject, HttpMethod.GET);
    GRrequest.addRequestParameter("x-amz-limitrate", "2048");
    System.out.println(oosClient.generatePresignedUrl(GRrequest));
}
```

以下是一个生成的共享链接：

```
http://oos-cn.ctyunapi.cn/mysite/a.png?Expires=1363760719&AWSAccessKeyId=
mailaddress %40email.cn&Signature=S7FTz%2BerLjBjirI7jKI4LmHxDRA%3D
```

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 KB/s)，将参数加到 `generatePresignedUrlRequest` 对象中，参与共享链接生成，以下为增加了下载速度限制的生成的共享链接的示例：

```
http://oos-cn.ctyunapi.cn/test-
20180604/6aa3df83gw1f35nhhp70pj20gj0r046l.jpg?Signature=8l7F/pabWm2%2Bi8iXyE
xZIXm/eGY%3D&AWSAccessKeyId=08f17977afa1a87736ac&Expires=1528438576&x-
amz-limitrate=2048
```

4.4.16 HEAD Object

Head 操作用于获取对象的元数据信息，而不返回数据本身。当只希望获取对象的属性信息时，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	如果对象被配置了到期时间，那么 OOS 返回此响应头。这个响应头包含键值对 <i>expiry-date</i> 和 <i>rule-id</i> 。 <i>rule-id</i> 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：String 取值： ● REDUCED_REDUNDANCY：低冗余存储 ● STANDARD_IA：低频访问存储

请求示例

下面示例中返回对象 my-image.jpg

```
HEAD /my-image.jpg HTTP/1.1
Host: bucket.oos-cn.ctyunapi.cn
Date: Mon, 03Sep 2012 22:32:00 GMT
Authorization: AWS 7799e793ce4624ee7e5a:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

返回示例

```
HTTP/1.1 200 OK
x-amz-request-id: 318BC8BC148832E5
Date: Mon, 03 Sep 2012 22:32:00 GMT
Last-Modified: Sat, 01 Sep 2012 17:50:00 GMT
ETag: "fba9dede5f27731c9771645a39863328"
Content-Length: 434234
Content-Type: image/ipeg
Connection: close
Server: CTYUN
```

4.5 关于 AccessKey 的操作

4.5.1 CreateAccessKey

详细参见 7.3.9 CreateAccesskey。

4.5.2 DeleteAccessKey

详细参见 7.3.12 DeleteAccessKey。

4.5.3 UpdateAccessKey

详细参见 7.3.11 UpdateAccessKey。7.3.10

4.5.4 ListAccessKey

详细参见 7.3.10 ListAccessKeys。

4.5.5 STS 临时授权访问

OOS 为用户提供临时授权访问。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。

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 中或者以请求参数的形式放入 URI 中，标头为 “X-Amz-Security-Token”。

请求语法

```
POST/?Action=GetSessionToken&DurationSeconds=seconds HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
Date: GMT Date
Authorization: SignatureValue
```

请求参数

名称	描述	是否必需
Action	值是GetSessionToken。	是

OOS 开发者文档

DurationSeconds	令牌的有效期限，单位秒，范围 15 分钟至 36 小时，若用户没有设置有效期限，报错 400 InvalidDurationSeconds。	是
-----------------	--	---

响应结果

响应头	描述
Credentials	用户访问凭证信息
SessionToken	安全令牌
AccessKeyId	临时访问 AK
SecretAccessKey	临时访问 SK
Expiration	过期时间
RequestId	请求ID

请求示例

```
POST /?Action=GetSessionToken&DurationSeconds=7200 HTTP/1.1
Host:oos-cn-iam.ctyunapi.cn
Date: Wed, 01 Mar 2012 12:00:00 GMT
Authorization: AWS 7799e793ce4624ee7e5a:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

响应示例

```
<GetSessionTokenResponse>
  <GetSessionTokenResult>
    <Credentials>
      <SessionToken>53732f2313e795ba5a649c40d64441b618f194d7c0c08
        33c337dbc954b113eb37b66abea6751b890320fc5f7253482e87a0ca50f7bcb
        221fdc9f4620e232721e
      </SessionToken>
      <AccessKeyId>sts.1310458e7d0d73109ba7</AccessKeyId>
      <SecretAccessKey>9af52496afb6c0f5ba6542
        ec1bec4e52c19093e2
      </SecretAccessKey>
      <Expiration>2018-10-29 17:38:21</Expiration>
    </Credentials>
  </GetSessionTokenResult>
  <ResponseMetadata>
    <RequestId>cb4cbf7cdd3f4572</RequestId>
  </ResponseMetadata>
</GetSessionTokenResponse>
```

4.6 Backoff 说明

当 OOS 系统服务繁忙，暂时不可使用时，OOS 会返回客户端 503 响应状态码和 Retry-After 响应头。示例如下：

```
HTTP/1.1 503 Service Unavailable
x-ctyun-request-id: abdcf4945d14406a-30383a4b4e4948542d6b6e696854
Retry-After: 90
Date: Tue, 6 May 2014 08:02:58 GMT
Content-Length: 133
Content-Type: text/xml
Connection: close
Server: CTYUN

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

4.7 错误响应

4.7.1 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>
```

响应元素说明如下。

名称	描述
Code	错误码是描述出错信息的字符串。详细的错误码信息可以参见后面的列表。
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。

4.7.2 错误码列表

错误码	描述	HTTP 状态码
AccessDenied	权限被拒绝。	403 Forbidden
AuthorizationHeaderMalformed	签名请求头不合法。	400 Bad Request
AuthorizationQueryParametersError	签名请求参数不合法。	400 Bad Request
BadDigest	用户指定的 Content-MD5 与 OOS 计算的不匹配。	400 Bad Request
BucketAlreadyExists	请求的 bucket name 已经存在，请选择一个不同的名字。	409 Conflict

OOS 开发者文档

BucketNotEmpty	试图删除的 bucket 非空。	409 Conflict
CanNotDeleteObjectLockConfiguration	合规保留已开启，不能进行删除。	400 Bad Request
CanNotModifyMetadataLocation	索引位置不允许修改。	400 Bad Request
FileImmutable	对象处于合规保留期，不能修改。	403 Forbidden
IncompleteBody	对象长度小于 Content-Length 请求头中指定的长度。	400 Bad Request
Internal Error	OOS 服务错误，请重试。	500 InternalServerError
InvalidAccessKeyId	AccessKeyId 不存在。	403 Forbidden
InvalidArgument	无效的请求参数。	400 Bad Request
InvalidArgument	max-parts 范围必须为 0~1000。	400 Bad Request
InvalidArgument	合规保留时长不能小于最近一次设置的时长。	400 Bad Request
InvalidArgument	合规保留时长超限。	400 Bad Request
InvalidArgument	合规保留时长必须为正整数。	400 Bad Request
InvalidArgument	合规保留功能已经启动，不能停止。	400 Bad Request
InvalidArgument	设置生命周期规则时，“Date” 必须在格林尼治标准时间午夜。	400 Bad Request

InvalidArgument	设置生命周期时长的参数“Days”必须为正整数。	400 Bad Request
InvalidArgument	设置生命周期规则时，需要指定 Expiration 和 Transition 中的一项。	400 Bad Request
InvalidArgument	设置生命周期规则时，如果 prefix 有重叠的两种生命周期规则 Expiration 和 Transition，Expiration 设置 Days 必须大于 Transition 设置的 Days。	400 Bad Request
InvalidArgument	设置生命周期规则时，如果 prefix 有重叠的两种生命周期规则 Expiration 和 Transition，Expiration 设置 Date 必须晚于 Transition 设置的 Date。	400 Bad Request
InvalidArgument	设置生命周期规则的转换存储类型不同。	400 Bad Request
InvalidBucketName	指定的 bucket 无效。	400 Bad Request
InvalidLocationConstraint	用户提供的配置索引位置和数据位置的 xml 格式不合法。	400 Bad Request
InvalidObjectName	对象名称不合法。	400 Bad Request
InvalidPolicyDocument	策略不合法。	400 Bad Request
InvalidRequest	IP 白名单不合法。	400 Bad Request
InvalidRequest	IP 白名单个数超过限制。	400 Bad Request

InvalidRequest	未发送 Content-MD5 请求头。	400 Bad Request
InvalidRequest	设置生命周期规则时，如果两种生命周期规则 Expiration 和 Transition 设置的前缀有重叠的，则只能同时使用 Date，或者同时使用 Days。	400 Bad Request
InvalidStorageClass	副本模式不合法。	400 Bad Request
InvalidURI	无法解析指定的 URI。	400 Bad Request
MalformedXML	XML 格式不合法。	400 Bad Request
MethodNotAllowed	指定的方法不允许操作该资源。	405 Method Not Allowed
MissingContentLength	在 HTTP 头中必须提供 ContentLength。	411 length required
NoSuchBucket	指定的 Bucket 不存在。	404 Not Found
NoSuchKey	指定对象的 Key 不存在。	404 Not Found
NoSuchLifecycleConfiguration	生命周期的配置不存在。	404 Not Found
NoSuchUpload	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。	404 Not Found
NotVerify	用户未激活。	403 Forbidden
ObjectLockConfigurationNotFound Error	Bucket 没有配置合规保留。	404 Not Found
PreconditionFailed	至少有一个指定的条件不满足。	412 Precondition Failed

OOS 开发者文档

PutObjectTooFast	并发写同一个对象或分段片段时冲突。	400 Bad Request
RequestTimeTooSkewed	请求的时间和服务器时间相差太大。	403 Bad Request
SignatureDoesNotMatch	我们计算出的请求中的签名与用户提供的签名不一致。请检查 <code>secretKey</code> 和签名算法。	403 Forbidden
SlowDown	OOS 服务繁忙，请重试。	503 Service Unavailable
TooManyBuckets	Bucket 个数超出限制。	400 Bad Request
--	客户端请求错误。	400 Bad Request
--	客户端请求超时。	408 Request Timeout
--	对象长度小于 Content-Length 请求头中指定的长度。	502 Bad Gateway
--	OOS 服务繁忙，请重试。	503 Service Unavailable
--	OOS 服务繁忙，请重试。	504 Gateway Timeout

说明：“--”表示没有返回错误码，只有状态码。

4.8 使用 HttpURLConnection 开发

