下载与安装

相关资源

* SDK下载地址: 见帮助中心

环境依赖

Go 1.9 版本及以上， [go1.16.4.linux-amd64.tar.gz](https://golang.org/dl/go1.16.4.linux-amd64.tar.gz)。

* 环境配置

tar -C /usr/local -zxvf go1.16.4.linux-amd64.tar.gz

echo -e 'GOROOT=/usr/local/go \nPATH=$PATH:$GOROOT/bin' >> /etc/profile

source /etc/profile

go env -w GO111MODULE=off

SDK安装

* 离线安装

go env -w GOPATH=/root/GoWorks #设置工作区目录

mkdir -p /root/GoWorks/src

tar -C /root/GoWorks/src -zxvf [zos-sdk-go.tar.gz](https://gitlab.engineering.ctyun.cn/CStor/zos/zos-sdk-go/raw/master/zos-sdk-go.tar.gz)

连接

在正式使用SDK接口之前，需要先连接ZOS，使用以下的示例可以连接ZOS(连接示例基于go1.16.4，同以下所有接口示例)。

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.16.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

}

Topic操作与其余操作不同，需要使用以下的示例连接ZOS(连接示例基于go1.16.4，同以下所有接口示例)。

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/service/sns"

)

func main() {

accessKey := "test"

secretKey := "test"

endPoint := "192.168.198.110:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(accessKey, secretKey, ""),

Endpoint: aws.\*string(endPoint),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

})

svc := sns.New(sess)

}

全局错误码定义

请求可能会返回相关错误，具体错误码编号及信息请参考下表。同一个错误码可能对应不同的错误码描述，具体由接口来决定。

|  |  |
| --- | --- |
| 错误码 | 错误码描述 |
| 100 | Continue |
| 200 | Success |
| 201 | Created |
| 202 | Accepted |
| 204 | NoContent |
| 206 | Partial content |
| 304 | NotModified |
| 400 | InvalidArgument |
| 400 | InvalidDigest |
| 400 | BadDigest |
| 400 | InvalidBucketName |
| 400 | InvalidObjectName |
| 400 | UnresolvableGrantByEmailAddress |
| 400 | InvalidPart |
| 400 | InvalidPartOrder |
| 400 | RequestTimeout |
| 400 | EntityTooLarge |
| 403 | AccessDenied |
| 403 | UserSuspended |
| 403 | RequestTimeTooSkewed |
| 404 | NoSuchKey |
| 404 | NoSuchBucket |
| 404 | NoSuchUpload |
| 405 | MethodNotAllowed |
| 408 | RequestTimeout |
| 409 | BucketAlreadyExists |
| 409 | BucketNotEmpty |
| 411 | MissingContentLength |
| 412 | PreconditionFailed |
| 416 | InvalidRange |
| 422 | UnprocessableEntity |
| 500 | InternalError |

1. Bucket操作

1.1、Create Bucket

功能说明

Create Bucket 请求可以在指定账号下创建一个新的Bucket。

方法原型

func (c \*S3) CreateBucket(input \*CreateBucketInput) (\*CreateBucketOutput

, error)

参数说明

* input:类型是个struct，具体定义属性如下:

type CreateBucketInput struct {

ACL \*string `location:"header" locationName:"x-amz-acl" type:"\*string" enum:"BucketCannedACL"`

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

ObjectLockEnabledForBucket \*bool `location:"header" locationName:"x-amz-bucket-object-lock-enabled" type:"boolean"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| ACL | Bucket的ACL配，可选类型有private、public-read、public-read-write和authenticated-read | \*string | 否 |
| Bucket | Bucket的名称 | \*string | 是 |
| ObjectLockEnabledForBucket | 开启关闭新创建Bucket的Object锁定功能 | \*bool | 否 |

返回结果及说明

* CreateBucketOutPut:类型是个struct，具体定义属性如下:

type CreateBucketOutput struct {

Location \*string `location:"header" locationName:"Location" type:"\*string"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| Location | Bucket所在位置 | \*string |

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

input := &s3.CreateBucketInput{

Bucket: aws.\*string("bucket-name"),

ObjectLockEnabledForBucket: aws.Bool(false),

ACL: aws.\*string("private"),

}

result, err := svc.CreateBucket(input)

if err != nil {

fmt.Println("request fail")

fmt.Println(err)

return

}

fmt.Println("request success")

fmt.Println(result)

}

1.2、Delete Bucket

功能说明

Delete Bucket 请求可以在指定账号下删除 Bucket，删除之前要求 Bucket 为空。

方法原型

func (c \*S3) DeleteBucket(input \*DeleteBucketInput)

参数说明

* input:类型是个DeleteBucketInput，具体定义属性如下:

type DeleteBucketInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | 需要删除的Bucket名称 | \*string | 是 |

返回结果说明

* 本接口调用成功返回为DeleteBucketOutput，没有需要处理的字段。失败返回awserr.Error。

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk

, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

input := &s3.DeleteBucketInput{

Bucket: aws.\*string("go-sdk-bucket"),

}

result, err := svc.DeleteBucket(input)

if err != nil {

fmt.Println("request fail")

fmt.Println(err)

return

}

fmt.Println("request success")

fmt.Println(result)

}

1.3、Head Bucket

功能说明

Head Bucket 请求可以判断某个Bucket是否存在或者是否有权限访问该Bucket(其他用户名下Bucket)。

方法原型

func (c \*S3) HeadBucket(input \*HeadBucketInput) (\*HeadBucketOutput, error)

参数说明

* input:类型是个truct，具体定义属性如下:

type HeadBucketInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | 需要删除的Bucket名称 | \*string | 是 |