```
package cn.ctyun.oos.sample;

import java.io.InputStream;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;
import java.util.Random;
import java.util.TimeZone;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import org.apache.commons.codec.binary.Base64;

public class HttpURLConnectionSample {
    private static final int CONN_TIMEOUT = 10000;
    private static final int READ_TIMEOUT = 30000;
    private static final String DATE_STR = "EEE, d MMM yyyy HH:mm:ss 'GMT'";
    private static final SimpleDateFormat DATE_FMT = new
SimpleDateFormat(DATE_STR, Locale.ENGLISH);
    static {
        TimeZone gmt = TimeZone.getTimeZone("GMT");
        DATE_FMT.setTimeZone(gmt);
    }

    private final String host;
    private final int port;
    private final String ak;
    private final String sk;

    public HttpURLConnectionSample(String host, int port, String ak, String
sk) {
        this.host = host;
        this.port = port;
        this.ak = ak;
        this.sk = sk;
    }
}
```

```

    }

    private String authorize(String httpVerb, String date, String bucket,
String objectName) throws
Exception {
    String stringToSign = httpVerb + "\n\n\n" + date + "\n/" + bucket + "/"
+ objectName;
    Mac mac = Mac.getInstance("HmacSHA1");
    mac.init(new SecretKeySpec(sk.getBytes("UTF-8"), "HmacSHA1"));
    byte[] macResult = mac.doFinal(stringToSign.getBytes("UTF-8"));
    String signature = new String(Base64.encodeBase64(macResult), "UTF-8");
    String authorization = "AWS " + ak + ":" + signature;
    return authorization;
}

    public void put(String bucket, String objName, byte[] data) throws
Exception {
    String date = DATE_FMT.format(new Date());
    String authorization = authorize("PUT", date, bucket, objName);
    URL url = new URL("http", host, port, "/" + bucket + "/" + objName);
    HttpURLConnection conn = (HttpURLConnection) url.openConnection();
    conn.setUseCaches(false);
    conn.setFixedLengthStreamingMode(data.length);
    conn.setRequestProperty("Date", date);
    conn.setRequestProperty("Authorization", authorization);
    conn.setConnectTimeout(CONN_TIMEOUT);
    conn.setReadTimeout(READ_TIMEOUT);
    conn.setDoOutput(true);
    conn.setRequestMethod("PUT");
    conn.connect();
    try (OutputStream out = conn.getOutputStream()) {
        out.write(data);
        out.flush();
    }
    int code = conn.getResponseCode();
    System.out.println("code=" + code);
}

```

```

public void get(String bucket, String objName) throws Exception {
    String date = DATE_FMT.format(new Date());
    String authorization = authorize("GET", date, bucket, objName);
    URL url = new URL("http", host, port, "/" + bucket + "/" + objName);
    HttpURLConnection conn = (HttpURLConnection) url.openConnection();
    conn.setUseCaches(false);
    conn.setRequestProperty("Date", date);
    conn.setRequestProperty("Authorization", authorization);
    conn.setConnectTimeout(CONN_TIMEOUT);
    conn.setReadTimeout(READ_TIMEOUT);
    conn.setDoInput(true);
    conn.setRequestMethod("GET");
    conn.connect();
    int code = conn.getResponseCode();
    System.out.println("code=" + code);
    try (InputStream in = conn.getInputStream()) {
        byte[] buffer = new byte[1024 * 8];
        while (in.read(buffer) != -1) {
            }
        }
    }

public void delete(String bucket, String objName) throws Exception {
    String date = DATE_FMT.format(new Date());
    String authorization = authorize("DELETE", date, bucket, objName);
    URL url = new URL("http", host, port, "/" + bucket + "/" + objName);
    HttpURLConnection conn = (HttpURLConnection) url.openConnection();
    conn.setRequestProperty("Date", date);
    conn.setRequestProperty("Authorization", authorization);
    conn.setConnectTimeout(CONN_TIMEOUT);
    conn.setReadTimeout(READ_TIMEOUT);
    conn.setRequestMethod("DELETE");
    conn.connect();
    int code = conn.getResponseCode();
    System.out.println("code=" + code);
}

public static void main(String[] args) throws Exception {

```

```
String host = "oos-nm2.ctyunapi.cn";
int port = 80;
String ak = "your_ak";
String sk = "your_sk";
String bucket = "bucket_name";
String objName = "object_name";
byte[] data = new byte[1024 * 1024 * 10];
Random rand = new Random();
rand.nextBytes(data);
HttpURLConnectionSample s = new HttpURLConnectionSample(host, port, ak,
sk);
s.put(bucket, objName, data);
s.get(bucket, objName);
s.delete(bucket, objName);
}
}
```

5 统计 API

5.1 统计 API 请求结构

- 接入地址

接入地址为：oos-cn-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	是否必填
Host	统计访问域名，访问域名是 oos-cn-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	V4 签名必填， V2 签名

	须首先将其转换为 UTC（协调世界时），然后提交为“20180801T083241Z”。	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

5.1.2 状态响应码

每个统计都包含状态响应码。

名称	描述
HTTP/1.1 xxx	状态码。 ● 200 xx 表示请求成功； ● 4xx, 5xx 表示请求失败。

5.1.3 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。

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**、**GetDeleteCapacity**、**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	是否必填
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 的间隔不大于 1 天； ● Freq=byHour，EndDate 与 BeginDate 的间隔不大于 7 天； ● Freq=byDay，EndDate 与 BeginDate 的间隔不大于 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。	否

	取值：3~63 位字符串，只能由小写字母、数字、短横线 (-) 和点 (.) 组成。	
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
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：ZhengZhou、ShenYang、ChengDu、WuLuMuQi、LanZhou、QingDao、GuiYang、LaSa、WuHu、WuHan、ShenZhen、SH2、SuZhou。	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 **lt-bucket1**、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=lt-bucket1&Freq=byHour&Region=ZhengZhou
HTTP/1.1
Date: Wed, 16 Oct 2019 06:24:41 GMT
Authorization: AWS 7799e793ce4624ee7e5a:fgiFELJ9zZTJ+MEfausUlfMCCAk=
Host: oos-cn-mg.ctyunapi.cn
```