返回结果说明

* HeadBucketOutput:类似是个Struct，具体定义属性如下:

type HeadBucketOutput struct {

}

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

input := &s3.HeadBucketInput{

Bucket: aws.\*string("bucket-name"),

}

result, err := svc.HeadBucket(input)

if err != nil {

fmt.Println("requeset fail ")

fmt.Println(err)

return

}

fmt.Println("requeset success ")

fmt.Println(result)

}

1. 4、List Bucket

功能说明

List Bucket 可以列出请求用户拥有的所有的bucket列表。

方法原型

func (c \*S3) ListBuckets(input \*ListBucketsInput) (\*ListBucketsOutput, error)

参数说明

* input: 类型为ListBucketsInput，没有要指定的字段。

返回结果说明

* ListBucketsOutput ： 类型是 struct，具体字段如下

type ListBucketsOutput struct {

Buckets []\*Bucket `locationNameList:"Bucket" type:"list"`

Owner \*Owner `type:"structure"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| Owner | 列出的bucket的属主 | \*Owner |
| Buckets | bucket的列表 | []\*Bucket |

* Bucket: 类型是一个struct，具体字段如下

type Bucket struct {

CreationDate \*time.Time `type:"timestamp"`

Name \*string `type:"\*string"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| Name | Bucket名称 | \*string |
| CreationDate | bucket的创建时间 | Time |

* Owner: 类型是一个struct，具体字段如下

type Owner struct {

DisplayName \*string `type:"\*string"`

ID \*string `type:"\*string"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| DisplayName | Owner 的用户名 | \*string |
| ID | Owner的 uid | \*string |

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

result, err := svc.ListBuckets(&s3.ListBucketsInput{})

if err != nil {

fmt.Println("request fail")

fmt.Println(err)

return

}

fmt.Println("request success")

fmt.Println(result)

}

1.5、List Bucket Object

功能说明

List Bucket Object请求可以列出该 Bucekt 下部分或者所有Object，发起该请求需要拥有 Read 权限。

方法原型

func (c \*S3) ListObjects(input \*ListObjectsInput) (\*ListObjectsOutput,

error)

参数说明

* input: ListObjectsInput, 类型是 struct，字段如下

type ListObjectsInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

Delimiter \*string `location:"query\*string" locationName:"delimiter" type:"\*string"`

Marker \*string `location:"query\*string" locationName:"marker" type:"\*string"`

MaxKeys \*int64 `location:"query\*string" locationName:"max-keys" type:"integer"`

Prefix \*string `location:"query\*string" locationName:"prefix" type:"\*string"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket的名称 | \*string | 是 |
| Delimiter | 对文件名称进行分组的字符 | \*string | 否 |
| Marker | 本次list操作的起始点 | \*string | 否 |
| MaxKeys | 一次返回keys的最大数目（默认值和上限为1000） | \*int64 | 否 |
| Prefix | 设置返回的key的前缀 | \*string | 否 |

返回结果说明

* ListObjectsOutput 类型是 struct， 具体定义字段如下

type ListObjectsOutput struct {

CommonPrefixes []\*CommonPrefix `type:"list" flattened:"true"`

Contents []\*Object `type:"list" flattened:"true"`

Delimiter \*string `type:"\*string"`

IsTruncated \*bool `type:"boolean"`

Marker \*string `type:"\*string"`

MaxKeys \*int64 `type:"integer"`

Name \*string `type:"\*string"`

NextMarker \*string `type:"\*string"`

Prefix \*string `type:"\*string"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| CommonPrefixes | 当指定了delimiter和Prefix 时，获取根据Prefix和delimieter得到的公共前缀集合，例如指定Prefix为 “notes/” delimiter为 “/”,则 notes/summer/july,和notes/summer/august 折叠为一个CommonPrefix “notes/summer/”。若指定MaxKey，则折叠后，CommonPrefix只占一个计数；其中CommonPrefix.Prefix可获取到公共前缀 | []\*CommonPrefix |
| Contents | 返回对象的列表 | []\*Object |
| Delimiter | 请求时指定的delimiter | \*string |
| IsTruncated | 返回结果是否截断 | \*bool |
| Marker | 若请求时设置了Marker,在返回结果中将包含该Marker | \*string |
| MaxKeys | 获取指定的MaxKeys | \*int64 |
| Name | Bucket 的名称 | \*string |
| NextMarker | 若结果被截断，则给出分页所需的Marker· | \*string |
| Prefix | 获取请求时指定的Prefix | \*string |

* Object是个 struct， 具体定义字段如下

type Object struct {

ETag \*string `type:"\*string"`

Key \*string `min:"1" type:"\*string"`

LastModified \*time.Time `type:"timestamp"`

Owner \*Owner `type:"structure"`

Size \*int64 `type:"integer"`

StorageClass \*string `type:"\*string" enum:"ObjectStorageClass"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| ETag | 对象的ETag | \*string |
| Key | 对象的名称 | \*string |
| LastModified | 对象的修改时间 | \*Time |
| Owner | 对象是属主信息 | \*Owner |
| Size | 对象大小（byte） | \*int64 |
| StorageClass | 对象的存储级别 | \*string |

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

input := &s3.ListObjectsInput{

Bucket: aws.\*string("go-sdk-bucket"),

}

result, err := svc.ListObjects(input)

if err != nil {

fmt.Println("request fail")

fmt.Println(err)

return

}

fmt.Println("request success")

fmt.Println(result)