● 返回示例

```
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@oos.cn</Account>
    <UserName>user_test1</UserName>
    <StorageClass>STANDARD</StorageClass>
    <TimeZone>UTC +0800</TimeZone>
    <Freq>byHour</Freq>
    <BucketName>lt-bucket1</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	是否必填
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=by5min，EndDate 与 BeginDate 的间隔不大于 1 天； ● Freq=byHour，EndDate 与 BeginDate 的间隔不大于 7 天； ● Freq=byDay，EndDate 与 BeginDate 的间隔不大于 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。	否

	默认值为 ALL 。	
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。 取值： 3~63 位字符串，只能由小写字母、数字、短横线 (-) 和点 (.) 组成。	否
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： ZhengZhou、ShenYang、ChengDu、WuLuMuQi、LanZhou、QingDao、GuiYang、LaSa、WuHu、WuHan、ShenZhen、SH2、SuZhou。	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: 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: ● 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.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: AWS 7799e793ce4624ee7e5a:e09bM67kLPEKaSCmZ4JERNCTmgQ=
Host: oos-cn-mg.ctyunapi.cn
```

● 返回示例

```
HTTP/1.1 200 OK
x-amz-request-id: 236A8905248E5A01
Date: Wed, 18Nov 2020 06:39:27 GMT
Content-Length:120
Connection: close
Server: CTYUN

<GetBilledStorageUsageResponse>

  <Account>test@test.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	是否必填
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 的间隔不大于 1 天； ● Freq=byHour，EndDate 与 BeginDate 的间隔不大于 7 天； ● Freq=byDay，EndDate 与 BeginDate 的间隔不大于 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。 取值 ：3~63 位字符串，只能由小写字母、数字、短横线 (-) 和点 (.) 组成。	否
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 ：ZhengZhou、ShenYang、ChengDu、WuLuMuQi、LanZhou、QingDao、GuiYang、LaSa、WuHu、WuHan、ShenZhen、SH2、SuZhou。	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
GetRestoreCapacityResponse. Account	发出请求的账户名。
GetRestoreCapacityResponse. UserName	发出请求的用户名。如果根用户发送的该请求，则 UserName 为 root。
GetRestoreCapacityResponse. StorageClass	存储类型： ● ALL：所有存储类型；

	● STANDARAD_IA: 低频访问类型。
GetRestoreCapacityResponse. TimeZone	返回数据的时区，统一为 UTC +0800。
GetRestoreCapacityResponse. 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。
GetRestoreCapacityResponse. BucketName	Bucket 名字。如果用户在请求中没有携带 Bucket 参数，BucketName 为空。
GetRestoreCapacityResponse. RegionName	数据位置，如果用户在请求中没有携带 Region 参数，RegionName 为空。
GetRestoreCapacityResponse. 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。
GetRestoreCapacityResponse. 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: AWS 7799e793ce4624ee7e5a:e09bM67kLPEKaSCmZ4JERNCTmgQ=
```

Host: oos-cn-mg.ctyunapi.cn

● 返回示例

```
HTTP/1.1 200 OK
x-amz-request-id: 236A8905248E5A01
Date: Wed, 18Nov 2020 07:39:02 GMT
Content-Length:120
Connection: close
Server: CTYUN

<GetRestoreCapacityResponse>
  <Account>example@oos.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	是否必填
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 的间隔不大于 1 天； ● Freq=byHour，EndDate 与 BeginDate 的间隔不大于 7 天； ● Freq=byDay，EndDate 与 BeginDate 的间隔不大于 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。	否

	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 位字符串，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： ZhengZhou、ShenYang、ChengDu、WuLuMuQi、LanZhou、QingDao、GuiYang、LaSa、WuHu、WuHan、ShenZhen、SH2、SuZhou。	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 **lt-bucket1**、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=lt-bucket1&Freq=byDay&Region=ZhengZhou HTTP/1.1
Date: Wed, 16 Oct 2019 06:38:27 GMT
Authorization: AWS 7799e793ce4624ee7e5a:e09bM67kLPEKaSCmZ4JERNCTmgQ=
```

Host: oos-cn-mg.ctyunapi.cn

● 返回示例

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@oos.cn</Account>

 <UserName>user_test1</UserName>

 <StorageClass>STANDARD</StorageClass >

 <TimeZone>UTC +0800</TimeZone>

 <Freq>byDay</Freq>

 <BucketName>lt-bucket1</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	是否必填
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 的间隔不大于 1 天； ● Freq=byHour，EndDate 与 BeginDate 的间隔不大于 7 天； ● Freq=byDay，EndDate 与 BeginDate 的间隔不大于 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。 取值： 3~63 位字符串，只能由小写字母、数字、短横线 (-) 和点 (.) 组成。	否
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： ZhengZhou、ShenYang、ChengDu、WuLuMuQi、LanZhou、QingDao、GuiYang、LaSa、WuHu、WuHan、ShenZhen、SH2、SuZhou。	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。	否

	类型：字符串。 取值： ● 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。	
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。
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 lt-bucket1、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=lt-bucket1&Freq=byDay&Region=ZhengZhou
HTTP/1.1

Date: Wed, 16 Oct 2019 06:41:57 GMT

Authorization: AWS 7799e793ce4624ee7e5a:5kj81zp55B/1+Kke0zjLzSMtEdQ=

Host: oos-cn-mg.ctyunapi.cn
```

● 返回示例

```
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@oos.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>lt-bucket1</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	是否必填
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 的间隔不大于 1 天； ● Freq=byHour，EndDate 与 BeginDate 的间隔不大于 7 天； ● Freq=byDay，EndDate 与 BeginDate 的间隔不大于 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 位字符串，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： ZhengZhou、ShenYang、ChengDu、WuLuMuQi、LanZhou、QingDao、GuiYang、LaSa、WuHu、WuHan、ShenZhen、SH2、SuZhou。	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； ● 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；	否

	● 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 lt-bucket1、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=lt-  
bucket1&Freq=byDay&Region=ZhengZhou HTTP/1.1  
  
Date: Wed, 16 Oct 2019 06:51:49 GMT  
  
Authorization: AWS 7799e793ce4624ee7e5a:4Rcw3Su7YLhYNiFmfS3mIMHoOl4=  
  
Host: oos-cn-mg.ctyunapi.cn
```

● 返回示例

```
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@oos.cn</Account>  
  <UserName>user_test1</UserName>  
  <TimeZone>UTC +0800</TimeZone>  
  <InternetType>all</InternetType>  
  <RequestsType>all</RequestsType>  
  <Freq>byDay</Freq>  
  <BucketName>lt-bucket1</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	是否必填
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 的间隔不大于 1 天； ● Freq=byHour，EndDate 与 BeginDate 的间隔不大于 7 天； ● Freq=byDay，EndDate 与 BeginDate 的间隔不大于 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 位字符串，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： ZhengZhou、ShenYang、ChengDu、WuLuMuQi、LanZhou、QingDao、GuiYang、LaSa、WuHu、WuHan、ShenZhen、SH2、SuZhou。	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。 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；	否

	● 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; ● Response503。 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	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;

OOS 开发者文档

	● 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.Internal.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.Internal.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.Internal.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.Internal.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; ● Response503。
Statistics.Standard_ia.Internal.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.Internal.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 **lt-bucket1**、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=lt-
```

```
bucket1&Freq=byDay&Region=ZhengZhou&RequestsType=get%2Bput&ResponseType=
Response200 HTTP/1.1
Date: Wed, 16 Oct 2019 07:03:54 GMT
Authorization: AWS 7799e793ce4624ee7e5a:BnaA3pDsSI mzdnPtN4eN2VcKh8g=
Host: oos-cn-mg.ctyunapi.cn
```

● 返回示例

```
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@oos.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>lt-bucket1</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	是否必填
Action	GetConcurrentConnection。	是
BeginDate	指定查询并发连接数的起始时间，统计数据的时间为开始日期（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 的间隔不大于 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 位字符串，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。	否

	取值：ZhengZhou、ShenYang、ChengDu、WuLuMuQi、LanZhou、QingDao、GuiYang、LaSa、WuHu、WuHan、ShenZhen、SH2、SuZhou。	
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 **lt-bucket1**、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=lt-bucket1&Freq=by5min&Region=ZhengZhou HTTP/1.1
Date: Wed, 16 Oct 2019 07:11:39 GMT
Authorization: AWS 7799e793ce4624ee7e5a:d4KttP1KihaC2YEI45Ot6dkqONE=
Host: oos-cn-mg.ctyunapi.cn
```

● 返回示例