}

1.6、Put Bucket Policy

功能说明

Put Bucket Policy请求用于为ZOS S3 bucket设置桶策略。

方法原型

func (c \*S3) PutBucketPolicy(input \*PutBucketPolicyInput) (\*PutBucketPolicyOutput, error)

参数说明

* input:类型是个struct，定义如下:

type PutBucketPolicyInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

Policy \*string `type:"\*string" required:"true"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket名称 | \*string | 是 |
| Policy | 给Bucket设置的策略 | \*string | 是 |

Bucket Policy各字段描述如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **字段** | **描述** | **类型** | **是否必须** |
| Version | 保持与Amazon S3一致，当前支持"2012-10-17" | \*string | 否 |
| Id | 桶策略ID，桶策略的唯一标识 | \*string | 否 |
| Statement | 桶策略描述，定义完整的权限控制。每条桶策略的Statement可由多条描述组成，每条描述是一个dict，每条描述可包含以下字段：  Sid  Effect  Principal  Action  ReSource  Condition | list | 是 |
| Sid | 本条桶策略描述的ID | \*string | 否 |
| Effect | 桶策略的效果，即指定本条桶策略描述的权限是接受请求还是拒绝请求。  接受请求：配置为“Allow”，  拒绝请求：配置为“Deny” | \*string | 是 |
| Principal | 被授权人，即指定本条桶策略描述所作用的用户，支持通配符“\*”，表示所有用户。当对某个user进行授权时，Principal格式为"AWS": "arn:aws:s3:::user/userId" | dict | 否 |
| Action | 操作，即指定本条桶策略描述所作用的ZOS操作。以列表形式表示，可配置多条操作，以逗号间隔。支持通配符”\*“，表示该资源能进行的所有操作。常用的Action有"s3:GetObject"，"s3:GetObjectAcl"，"s3:PutObject"，  "s3:PutObjectAcl"等 | list | 否 |
| Condition | 条件语句，指定本条桶策略所限制的条件。可以通过Condition对ZOS资源设置防盗链，形如：  "Condition": {"\*stringEquals":{"aws:Referer":["www.example.com"]}，此时如果Effect为“Allow”，则允许来自"www.example.com"的请求；如果为“Deny”，则拒绝。 | dict | 否 |

返回结果说明

* 正常输出类型为PutBucketPolicyOutput，否则返回error，PutBucketPolicyOutput是struct类型，定义如下:

type PutBucketPolicyOutput struct {

// contains filtered or unexported fields

}

示例

package main

import (

"fmt"

"encoding/json"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

bucket\_name := "bucket1"

readOnlyPolicy := map[\*string]interface{}{

"Version": "2012-10-17",

"Statement": []map[\*string]interface{}{

{

"Sid": "AddPerm",

"Effect": "Allow",

"Principal": "\*",

"Action": []\*string{

"s3:GetObject",

},

"Resource": []\*string{

"arn:aws:s3:::" + bucket\_name + "/\*",

},

},

},

}

policy, err := json.Marshal(readOnlyPolicy)

if err != nil {

fmt.Println("bucket policy convert to json failed")

fmt.Println(err)

return

}

var bucket \*string = &bucket\_name

\_, err = svc.PutBucketPolicy(&s3.PutBucketPolicyInput{

Bucket: bucket,

Policy: aws.\*string(\*string(policy)),

})

if err != nil {

fmt.Println("Got an error setting bucket policy:")

fmt.Println(err)

return

} else {

fmt.Println("Set the bucket policy successfully")

}

}

1.7、Get Bucket Policy

功能说明

Get Bucket Policy请求为获取设置在一个bucket上的策略

方法原型

func (c \*S3) GetBucketPolicy(input \*GetBucketPolicyInput) (\*GetBucketPolicyOutput, error)

参数说明

* input:类型是个struct，定义如下:

type GetBucketPolicyInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"string" required:"true"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket名称 | \*string | 是 |

返回结果说明

* 正常输出类型为GetBucketPolicyOutput，否则返回error，GetBucketPolicyOutput是struct类型，定义如下:

type GetBucketPolicyOutput struct {

Policy \*string `type:"\*string"`

}

|  |  |  |
| --- | --- | --- |
| **成员名称** | **描述** | **类型** |
| Policy | 返回的Policy策略 | \*string |

示例

package main

import (

"fmt"

"bytes"

"encoding/json"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

bucket\_name := "bucket1"

var bucket \*string = &bucket\_name

result, err := svc.GetBucketPolicy(&s3.GetBucketPolicyInput{

Bucket: bucket,

})

if err != nil {

fmt.Println("Got an error when getting bucket policy:")

fmt.Println(err)

return

} else {

out := bytes.Buffer{}

policyPtr := aws.\*stringValue(result.Policy)

err = json.Indent(&out, []byte(policyPtr), "", " ")

if err != nil {

fmt.Println("parse bucket policy failed:")

fmt.Println(err)

return

}

fmt.Println("Policy:")

fmt.Println(out.\*string())

}

}

1.8、Delete Bucket Policy

功能说明

Delete Bucket Policy请求为删除设置在某个bucket上的策略

方法原型

func (c \*S3) DeleteBucketPolicy(input \*DeleteBucketPolicyInput) (\*DeleteBucketPolicyOutput, error)

参数说明

* input:类型是个struct，定义如下:

type DeleteBucketPolicyInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"string" required:"true"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket名称 | \*string | 是 |