```
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@oos.cn</Account>

<UserName>user_test1</UserName>

<TimeZone>UTC +0800</TimeZone>

<InternetType>all</InternetType>

<Freq>by5min</Freq>

<BucketName>lt-bucket1</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、GetDeleteCapacity、GetRequests、GetReturnCode、GetConcurrentConnection。

● 请求参数

名称	描述	是否必填
Action	GetUsage	是
BeginDate	指定查询的起始时间，统计数据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。 ● 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 位字符串，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 **my-bucket-test**，**2018-12-24** 的流量。

```
GET /?Action=GetUsage&BucketName=my-bucket-test&Freq=byDay&EndDate=2018-12-24&BeginDate=2018-12-24 HTTP/1.1
Host: oos-cn-mg.ctyunapi.cn
Authorization: AWS fad0e782cd5132563e38:xQE0diMbLRepdf3YB+FIEXAMPLE=
Date: Wed, 26 Dec 2018 09:01:30 GMT
Content-Type: application/xml; charset=utf-8
```

● 返回示例

```
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
Connection: close
Server: CTYUN

<GetUsageResponse>
<UserName>test@oos.cn</UserName>
```

```

<BucketName>my-bucket-test</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。

说明：不建议使用此 API，建议根据要查询的项，使用下列 API：

GetCapacity、GetDeleteCapacity、GetRequests、GetReturnCode、
GetConcurrentConnection。

● 请求参数

名称	描述	是否必填
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 位字符串，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 名称。
Region.Name	数据位置。

Region.Date.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 **my-bucket-test** 的已用带宽。

```
GET /?Action=GetBandwidth&BucketName=my-bucket-test&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:
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
Authorization: AWS4-HMAC-SHA256 Credential=fe49633483061d5b93e3/20181226/cn-mg/s3/aws4_request, SignedHeaders=content-type;host;x-amz-content-sha256;x-amz-date, Signature=40051b7b2fce14cd6025c8935019166814883696d017040cfd9334836fd40533
x-amz-date: 20181226T090108Z
Content-Type: application/xml; charset=utf-8
```

● 返回示例

```
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
Connection: close
Server: CTYUN

<?xml version="1.0" encoding="UTF-8"?>
<GetBandwidthResult>
  <UserName>ctyuntest@oos.cn</UserName>
  <BucketName>my-bucket-test</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

HTTP 响应码	错误码(code)	错误信息(message)	错误描述
403	AccessDenied	User: <user_Arn> is not authorized to perform: statistics: GetAccountStatisticsSummary on resource: <statistics_Arn>.	没有查询统计信息的权限
403	AuthorityNotMatch	AccountID: <account_ID> is not authorized to perform AvailableBandwidth.	无查询可用带宽权限或查询的数据位置无对应权限
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	Missing Freq.	没有输入参数 freq

6 操作跟踪 API

6.1 操作跟踪 API 请求结构

- 接入地址

接入地址为：oos-cn-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	是否必填
Host	操作跟踪访问域名，访问域名是 oos-cn-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

6.1.2 状态响应码

每个操作跟踪都包含状态响应码。

名称	描述
HTTP/1.1 xxx	状态码。 ● 200 xx 表示请求成功； ● 4xx, 5xx 表示请求失败。

6.1.3 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。

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	是否必填
Name	指定跟踪的名称。 类型：字符串。 取值：3~128 位字符串。 ● 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：oos://<bucket>/<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：oos://<bucket>/OOSLogs/<账号 ID>/CloudTrail/<年>/<月>/<日>/<日志数据文件>。	否

	类型： 字符串。 取值： 0-200 个字符。	
--	--	--

● 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 **test_bucket** 的 Bucket 中，日志前缀为 **auditlog**。

```
POST / HTTP/1.1
Host:oos-cn-cloudtrail.ctyunapi.cn
Authorization:signatureValue
X-Amz-Date:date
X-Amz-Target:cn.ctyunapi.oos-cn-cloudtrail.v20131101.CloudTrail_20131101.CreateTrail
Content-Type:application/json

{"Name":"test_cloud_trail","S3BucketName":"test_bucket","S3KeyPrefix":"auditlog"}
```

● 返回示例

```
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":"test_bucket","TrailARN":"arn:ctyun:cloudtrail::10rc2arpn6306:trail/test_cloud_trail","S3KeyPrefix":"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	是否必填
Name	指定跟踪的名称。 类型： 字符串。 取值： 3~128 位字符串。 ● 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
Authorization:signatureValue
X-Amz-Date:date
X-Amz-Target:cn.ctyunapi.oos-cn-cloudtrail.v20131101.CloudTrail_20131101.DeleteTrail
Content-Type:application/json

{"Name":"test_cloud_trail"}
```

● 返回示例

```
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	是否必填
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 位字符串。 ● 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（..）,点下划线（._）的组合。 说明： ● 如果名称列表为空，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":[]}	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

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
Authorization:signatureValue
X-Amz-Date:date
X-Amz-Target:cn.ctyunapi.oos-cn-cloudtrail.v20131101.CloudTrail_20131101.DescribeTrails
Content-Type:application/json

{"TrailNameList":["test_cloud_trail"]}
```

- 返回示例

```
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	是否必填
Name	指定跟踪的名称。 类型： 字符串。 取值： 3~128 位字符串。 ● 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。

- 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
Authorization:signatureValue
X-Amz-Date:date
X-Amz-Target:cn.ctyunapi.oos-cn-
cloudtrail.v20131101.CloudTrail_20131101.GetTrailStatus
Content-Type:application/json

{"Name":"test_cloud_trail"}
```

- 返回示例

```
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":155598466
3079,"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	是否必填
TrailName	指定跟踪的名称。 类型： 字符串。 取值： 3~128 位字符串。 ● 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
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
Authorization:signatureValue
X-Amz-Date:date
X-Amz-Target:cn.ctyunapi.oos-cn-
cloudtrail.v20131101.CloudTrail_20131101.PutEventSelectors
Content-Type:application/json

{"TrailName":"test_cloud_trail","EventSelectors":[{"ReadWriteType":"WriteOnly"}]}
```

● 返回示例

```
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::10rc2arpn6306: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	是否必填
TrailName	跟踪的名称。 类型： 字符串。 取值： 3~128 位字符串。 ● 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（..），点下划线（._）的组合。	是

● 响应结果

名称	描述
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
Authorization:signatureValue
X-Amz-Date:date
```

```
X-Amz-Target:cn.ctyunapi.oos-cn-  
cloudtrail.v20131101.CloudTrail_20131101.GetEventSelectors  
Content-Type:application/json  
  
{"TrailName":"test_cloud_trail"}
```

● 返回示例

```
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::1  
0rc2arpn6306: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	是否必填
Name	跟踪的名称。 类型：字符串型。 取值：3~128 位字符串。 ● 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
Authorization:signatureValue
X-Amz-Date:date
X-Amz-Target:cn.ctyunapi.oos-cn-
cloudtrail.v20131101.CloudTrail_20131101.UpdateTrail
Content-Type:application/json

{"Name":"test_cloud_trail","S3KeyPrefix":"auditLog2"}
```

- 返回示例

```
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",
"TrailARN":"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	是否必填
Name	跟踪的名称。 类型：字符串。 取值：3~128 位字符串。 ● 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
Authorization:signatureValue
X-Amz-Date:date
X-Amz-Target:cn.ctyunapi.oos-cn-
cloudtrail.v20131101.CloudTrail_20131101.StartLogging
Content-Type:application/json

{"Name":"test_cloud_trail"}
```

● 返回示例

```
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	是否必填
Name	跟踪的名称。 类型：字符串。 取值：3~128 位字符串。 ● 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
Authorization:signatureValue
X-Amz-Date:date
X-Amz-Target:cn.ctyunapi.oos-cn-
cloudtrail.v20131101.CloudTrail_20131101.StopLogging
Content-Type:application/json

{"Name":"test_cloud_trail"}
```

● 返回示例

```
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。用户可以查看过去 180 天内发生的管理事件。

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	是否必填
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 180 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 180 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 180 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。 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 180 天前的时间，则返回距当前时间往前数的第 180 天那天的数据。	否

LookupAttributes	指定管理事件的查询属性。 类型： 字符串。 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。	否

	取值：与上条响应结果中的参数值一样。	
--	--------------------	--

● 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
Authorization:signatureValue
X-Amz-Date:date
X-Amz-Target:cn.ctyunapi.oos-cn-
cloudtrail.v20131101.CloudTrail_20131101.LookupEvents
Content-Type:application/json

{"LookupAttributes":[{"AttributeKey":"ReadOnly","AttributeValue":"true"}]}
```