返回结果说明

* 正常输出类型为DeleteBucketPolicyOutput，否则返回error，DeleteBucketPolicyOutput是struct类型，定义如下:

type DeleteBucketPolicyOutput struct {

// contains filtered or unexported fields

}

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

bucket\_name := "bucket1"

var bucket \*string = &bucket\_name

\_, err := svc.DeleteBucketPolicy(&s3.DeleteBucketPolicyInput{

Bucket: bucket,

})

if err != nil {

fmt.Println("Got an error when deleting bucket policy:")

fmt.Println(err)

return

} else {

fmt.Println("Removed the bucket policy successfully")

}

}

1.9、Put Bucket ACL

功能说明

设置Bucket的ACL，控制对Bucket的访问权限。该操作需要用户具有WRITE\_ACP权限。

有三种方式设置ACL，三种方式不可同时使用，每次只能给一种参数赋值。其中，通过ACL参数方式进行操作，是设置预定义的固定的ACL，不能针对特定用户进行授权，且该参数实现的效果，也可以借由另外两种方式实现，该参数使用请求头进行传递；AccessControlPolicy参数方式和Grant\*参数方式则可以针对特定用户进行授权，AccessControlPolicy方式通过请求体传递，而Grant\*方式通过请求头传递。三种方式都会覆盖原有ACL属性，包括桶所有者自身的权限，如需保留原有ACL属性，应将需要保留的原ACL添加到本次操作的授权中（ACL参数方式会默认将桶所有者权限设为FULL\_CONTROL，而另外两种方式则不会保留任何原ACL属性）。

方法原型

func (c \*S3) PutBucketAcl(input \*PutBucketAclInput) (\*PutBucketAclOutput, error)

参数说明

* input:类型是个struct，具体定义属性如下:

type PutBucketAclInput struct {

ACL \*string `location:"header" locationName:"x-amz-acl" type:"\*string" enum:"BucketCannedACL"`

AccessControlPolicy \*AccessControlPolicy `locationName:"AccessControlPolicy" type:"structure" xmlURI:"http://s3.amazonaws.com/doc/2006-03-01/"`

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

GrantFullControl \*string `location:"header" locationName:"x-amz-grant-full-control" type:"\*string"`

GrantRead \*string `location:"header" locationName:"x-amz-grant-read" type:"\*string"`

GrantReadACP \*string `location:"header" locationName:"x-amz-grant-read-acp" type:"\*string"`

GrantWrite \*string `location:"header" locationName:"x-amz-grant-write" type:"\*string"`

GrantWriteACP \*string `location:"header" locationName:"x-amz-grant-write-acp" type:"\*string"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket的名称 | \*string | 是 |
| ACL | 预定义的固定ACL，  取值范围  s3.BucketCannedACLPrivate  s3.BucketCannedACLPublicRead  s3.BucketCannedACLPublicReadWrite  s3.BucketCannedACLAuthenticatedRead | \*string | ACL参数方式则必须，其他两种方式，则不能使用 |
| AccessControlPolicy | 包含授权列表和桶所有者参数 | \*AccessControlPolicy | 该方式下必须，其他两种方式下则不能使用 |
| GrantFullControl | 被授权用户可以对桶进行read, write, read ACP, and write ACP操作  以下Grant\*参数，格式都是”id=xxxx”或”emailAddress=xxxx”或者”uri=xxxx”以及他们的组合（用逗号连接） | \*string | 否 |
| GrantRead | 被授权用户可以对桶进行读操作，即list object | \*string | 否 |
| GrantWrite | 被授权用户可以对桶进行写操作，创建新的对象，删除或覆盖写属于自己的对象 | \*string | 否 |
| GrantReadACP | 被授权用户可以读取桶的ACL | \*string | 否 |
| GrantWriteACP | 被授权用户可以修改桶的ACL | \*string | 否 |

AccessControlPolicy:类型是个struct，具体定义属性如下:

type AccessControlPolicy struct {

Grants []\*Grant `locationName:"AccessControlList" locationNameList:"Grant" type:"list"`

Owner \*Owner `type:"structure"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Grants | 授权列表 | []\*Grant | 该方式下必须 |
| Owner | Bucket所有者 | \*Owner | 该方式下必须 |

Grant:类型是个struct，具体定义属性如下:

type Grant struct {

Grantee \*Grantee `type:"structure" xmlPrefix:"xsi" xmlURI:"http://www.w3.org/2001/XMLSchema-instance"`

Permission \*string `type:"\*string" enum:"Permission"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Grantee | 被授权用户 | \*Grantee | 该方式下必须 |
| Permission | 向被授权用户授予的权限，  取值范围  s3.PermissionFullControl  s3.PermissionWrite  s3.PermissionWriteAcp  s3.PermissionRead  s3.PermissionReadAcp | \*string | 该方式下必须 |

Grantee:类型是个struct，具体定义属性如下:

type Grantee struct {

DisplayName \*string `type:"\*string"`

EmailAddress \*string `type:"\*string"`

ID \*string `type:"\*string"`

Type \*string `locationName:"xsi:type" type:"\*string" xmlAttribute:"true" required:"true" enum:"Type"`

URI \*string `type:"\*string"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Type | 被授权用户类型，  取值范围  s3.TypeCanonicalUser  s3.TypeAmazonCustomerByEmail  s3.TypeGroup | \*string | 该方式下必须 |
| ID | 被授权用户ID | \*string | Type为'CanonicalUser'，则该字段必须 |
| DisplayName | 被授权用户display name | \*string | 否 |
| EmailAddress | 被授权用户邮箱 | \*string | 如果Type为'AmazonCustomerByEmail'，则该字段必须 |
| URI | 被授权组URI  取值范围为  所有用户：http://acs.amazonaws.com/groups/global/AllUsers  所有认证用户：http://acs.amazonaws.com/groups/global/AuthenticatedUsers | \*string | 如果Type为'Group'，则该字段必须 |