● 返回示例

```
HTTP/1.1 200 OK
x-amz-request-id:c1a075bc84f4277
Content-Type: application/json; charset=utf-8
Content-Length:23994
Date:Thu, 02 Jan 2020 07:16:20 GMT
Server:CTYUN

{
  "Events": [
    {
      "CloudTrailEvent": {
```

```

        "eventId": "7645426409000523408",
        "resource":
    [{"Resources":[{"name":"testTrail1","type":"CloudTrail
Trail","arn":"arn:ctyun:cloudtrail::1rl1uq2mvq6ob:trail/testTrail1"}]}],
        "eventVersion": "1.06",
        "eventSource": "oos-cn.ctyun.cn",
        "requestParameters": {"Name":"testTrail1"},
        "userAgent": "oos-cn.ctyun.cn",
        "readOnly": true,
        "userIdentity": {
            "accountId": "1rl1uq2mvq6ob",
            "principalId": "1rl1uq2mvq6ob",
            "type": "Root",
            "arn": "arn:ctyun:iam::1rl1uq2mvq6ob:root"
        },
        "eventType": "ApiCall",
        "serviceName": "CloudTrail",
        "sourceIp": "127.0.0.1",
        "requestId": "8bb75b1394fa42cb",
        "requestURL": "http://oos-cn-cloudtrail.ctyunapi.cn/",
        "eventTime": "2020-01-02T06:13:47Z",
        "eventName": "GetTrailStatus",
        "requestRegion": "cn",
        "managementEvent": true
    }
},
{
    "CloudTrailEvent": {
        "eventId": "7645426419220464293",

```

```

        "eventVersion": "1.06",
        "eventSource": "oos-cn.ctyun.cn",
        "requestParameters": "{\"TrailNameList\": [\"testTrail1\"]}",
        "userAgent": "oos-cn.ctyun.cn",
        "readOnly": true,
        "userIdentity":
        {
            "accessKeyId": "d2756b7f524227e79a83",
            "accountId": "1fmr2raot5sp",
            "principalId": "e943d0ae1a3b461bb0a18fb8f2e55d44",
            "type": "IAMUser",
            "arn": "arn:ctyun:iam::1fmr2raot5sp:user/subUser",
            "userName": "subUser"
        },
        "eventType": "ApiCall",
        "serviceName": "CloudTrail",
        "sourceIp": "127.0.0.1",
        "requestId": "a8cde8aea5744b74",
        "requestURL": "http://oos-cn-cloudtrail.ctyunapi.cn/",
        "eventTime": "2020-01-02T06:13:37Z",
        "eventName": "DescribeTrails",
        "requestRegion": "cn",
        "managementEvent": true
    }
}
]
}

```

6.4 错误码列表

HTTP 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 <i>trailName</i> already exists for the customer: <i>accountId</i>	跟踪已经存在。
400	TrailNotFoundException	Unknown trail: arn:ctyun:CloudTrail:ctyun:accountId:trail/ <i>trailName</i> for the customer: <i>accountId</i>	跟踪不存在。
400	MaximumNumberOfTrailsExceededException	User: <i>accountId</i> already has 10 trails.	跟踪超过最大数量限制。
400	S3BucketDoesNotExistException	Bucket name does not exist: <i>bucketName</i>	创建跟踪时， <i>bucketName</i> 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: <i>s3Prefix</i>	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 names must not be formatted as an IP address.	trail 名称不合法，是 ip 格式。
400	InvalidTrailNameException	Trail name must ends 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 starts 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
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
	DeleteAccessKey
	ListAccessKeys
	UpdateAccessKey
	TagUser
	ChangePassword
	CreateLoginProfile
	CreateVirtualMFADevice
	DeactivateMFADevice
	DeleteAccountPasswordPolicy
	DeleteLoginProfile
	DeleteVirtualMFADevice
	EnableMFADevice
	GetAccountPasswordPolicy
	GetLoginProfile
	ListVirtualMFADevices
	UpdateAccountPasswordPolicy
	UpdateLoginProfile
	CreatePolicy
	DeletePolicy
	AttachGroupPolicy
	DetachGroupPolicy
	GetPolicy
	ListAttachedUserPolicies
	AttachUserPolicy
	ListAttachedGroupPolicies

OOS 开发者文档

	ListPolicies
	GetAccountSummary
	DetachUserPolicy
	ListEntitiesForPolicy
	UnTagUser
	ListMFADevices
	GetSessionToken

7 访问控制（IAM）API

7.1 IAM API 请求结构

- 接入地址

IAM 服务的接入地址为：oos-cn-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	是否必填
Host	IAM 访问域名，IAM 的访问域名是 oos-cn-iam.ctyunapi.cn。	是
Authorization	请求头签名。	是

	支持 V4 签名认证。 类型： 字符串。	
X-Amz-Date	日期和时间格式必须遵循 ISO 8601 标准，并且必须使用“yyyyMMddTHH:mm:ssZ”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

● 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
HTTP/1.1 XXX	状态码。 ● 200 OK 表示请求成功； ● 4XX, 5XX 表示请求失败。
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。

● 公共返回参数

每个 IAM API 响应结果中，除了包含公共响应头、每个 IAM API 相应的响应参数（见具体 API 的响应结果），还包含公共返回参数。下表为公共的响应参数描述。

名称	描述
RequestId	请求 ID。

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。本文档中的返回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。
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。
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。

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	是否必填
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 位字符串组成，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 位字符串组成，字符串符合表达式 <code>[\p{L}\p{Z}\p{N}_:/=+\-@]+</code> ；Tags.member.N.Value 由 0~256 位字符串组成，字符串符合表达式 <code>[\p{L}\p{Z}\p{N}_:/=+\-@]*</code> 。 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。	否

- 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
Authorization: signatureValue
X-Amz-Date: date
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
```

● 返回示例

```
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	是否必填
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 位字符串组成，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
Authorization:signatureValue
X-Amz-Date:date
Content-Type: application/octet-stream

Action=GetUser&Version=2010-05-08

```

● 返回示例

```

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	是否必填
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 响应元素为 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
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 位字符串组成，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。 UserName、AccessKeyId 共用时，查询出同时满足多个查询条件的结果。	否

● 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.PasswordLast Used	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
Authorization:signatureValue
X-Amz-Date:date
Content-Type: application/octet-stream

Action=ListUsers&Version=2010-05-08
```

● 返回示例

```
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。

- 请求参数

名称	描述	是否必填
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 位字符串组成，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
Authorization:signatureValue
X-Amz-Date:date
Content-Type: application/octet-stream

Action=DeleteUser&UserName=test_user&Version=2010-05-08
```

- 返回示例

```
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	是否必填
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 位字符串组成，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 位字符串组成，字符串符合表达式 <code>[\p{L}\p{Z}\p{N}_./=+\-@]+</code> 。 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。 ● 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。	是
Tags.member.N.Value	用户标签值。 类型：字符串。	是

	取值： 由 0~256 位字符串组成。字符串符合表达式 $[\backslash p\{L\}\backslash p\{Z\}\backslash p\{N\}_\cdot : / = + \backslash - @]^*$ 。	
--	---	--

● 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
Authorization: signatureValue
X-Amz-Date: date
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
```

● 返回示例

```
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	是否必填
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 位字符串组成，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 位字符串组成，字符串符合表达式 <code>[\p{L}\p{Z}\p{N}_./=+\-@]+</code> 。	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
Authorization: signatureValue
X-Amz-Date: date
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
```

- 返回示例

```
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	是否必填
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 位字符串组成，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 响应元素为 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	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
Authorization: signatureValue
X-Amz-Date: date
Content-Type: application/octet-stream
Content-Length: 55

Action=ListUserTags&Version=2010-05-08&UserName=test_user
```

● 返回示例

```
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>
```

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	是否必填
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 位字符串组成，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 响应元素为 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
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
Authorization: signatureValue
X-Amz-Date: date
Content-Type: application/octet-stream
Content-Length: 64

Action=ListGroupForUser&Version=2010-05-08&UserName=test_user_2
```