Owner:类型是个struct，具体定义属性如下:

type Owner struct {

DisplayName \*string `type:"\*string"`

ID \*string `type:"\*string"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| ID | Bucket所有者ID | \*string | 该方式下必须 |
| DisplayName | Bucket所有者display name | \*string | 否 |

返回结果说明

* 正常输出类型为PutBucketAclOutput，否则返回error，

PutBucketAclOutput是struct类型，定义如下:

type PutBucketAclOutput struct {

}

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

resp, err := svc.PutBucketAcl(&s3.PutBucketAclInput{

Bucket: aws.\*string("bucket-1"),

// ACL: aws.\*string(s3.BucketCannedACLPublicRead),

// GrantReadACP: aws.\*string("id=test-2,emailAddress=abc@abc.com"),

AccessControlPolicy: &s3.AccessControlPolicy{

Owner: &s3.Owner{

ID: aws.\*string("test-1"),

},

Grants: []\*s3.Grant{

{

Grantee: &s3.Grantee{

Type: aws.\*string(s3.TypeCanonicalUser),

ID: aws.\*string("test-2"),

},

Permission: aws.\*string(s3.PermissionFullControl),

},

{

Grantee: &s3.Grantee{

Type: aws.\*string(s3.TypeAmazonCustomerByEmail),

EmailAddress: aws.\*string("def@def.com"),

},

Permission: aws.\*string(s3.PermissionFullControl),

},

},

},

})

}

1.10、Get Bucket ACL

功能说明

Get Bucket ACL 接口用来获取 Bucket 的 ACL， 即存储桶（Bucket）的访问权限控制列表。该操作需要READ\_ACP权限。该功能返回的结果与Put Bucket ACL参数一致，但是需要注意的是，如果以邮箱类型授权，返回结果中将会以对应被授权用户ID形式出现，即Type不会是AmazonCustomerByEmail，而是CanonicalUser。

方法原型

func (c \*S3) GetBucketAcl(input \*GetBucketAclInput) (\*GetBucketAclOutput, error)

参数说明

* input:类型是个struct，定义如下:

type GetBucketAclInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket的名称 | \*string | 是 |

返回结果说明

* 正常输出类型为GetBucketAclOutput，否则返回error，

GetBucketAclOutput是struct类型，定义如下:

type GetBucketAclOutput struct {

Grants []\*Grant `locationName:"AccessControlList" locationNameList:"Grant" type:"list"`

// Container for the bucket owner's display name and ID.

Owner \*Owner `type:"structure"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| Grants | 授权列表 | []\*Grant |
| Owner | Bucket所有者 | \*Owner |

Grant:类型是个struct，具体定义属性如下:

type Grant struct {

Grantee \*Grantee `type:"structure" xmlPrefix:"xsi" xmlURI:"http://www.w3.org/2001/XMLSchema-instance"`

Permission \*string `type:"\*string" enum:"Permission"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| Grantee | 被授权用户 | \*Grantee |
| Permission | 被授权权限 | \*string |

Grantee:类型是个struct，具体定义属性如下:

type Grantee struct {

DisplayName \*string `type:"\*string"`

EmailAddress \*string `type:"\*string"`

ID \*string `type:"\*string"`

Type \*string `locationName:"xsi:type" type:"\*string" xmlAttribute:"true" required:"true" enum:"Type"`

URI \*string `type:"\*string"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| DisplayName | 被授权用户的展示名 | \*string |
| EmailAddress | 被授权用户邮箱 | \*string |
| ID | 被授权用户ID | \*string |
| Type | 被授权用户类型 | \*string |
| URI | 被授权组uri | \*string |

Owner:类型是个struct，具体定义属性如下:

type Owner struct {

DisplayName \*string `type:"\*string"`

ID \*string `type:"\*string"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| DisplayName | Owner 的展示名 | \*string |
| ID | Owner的 ID | \*string |

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

resp, err := svc.GetBucketAcl(&s3.GetBucketAclInput{

Bucket: aws.\*string("bucket-1"),

})

if err != nil {

fmt.Println("failed", err)

} else {

fmt.Println("success", resp)

}

}

1.11、Put Bucket Lifecycle Configuration

功能说明

Put Bucket Lifecycle Configuration接口用来写入 Bucket 的生命周期规则。

方法原型

func (c \*S3) PutBucketLifecycleConfiguration(input \*PutBucketLifecycleConfigurationInput) (\*PutBucketLifecycleConfigurationOutput, error)

参数说明

* input:类型\*s3.PutBucketLifecycleConfigurationInput，创建Bucket Lifecycle请求接口参数，定义的属性如下:

type PutBucketLifecycleConfigurationInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

LifecycleConfiguration \*BucketLifecycleConfiguration `locationName:"LifecycleConfiguration" type:"structure" xmlURI:"http://s3.amazonaws.com/doc/2006-03-01/"`

}

BucketLifecycleConfiguration 的定义方法如下：

type BucketLifecycleConfiguration struct {

Rules []\*LifecycleRule `locationName:"Rule" type:"list" flattened:"true" required:"true"`

}

LifecycleRule的定义方法如下：

type LifecycleRule struct {

AbortIncompleteMultipartUpload \*AbortIncompleteMultipartUpload `type:"structure"`

Expiration \*LifecycleExpiration `type:"structure"`

Filter \*LifecycleRuleFilter `type:"structure"`

ID \*string `type:"\*string"`

NoncurrentVersionExpiration \*NoncurrentVersionExpiration `type:"structure"`

NoncurrentVersionTransitions []\*NoncurrentVersionTransition `locationName:"NoncurrentVersionTransition" type:"list" flattened:"true"`

Prefix \*string `deprecated:"true" type:"\*string"`

//ExpirationStatus枚举值：“Enabled”|“Disabled”

Status \*string `type:"\*string" required:"true" enum:"ExpirationStatus"`

Transitions []\*Transition `locationName:"Transition" type:"list" flattened:"true"`

}

AbortIncompleteMultipartUpload的定义方法如下：

type AbortIncompleteMultipartUpload struct {

DaysAfterInitiation \*int64 `type:"integer"`

}

LifecycleExpiration的定义方法如下：

type LifecycleExpiration struct {

Date \*time.Time `type:"timestamp" timestampFormat:"iso8601"`

Days \*int64 `type:"integer"`

ExpiredObjectDeleteMarker \*bool `type:"boolean"`

}

LifecycleRuleFilter的定义方法如下：

type LifecycleRuleFilter struct {

And \*LifecycleRuleAndOperator `type:"structure"`

Prefix \*string `type:"\*string"`

Tag \*Tag `type:"structure"`

}

LifecycleRuleAndOperator的定义方法如下：

type LifecycleRuleAndOperator struct {

Prefix \*string `type:"\*string"`

Tags []\*Tag `locationName:"Tag" locationNameList:"Tag" type:"list" flattened:"true"`

}

Tag 的定义方法如下：

type Tag struct {

Key \*string `min:"1" type:"\*string" required:"true"`

Value \*string `type:"\*string" required:"true"`

}

NoncurrentVersionExpiration 的定义方法如下：

type NoncurrentVersionExpiration struct {

NoncurrentDays \*int64 `type:"integer"`

}

NoncurrentVersionTransition 的定义方法如下：

type NoncurrentVersionTransition struct {

NoncurrentDays \*int64 `type:"integer"`

//TransitionStorageClass的枚举值：“GLACIER”|“STANDARD\_IA”|“ONEZONE\_IA”|“INTELLIGENT\_TIERING”|“DEEP\_ARCHIVE”

StorageClass \*string `type:"\*string" enum:"TransitionStorageClass"`

}

Transtion的定义方法如下：

type Transition struct {

Date \*time.Time `type:"timestamp" timestampFormat:"iso8601"`

Days \*int64 `type:"integer"`

StorageClass \*string `type:"\*string" enum:"TransitionStorageClass"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | 要设置生命周期的存储桶名字 | \*string | 是 |
| LifecycleConfiguration | 生命周期规则的容器 | \*struct | 是 |
| Rules | 生命周期规则 | array | 是 |
| Expiration | 用日期或天数指定对象的过期时间 | \*struct | 否 |
| Date | 标识对象的过期日期,日期为ISO8601格式，必须为UTC午夜0时 | \*time.Time | Date与Days二选一 |
| Days | 标识对象受规则约束的天数 | \*integer | Date与Days二选一 |
| ID | 标识唯一的规则 | \*string | 否 |
| Filter | 过滤应用规则的对象 | \*struct | 否 |
| Prefix | 标识应用规则的对象前缀 | \*string | 否 |
| Tag | 应用规则到拥有指定标签的对象 | \*struct | 否 |
| Key | 标签的名称 | \*string | 否 |
| Value | 标签的值 | \*string | 否 |
| Status | 标识是否应用规则 ，可选值：Enabled, Disabled | \*string | 是 |
| Transtions | 标识对象何时转存到指定的Storage Class | array | 否 |
| StorageClass | 标识要转存储到哪种存储类别,如STANDARD/STANDARD\_IA/GLACIER | \*string | 如设置转存储规则，该字段必选 |
| NoncurrentVersionTransitions | 标识历史版本的转存储规则 | array | 否 |
| NoncurrentDays | 标识对象的历史版本受规则约束的天数 | \*int64 | 否 |
| NoncurrentVersionExpiration | 标识历史版本的过期规则 | \*struct | 否 |
| AbortIncompleteMultipartUpload | 标识清除未完成的分段上传 | \*struct | 否 |
| DaysAfterInitiation | 标识一次分段上传最多持续天数 | \*int64 | 否 |

返回结果说明

返回结果无有意义字段，无需处理。

示例

package main

import (

"time"

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/awserr"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

localtime,\_ := time.Parse(time.RFC3339,"2020-06-20T00:00:00Z")

lc := &s3.PutBucketLifecycleConfigurationInput{

Bucket: aws.\*string("verbuck"),

LifecycleConfiguration: &s3.BucketLifecycleConfiguration {

Rules: []\*s3.LifecycleRule{

{

Expiration: &s3.LifecycleExpiration{

Date: aws.Time(localtime),

},

Filter: &s3.LifecycleRuleFilter{

Prefix: aws.\*string("test"),

},

ID: aws.\*string("TestOnly"),

NoncurrentVersionExpiration: &s3.NoncurrentVersionExpiration{

NoncurrentDays: aws.Int64(365),

},

Status: aws.\*string("Enabled"),

},

},

},

}

\_, err := svc.PutBucketLifecycleConfiguration(lc)

if err != nil {

if aerr, ok := err.(awserr.Error); ok {

switch aerr.Code() {

default:

fmt.Println(aerr.Error())

}

} else {

fmt.Println(err.Error())

}

}

}

1.12、Get Bucket Lifecycle Configuration

功能说明

Get Bucket Lifecycle Configuration接口用来获取 Bucket 的生命周期规则。

方法原型

func (c \*S3) GetBucketLifecycleConfiguration(input \*GetBucketLifecycleConfigurationInput) (\*GetBucketLifecycleConfigurationOutput, error)