● 返回示例

```
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

<ListGroupForUserResponse>
  <ListGroupForUserResult>
    <IsTruncated>>false</IsTruncated>
    <Groups>
      <member>
        <GroupName>test_group1</GroupName>
```

```
<Arn>arn:ctyun:iam::10rc2arpn6306:group/test_group1</Arn>
  <GroupId>514648bfb4e423f867a25281642cdfc</GroupId>
  <CreateDate>2019-01-30T06:11:39Z</CreateDate>
</member>
</Groups>
</ListGroupForUserResult>
<ResponseMetadata>
  <RequestId>2451e997f937470a</RequestId>
</ResponseMetadata>
</ListGroupForUserResponse>
```

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	是否必填
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 位字符串组成，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
Authorization: signatureValue
X-Amz-Date:date
Content-Type: application/octet-stream

Action=CreateAccessKey&UserName=test_user_2&Version=2010-05-08
```

● 返回示例

```
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>
    </AccessKey>
    <UserName>test_user_2</UserName>
    <CreateDate>2019-01-31T09:18:00Z</CreateDate>
    <Status>Active</Status>
  </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	是否必填
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 位字符串组成，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 响应元素为 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
----	----

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
Authorization:signatureValue
X-Amz-Date:date
Content-Type: application/octet-stream

Action=ListAccessKeys&Version=2010-05-08
```

● 返回示例

```
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 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	是否必填
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 位字符串组成，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
Authorization: signatureValue
X-Amz-Date: date
Content-Type: application/octet-stream

Action=UpdateAccessKey&Status=Inactive&Version=2010-05-08&AccessKeyId=535d97103885f22be7a1
```

● 返回示例

```
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>
```

7.3.12 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	是否必填
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 位字符串组成，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
Authorization: signatureValue
X-Amz-Date: date
Content-Type: application/octet-stream

Action=DeleteAccessKey&Version=2010-05-08&AccessKeyId=535d97103885f22be7a1

```

● 返回示例

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>

7.3.13 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	是否必填
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 位字符串组成，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。 ● 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

● 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
Authorization: signatureValue
X-Amz-Date: date
Content-Type: application/octet-stream
Content-Length: 112

Action=CreateLoginProfile&Version=2010-05-08&UserName=test_ch_passwd&Password=a12345678&PasswordResetRequired=true
```

● 返回示例

```
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>
```

7.3.14 GetLoginProfile

此操作用来获取 IAM 用户控制台登录密码创建的时间、及用户首次登录后是否需要修改密码。

● 请求参数

名称	描述	是否必填
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 位字符串组成，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

Authorization: *signatureValue*

X-Amz-Date: *date*

Content-Type: application/octet-stream

Content-Length: 58

Action=GetLoginProfile&Version=2010-05-08&UserName=test_user

● 返回示例

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>

7.3.15 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	是否必填
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 位字符串组成，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
Authorization: signatureValue
X-Amz-Date: date
Content-Type: application/octet-stream
Content-Length: 68

Action=UpdateLoginProfile&Version=2010-05-08&UserName=test_ch_passwd&Password=a12345678
```

● 返回示例

```
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>
```

7.3.16 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	是否必填
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 位字符串组成，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
Authorization: signatureValue
X-Amz-Date: date
Content-Type: application/octet-stream
Content-Length: 68

Action=DeleteLoginProfile&Version=2010-05-08&UserName=test_ch_passwd
```

- 返回示例

```
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>

7.3.17 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	是否必填
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
Authorization:signatureValue
X-Amz-Date: date
Content-Type: application/octet-stream
Content-Length: 72

Action=ChangePassword&Version=2010-05-08&OldPassword=12345678a&NewPassword=12345678b
```

● 返回示例

```
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

<UpdateAccessKeyResponse>

 <ResponseMetadata>

 <RequestId>3d19f89e195b4c77</RequestId>

 </ResponseMetadata>

</UpdateAccessKeyResponse>

7.3.18 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	是否必填
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 位字符串组成，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
Authorization: signatureValue
X-Amz-Date: date
Content-Type: application/octet-stream
Content-Length: 74

Action=CreateVirtualMFADevice&Version=2010-05-08&VirtualMFADeviceName=mfa1
```

● 返回示例

```
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>

        OBAZWZBY2WENX22KUJUUVPKZ7OHGXBO23L52ZXHLVYIBNXSJTKAR7
        AZ5FH5RQLV          </Base32StringSeed>
      <QRCodePNG><!-- byte array of png file --></QRCodePNG>
      <SerialNumber>arn:ctyun:iam::10rc2arpn6306:mfa/mfa1</SerialNumber>
    </VirtualMFADevice>
  </CreateVirtualMFADeviceResult>
  <ResponseMetadata>
    <RequestId>1a7e651e412c4ecc</RequestId>
  </ResponseMetadata>
</CreateVirtualMFADeviceResponse>
```

7.3.19 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	是否必填
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 位字符串组成，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
Authorization: signatureValue
X-Amz-Date: date
Content-Type: application/octet-stream
Content-Length: 175

Action=EnableMFADevice&Version=2010-05-08&UserName=test_user&SerialNumber=arn%3Aactyun%3Aiam%3A%3A10rc2arpn6306%3Amfa%2Fmfa1&AuthenticationCode1=048078&AuthenticationCode2=856474
```

● 返回示例

```
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>
```

7.3.20 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	是否必填
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 响应元素为 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	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
Authorization: signatureValue
X-Amz-Date: date
Content-Type: application/octet-stream
Content-Length: 73

Action=ListVirtualMFADevices&Version=2010-05-08&AssignmentStatus=Assigned
```

● 返回示例

```
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>
```

7.3.21 ListMFADevices

此操作用来列出 IAM 用户的虚拟 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的虚拟 MFA 设备。

● 请求参数

名称	描述	是否必填
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 位字符串组成，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 响应元素为 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	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
Authorization: signatureValue
X-Amz-Date: date
Content-Type: application/octet-stream
Content-Length: 76
Connection: Keep-Alive

Action=ListMFADevices&Version=2010-05-08&UserName=test_ch_passwd&MaxItems=10
```

● 返回示例

```
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>
```

7.3.22 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	是否必填
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 位字符串组成，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
Authorization: signatureValue
X-Amz-Date: date
Content-Type: application/octet-stream
Content-Length: 125

```

```
Action=DeactivateMFADevice&Version=2010-05-08&UserName=test_user&SerialNumber=arn%3Aactyun%3Aiam%3A%3A10rc2arpn6306%3Amfa%2Fmfa1
```

● 返回示例

```
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>
```

7.3.23 DeleteVirtualMFADevice

此操作用来删除指定的虚拟 MFA 设备。

● 请求参数

名称	描述	是否必填
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
Authorization: signatureValue
X-Amz-Date: date
Content-type: application/octet-stream
Content-Length: 108

Action=DeleteVirtualMFADevice&Version=2010-05-08&SerialNumber=arn%3Actyun%3Aiam%3A%3A10rc2arpn6306%3Amfa%2Fmfa2

```

● 返回示例

```

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	是否必填
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 位字符串组成，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
Authorization:signatureValue
X-Amz-Date:date
Content-type: application/octet-stream

Action=CreateGroup&Version=2010-05-08&GroupName=test_group
```

● 返回示例

```
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>514648bfb4e423f867a25281642cdfc</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	是否必填
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 位字符串组成，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 响应元素为 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 Authorization: <i>signatureValue</i> X-Amz-Date: <i>date</i> Content-type: application/octet-stream Action=GetGroup&Version=2010-05-08&GroupName=test_group
--

● 返回示例

```

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>

```

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	是否必填
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 位字符串组成，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 位字符串组成，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
Authorization:signatureValue
X-Amz-Date:date
Content-type: application/octet-stream
```

Action=AddUserToGroup&Version=2010-05-08&UserName=test_user&GroupName=test_group

- 返回示例

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	是否必填
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 位字符串组成，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 位字符串组成，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
Authorization:signatureValue
X-Amz-Date:date
Content-type: application/octet-stream

Action=RemoveUserFromGroup&Version=2010-05-08&GroupName=test_group&UserName=test_user
```

- 返回示例

```
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	是否必填
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 位字符串组成，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 响应元素为 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	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
Authorization:signatureValue
X-Amz-Date:date
Content-type: application/octet-stream