参数说明

* input:类型\*s3.GetBucketLifecycleConfigurationInput，获取Bucket Lifecycle请求接口参数，定义的方法如下:

type GetBucketLifecycleConfigurationInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | 指定存储桶名称 | \*string | 是 |

返回结果说明

{

Rules: [

{

Expiration: {

Date: \*time.Time,

Days: \*int64,

ExpiredObjectDeleteMarker: \*bool

},

ID: \*string,

Prefix: \*string,

Filter: {

Prefix: \*string,

Tag: {

Key: \*string,

Value: \*string

},

And: {

Prefix: \*string,

Tags: [

{

Key: \*string,

Value: \*string

}

]

}

},

Status: \*string,

Transitions: [

{

Date: \*time.Time,

Days: \*int64,

StorageClass: \*string

}

],

NoncurrentVersionTransitions: [

{

NoncurrentDays: \*int64,

StorageClass: \*string

},

],

NoncurrentVersionExpiration: {

NoncurrentDays: \*int64

},

AbortIncompleteMultipartUpload: {

DaysAfterInitiation: \*int64

}

}

]

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| Rules | 生命周期规则 | array |
| Expiration | 对象的过期规则 | \*struct |
| Date | 对象的过期日期 | \*time.Time |
| Days | 对象受规则约束的天数 | \*int64 |
| ID | 标识唯一的规则 | \*string |
| Filter | 过滤应用规则的对象 | \*struct |
| Prefix | 标识应用规则的对象前缀 | \*string |
| Tag | 应用规则到拥有指定标签的对象 | \*struct |
| Key | 标签的名称 | \*string |
| Value | 标签的值 | \*string |
| Status | 标识是否应用规则 ，可选值：Enabled, Disabled | \*string |
| Transtions | 标识对象何时转存到指定的Storage Class | array |
| StorageClass | 标识要转存储到哪种存储类别,如STANDARD/STANDARD\_IA/GLACIER | \*string |
| NoncurrentVersionTransitions | 标识历史版本的转存储规则 | array |
| NoncurrentDays | 标识对象的历史版本受规则约束的天数 | \*int64 |
| NoncurrentVersionExpiration | 标识历史版本的过期规则 | \*struct |
| AbortIncompleteMultipartUpload | 标识清除未完成的分段上传 | \*struct |
| DaysAfterInitiation | 标识一次分段上传最多持续天数 | \*int64 |

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/awserr"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

input := &s3.GetBucketLifecycleConfigurationInput{

Bucket: aws.\*string("verbuck"),

}

result, err := svc.GetBucketLifecycleConfiguration(input)

if err != nil {

if aerr, ok := err.(awserr.Error); ok {

switch aerr.Code() {

default:

fmt.Println(aerr.Error())

}

} else {

fmt.Println(err.Error())

}

}

fmt.Println(result)

}

1.13、Delete Bucket Lifecycle

功能说明

Delete Bucket Lifecycle 接口用来删除 Bucket 的生命周期规则。

方法原型

func (c \*S3) DeleteBucketLifecycle(input \*DeleteBucketLifecycleInput) (\*DeleteBucketLifecycleOutput, error)

参数说明

* input:类型\*s3.DeleteBucketLifecycleInput，删除Bucket Lifecycle请求接口参数，定义的方法如下:

type DeleteBucketLifecycleInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | 指定存储桶名称 | \*string | 是 |

返回结果说明

返回结果无有意义字段，无需处理。

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/awserr"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

input := &s3.DeleteBucketLifecycleInput{

Bucket: aws.\*string("verbuck"),

}

result, err := svc.DeleteBucketLifecycle(input)

if err != nil {

if aerr, ok := err.(awserr.Error); ok {

switch aerr.Code() {

default:

fmt.Println(aerr.Error())

}

} else {

fmt.Println(err.Error())

}

}

fmt.Print(result)

}

1.14、Put Bucket Website

功能说明

调用PutBucketWebsite接口将存储空间（Bucket）设置成静态网站托管模式并设置跳转规则（RoutingRule）

方法原型

func (c \*[S3](#S3)) PutBucketWebsite(input \*[PutBucketWebsiteInput](#PutBucketWebsiteInput)) (\*[PutBucketWebsiteOutput](#PutBucketWebsiteOutput), error)

参数说明

type PutBucketWebsiteInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

WebsiteConfiguration \*WebsiteConfiguration

`locationName:"WebsiteConfiguration" type:"structure" required:"true" xmlURI:"http://s3.amazonaws.com/doc/2006-03-01/"`