Action=ListGroup&Version=2010-05-08&MaxItems=2
```

● 返回示例

```
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

<ListGroupResponse>
  <ListGroupResult>
    <IsTruncated>true</IsTruncated>
    <Groups>
      <member>
        <GroupName>PM</GroupName>
        <Arn>arn:ctyun:iam::10rc2arpn6306:group/PM</Arn>
```

```
<GroupId>514648bfb4e423f867a25281642cdfc</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>614648bfb4e423f867a25281642cdfc</GroupId>
  <CreateDate>2019-01-07T05:39:03Z</CreateDate>
  <Policies>10</Policies>
  <Users>10</Users>
</member>
</Groups>
<Marker>10rc2arpn6306|test_group</Marker>
</ListGroupResult>
<ResponseMetadata>
  <RequestId>1d1089f15291424a</RequestId>
</ResponseMetadata>
</ListGroupResponse>
```

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	是否必填
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 位字符串组成，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
Authorization:signatureValue
X-Amz-Date:date
Content-type: application/octet-stream

Action=DeleteGroup&Version=2010-05-08&GroupName=test_group
```

● 返回示例

```
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	是否必填
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 位字符串组成，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。 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； ● 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）	是

● 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。

```
POST / HTTP/1.1
Host: oos-cn-iam.ctyunapi.cn
Authorization: signatureValue
X-Amz-Date: date
Content-type: application/octet-stream
Content-Length: 502

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%22s3%3AListAllMyBuckets%22%2C%22Resource%22%3A%22arn%3Aactyun%3As3%3A%3A%3A*%22%7D%2C%7B%22Effect%22%3A%22Allow%22%2C%22Action%22%3A%5B%22s3%3AGet*%22%2C%22s3%3AList*%22%5D%2C%22Resource%22%3A%5B%22arn%3Aactyun%3As3%3A%3A%3AEXAMPLE-BUCKET%22%2C%22arn%3Aactyun%3As3%3A%3A%3AEXAMPLE-BUCKET%2F*%22%5D%7D%5D%7D&Description=test_des
```

● 返回示例

```
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::10rc2arpn6306: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	是否必填
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
Authorization: signatureValue
X-Amz-Date: date
Content-type: application/octet-stream
Content-Length: 103

Action=GetPolicy&Version=2010-05-
08&PolicyArn=arn%3Actyun%3Aiam%3A%3A10rc2arpn6306%3Apolicy%2Ftest_pol
icy0
```

● 返回示例

```
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>
    </Policy>
  </GetPolicyResult>
</GetPolicyResponse>
```

```
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	是否必填
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 位字符串组成，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 响应元素为 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
Authorization: signatureValue
X-Amz-Date: date
Content-type: application/octet-stream
Content-Length: 61

Action=ListPolicies&Version=2010-05-08&MaxItems=1
```

- 返回示例

```

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	是否必填
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 响应元素为 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	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
Authorization: signatureValue
X-Amz-Date: date
Content-type: application/octet-stream
Content-Length: 115

Action=ListEntitiesForPolicy&Version=2010-05-08&PolicyArn=arn%3Aactyun%3Aiam%3A%3A10rc2arpn6306%3Apolicy%2Ftest_policy0
```

● 返回示例

```
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>514648bfb4e423f867a25281642cdfc</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	是否必填
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
Authorization: signatureValue
X-Amz-Date: date
Content-type: application/octet-stream
Content-Length: 105

Action=DeletePolicy&Version=2010-05-08&PolicyArn=arn%3Actyun%3Aiam%3A%3A10rc2arpn6306%3Apolicy%2Ftest_policy
```

● 返回示例

```
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	是否必填
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 位字符串组成，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
Authorization: signatureValue
X-Amz-Date: date
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
```

● 返回示例

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	是否必填
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 位字符串组成，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 响应元素为 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	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
Authorization: signatureValue
X-Amz-Date: date
Content-type: application/octet-stream
Content-Length: 82

Action=ListAttachedUserPolicies&Version=2010-05-08&UserName=test_user_2&MaxItems=2
```

● 返回示例

```
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	是否必填
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 位字符串组成，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
Authorization: signatureValue
X-Amz-Date: date
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
```

● 返回示例

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	是否必填
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 位字符串组成，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
Authorization: signatureValue
X-Amz-Date: date
Content-type: application/octet-stream
Content-Length: 133

Action=AttachGroupPolicy&Version=2010-05-08&GroupName=test_group1&PolicyArn=arn%3Actyun%3Aiam%3A%3A10rc2arpn6306%3Apolicy%2Ftest_policy0
```

● 返回示例

```
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	是否必填
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 位字符串组成，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 响应元素为 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	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
Authorization: signatureValue
X-Amz-Date: date
Content-type: application/octet-stream
Content-Length: 84

Action=ListAttachedGroupPolicies&Version=2010-05-08&GroupName=test_group1&MaxItems=2
```

● 返回示例

```
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	是否必填
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 位字符串组成，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
Authorization: signatureValue
X-Amz-Date: date
Content-type: application/octet-stream
Content-Length: 133

Action=DetachGroupPolicy&Version=2010-05-08&GroupName=test_group1&PolicyArn=arn%3Actyun%3Aiam%3A%3A10rc2arpn6306%3Apolicy%2Ftest_policy0
```

● 返回示例

```
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	是否必填
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：整数形式。 取值：0~1095, 0 表示永不过期，默认值为 0。	否
MinimumPasswordLength	控制台登录密码最短的长度。 类型：整数形式。 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： 整数形式。 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。	否

	默认值为 false 。	
--	---------------------	--

● 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
Authorization: signatureValue
X-Amz-Date: date
Content-type: application/octet-stream
Content-Length: 152

Action=UpdateAccountPasswordPolicy&Version=2010-05-08&RequireLowercaseCharacters=true&AllowUsersToChangePassword=true&MaxPasswordAge=10&HardExpiry=false
```

● 返回示例

```
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	是否必填
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
Authorization: signatureValue
X-Amz-Date: date
Content-type: application/octet-stream
Content-Length: 50

Action=GetAccountPasswordPolicy&Version=2010-05-08
```

● 返回示例

```
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	是否必填
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
Authorization: signatureValue
X-Amz-Date: date
Content-type: application/octet-stream
Content-Length: 53

Action=DeleteAccountPasswordPolicy&Version=2010-05-08
```

● 返回示例

```
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>
```

7.6 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	是否必填
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
Authorization: signatureValue
X-Amz-Date: date
Content-type: application/octet-stream
```

Content-Length: 43

Action=GetAccountSummary&Version=2010-05-08

● 返回示例

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

HTTP 响应码	错误码(code)	错误信息(message)	错误描述
403	AccessDenied	The old password was incorrect.	更改密码时，旧密码不正确。
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: <userArn> is not authorized to perform: iam:DeleteAccessKey on resource: <userArn>.	没有权限删除 AK。
403	AccessDenied	User: <userArn> is not authorized to perform: iam:RemoveUserFromGroup on resource: <groupArn>.	用户没有从用户组中移除用户的权限。
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/SecretKey 时，不允许删除。
409	DeleteConflict	Cannot delete entity, must delete MFA device first.	DeleteUser 时，必须先删除 MFA 设备。
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	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。

OOS 开发者文档

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。
400	InvalidInput	Arn <policyArn> is not valid.	策略的 Arn 不合法。
400	InvalidAction	Could not find operation <APIName> for version <version>.	Version 错误。
403	InvalidAuthenticationCode	Authentication code for device is not valid.	MFA 身份认证码不正确。
400	InvalidInput	Duplicate tag keys found. Please note that Tag keys are case insensitive.	设置了重复的标签 key。

500	InternalError	We encountered an internal error. Please try again.	发生内部错误
403	InvalidAccessKeyId	The AccessKeyId is invalid.	密钥无效。
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: Allow1	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 empty.	创建策略时，有 Resource，但是 Resource 是空。

400	MalformedPolicyDocument	Statement/policy already has instance of Resource.	创建策略时，Resource 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。
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 设备未分配给用户。
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::10rc2arpn6308:policy/test_2policy 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 535d97103885f22be7m1 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。
400	PasswordPolicy Violation	Password does not conform to the account password policy.	更改密码时，密码不符合密码策略。
400	PasswordPolicy Violation	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 must satisfy regular expression pattern: [\p{L}\p{Z}\p{N}_.:/=+\-@]*; Value 'test_key?' at 'tags.1.member.key' failed to	创建用户时，标签名称或值非法。

		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 '11111' at 'assignmentStatus' failed to satisfy constraint: Member must satisfy enum value set: [Unassigned, Any, Assigned]	ListVirtualMFADevicesResult 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 satisfy enum value set: [User, Group]	ListEntitiesForPolicy 时，EntityFilter 不正确，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。

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/*"
      ]
    }
  ]
}
```

```

    ]
  }
]
}

```

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。

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。

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：获取对象操作。

```
"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*"
```

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": "*",
    }]
}

```

Resource

Resource 元素指定执行策略的资源，可以指定一个或多个对象。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。

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。

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。

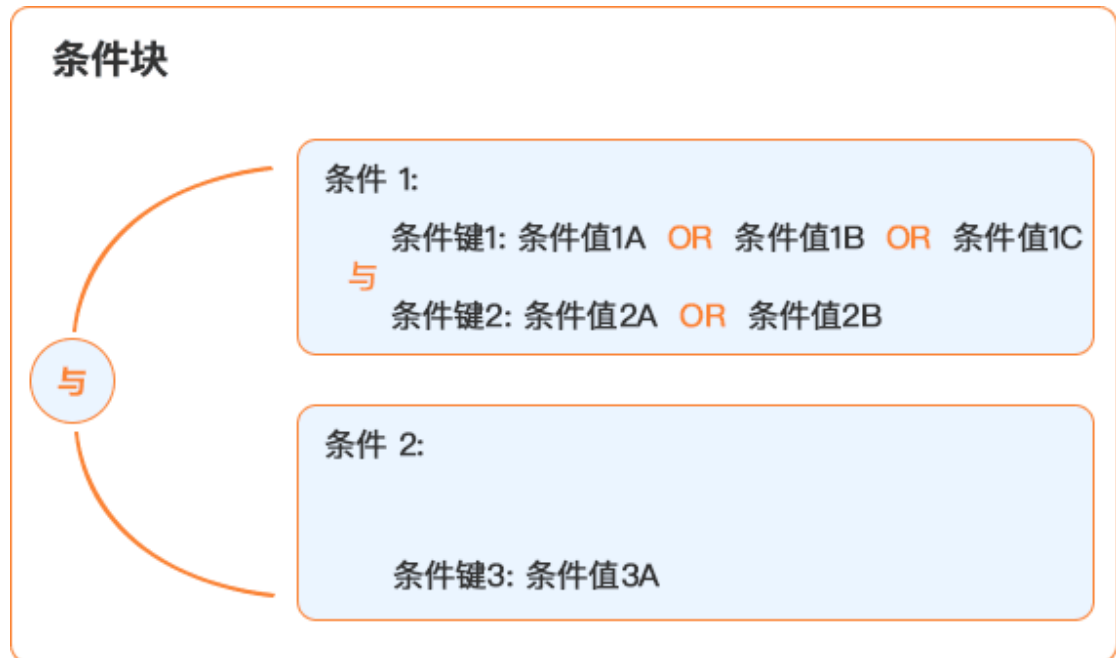
Condition

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	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。

	定日期； ● DateGreaterThanEquals: 晚于或等于指定日期。	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。 ● 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。	包含数字和小写字母的 32 位字符串。 运算符为 StringLike 和 StringNotLike ，可以包含通配符。
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；	1~64 位字符串组成，字符只能包含字母、数字或特殊字符，特殊字符只

	● 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。	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； ● 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。	字符串，可以包含特殊字符。
ctyun:Referer	● 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: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。	整数形式。
oos:prefix	● StringEquals: 精准匹配指	字符串形式。

	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 (?)。	说明：本条件键仅对 ListBucket 生效。
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	字符串形式。 取值为： ● private: 私有 ● public-read: 只读 ● public-read-write: 公有 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。

	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 (?)。	
--	--	--

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。

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 } }
```

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}`**”。当策略执行时，策略变量将被替换为请求本身的用户名。

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

表1 OOS 的操作权限与 API 对应关系

操作权限		API
Bucket 列表	ListBucket	GET Bucket（List Object）、HEAD Bucket
	ListAllMyBucket	GET Service
	GetRegions	GET Regions
Bucket 读取	ListBucketMultipartUploads	List Multipart Uploads
	GetBucketAcl	GET Bucket acl
	GetBucketAccelerate	GET Bucket accelerate
	GetBucketLocation	GET Bucket location
	GetBucketPolicy	GET Bucket policy
	GetLifecycleConfiguration	GET Bucket lifecycle
	GetBucketWebsite	GET Bucket website
	GetBucketCORS	GET Bucket cors
	GetBucketLogging	GET Bucket logging
	GetBucketObjectLockConfiguration	GET Bucket Object Lock
Bucket 写入	PutBucket	PUT Bucket
	DeleteBucket	DELETE Bucket
	DeleteMultipleObjects	DELETE Multiple Objects
	PutLifecycleConfiguration	PUT Bucket lifecycle、DELETE Bucket lifecycle
	PutBucketWebsite	PUT Bucket website
	DeleteBucketWebsite	DELETE Bucket website
	PutBucketCORS	PUT Bucket cors、DELETE Bucket cors
	PutBucketLogging	PUT Bucket logging
	PutAccelerateConfiguration	PUT Bucket accelerate
	PutBucketObjectLockConfiguration	PUT Bucket Object Lock
	DeleteBucketObjectLockConfiguration	DELETE Bucket Object Lock
Bucket 权限	PutBucketPolicy	PUT Bucket policy
	DeleteBucketPolicy	DELETE Bucket policy
Object 读取	ListMultipartUploadParts	List Parts
	GetObject	GET Object、HEAD Object
Object 写入	PutObject	PUT Object、PUT Object-Copy、POST Object、Initiate Multipart Upload、Upload Part、Complete Multipart Upload、Upload Part - Copy
	DeleteObject	DELETE Object
	AbortMultipartUpload	Abort Multipart Upload

表2 统计的操作权限与 API 对应关系

操作权限	API
GetAccountStatisticsSummary	GET Capacity、GET DeleteCapacity、GET Traffics、GET AvailableBandwidth、GET Requests、GET RetainCode、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	API
列表	DescribeTrails
	LookupEvents
读取	GetEventSelectors
	GetTrailStatus
写入	PutEventSelectors
	StopLogging
	CreateTrail
	UpdateTrail
	DeleteTrail
	StartLogging

表4 IAM 的操作权限与 API 对应关系

操作权限	API
列表	GetAccountSummary
	GetLoginProfile
	ListAccessKeys
	ListUsers
	ListUserTags
	ListGroups
	ListGroupsForUser
	ListPolicies
	ListAttachedGroupPolicies
	ListAttachedUserPolicies
	ListEntitiesForPolicy
	ListMFADevices
	ListVirtualMFADevices
读取	GetUser
	GetGroup
	GetPolicy
	GetAccountPasswordPolicy
写入	CreateAccessKey

	DeleteAccessKey	DeleteAccessKey
	UpdateAccessKey	UpdateAccessKey
	CreateUser	CreateUser
	DeleteUser	DeleteUser
	TagUser	TagUser
	UntagUser	UntagUser
	CreateGroup	CreateGroup
	DeleteGroup	DeleteGroup
	AddUserToGroup	AddUserToGroup
	RemoveUserFromGroup	RemoveUserFromGroup
	ChangePassword	ChangePassword
	UpdateAccountPasswordPolicy	UpdateAccountPasswordPolicy
	DeleteAccountPasswordPolicy	DeleteAccountPasswordPolicy
	CreateVirtualMFADevice	CreateVirtualMFADevice
	DeactivateMFADevice	DeactivateMFADevice
	DeleteVirtualMFADevice	DeleteVirtualMFADevice
	EnableMFADevice	EnableMFADevice
	CreateLoginProfile	CreateLoginProfile
	DeleteLoginProfile	DeleteLoginProfile
	UpdateLoginProfile	UpdateLoginProfile
权限	CreatePolicy	CreatePolicy
	DeletePolicy	DeletePolicy
	AttachUserPolicy	AttachUserPolicy
	DetachUserPolicy	DetachUserPolicy
	AttachGroupPolicy	AttachGroupPolicy
	DetachGroupPolicy	DetachGroupPolicy