}

type WebsiteConfiguration struct {

ErrorDocument \*ErrorDocument `type:"structure"`

IndexDocument \*IndexDocument `type:"structure"`

RedirectAllRequestsTo \*RedirectAllRequestsTo `type:"structure"`

RoutingRules []\*RoutingRule `locationNameList:"RoutingRule" type:"list"`

}

type ErrorDocument struct {

Key \*string `min:"1" type:"\*string" required:"true"`

}

type IndexDocument struct {

Suffix \*string `type:"\*string" required:"true"`

}

type RedirectAllRequestsTo struct {

HostName \*string `type:"\*string" required:"true"`

Protocol \*string `type:"\*string" enum:"Protocol"`

}

type RoutingRule struct {

Condition \*Condition `type:"structure"`

Redirect \*Redirect `type:"structure" required:"true"`

}

type Condition struct {

HttpErrorCodeReturnedEquals \*string `type:"\*string"`

KeyPrefixEquals \*string `type:"\*string"`

}

type Redirect struct {

HostName \*string `type:"\*string"`

HttpRedirectCode \*string `type:"\*string"`

Protocol \*string `type:"\*string" enum:"Protocol"`

ReplaceKeyPrefixWith \*string `type:"\*string"`

ReplaceKeyWith \*string `type:"\*string"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| PutBucketWebsiteInput | 配置静态网站请求参数据容器 | struct | 是 |
| Bucket | bucket 名称 | \*string | 是 |
| WebsiteConfiguration | 请求的配置信息的容器 | struct | 是 |
| ErrorDocument | 错误文档配置 | struct | 否 |
| Key | 指定通用错误文档的对象键，当发生错误且未命中重定向规则中的错误码重定向时，将返回该对象键的内容 | \*string | 是 |
| IndexDocument | 索引文档配置 | struct | 否 |
| Suffix | 指定索引文档的对象键后缀。例如指定为index.html，那么当访问到存储桶的根目录时，会自动返回 index.html 的内容，或者当访问到article/目录时，会自动返回 article/index.html的内容 | \*string | 是 |
| RedirectAllRequestsTo | 重定向所有请求配置，使用了重定向规则就不能配置其他规则。 | struct | 否 |
| HostName | 要重定向的主机名 | \*string | 是 |
| Protocol | 重定向时使用的协议，默认使用原请求的协议 | \*string | 否 |
| RoutingRules | 重定向规则配置 | struct | 否 |
| Condition | 重定向规则的条件配置 | struct |  |
| HttpErrorCodeReturnedEquals | 指定重定向规则的错误码匹配条件，只支持配置4XX返回码，例如403或404 | \*string | 当condition配置后，HttpErrorCodeReturnedEquals 和KeyPrefixEquals 两者只能配置一个。 |
| KeyPrefixEquals | 指定重定向规则的对象键前缀匹配条件 | \*string | 当condition配置后，HttpErrorCodeReturnedEquals 和KeyPrefixEquals 两者只能配置一个。 |
| Redirect | 重定向格则容器，可以配置规则重定向其他主机、页面或其他协议，当发生错误时，也可以配置错误码。 | struct | RoutingRules中的必要配置。 |

|  |  |  |  |
| --- | --- | --- | --- |
| HostName | 重定向机器名 | \*string | 否 |
| HttpRedirectCode | http返回码规则， | \*string | 否 |
| Protocol | 重定向请求要用的协议，默认使用原请求所是应用的协议。 | \*string | 否 |
| ReplaceKeyPrefixWith | 指定重定向规则的具体重定向目标的对象键，替换方式为替换原始请求中所匹配到的前缀部分，仅可在 Condition 为 KeyPrefixEquals 时设置 | \*string | ReplaceKeyWith 与 ReplaceKeyPrefixWith 必选其一 |
| ReplaceKeyWith | 指定重定向规则的具体重定向目标的对象键，替换方式为替换整个原始请求的对象键 | \*string | ReplaceKeyWith 与 ReplaceKeyPrefixWith 必选其一 |

返回结果说明

type PutBucketWebsiteOutput struct {

// contains filtered or unexported fields

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| PutBucketWebsiteOutput | 返回的结果，无需特殊处理 | struct |

示例

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

"github.com/aws/aws-sdk-go/aws/awserr"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

input := &s3.PutBucketWebsiteInput{

Bucket: aws.\*string("rgwuser01-testbucket03"),

WebsiteConfiguration: &s3.WebsiteConfiguration{

ErrorDocument: &s3.ErrorDocument{

Key: aws.\*string("error.html"),

},

IndexDocument: &s3.IndexDocument{

Suffix: aws.\*string("index.html"),

},

RedirectAllRequestsTo:&s3.RedirectAllRequestsTo{

HostName:aws.\*string("centos"),

Protocol:aws.\*string("http"),

},

},

}

result, err := svc.PutBucketWebsite(input)

if err != nil {

if aerr, ok := err.(awserr.Error); ok {

switch aerr.Code() {

default:

fmt.Println(aerr.Error())

}

} else {

// Print the error, cast err to awserr.Error to get the Code and

// Message from an error.

fmt.Println(err.Error())

}

return

}

fmt.Println(result)

}

1.15、Get Bucket Website

功能说明

GET Bucket website 请求用于查询与存储桶关联的静态网站配置信息。

方法原型

func (c \*[S3](#S3)) GetBucketWebsite(input \*[GetBucketWebsiteInput](#GetBucketWebsiteInput)) (\*[GetBucketWebsiteOutput](#GetBucketWebsiteOutput), error)

参数说明

type GetBucketWebsiteInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| [GetBucketWebsiteInput](#GetBucketWebsiteOutput) | 获取配置静态网站请求参数据容器 | struct | 是 |
| Bucket | 获取配置静态网站关联的桶名称 | \*string | 是 |

返回结果说明

type GetBucketWebsiteOutput struct {

ErrorDocument \*ErrorDocument `type:"structure"`

IndexDocument \*IndexDocument `type:"structure"`

RedirectAllRequestsTo \*RedirectAllRequestsTo `type:"structure"`

RoutingRules []\*RoutingRule `locationNameList:"RoutingRule" type:"list"`

}

type ErrorDocument struct {

Key \*string `min:"1" type:"\*string" required:"true"`

}

type IndexDocument struct {

Suffix \*string `type:"\*string" required:"true"`

}

type RedirectAllRequestsTo struct {

HostName \*string `type:"\*string" required:"true"`

Protocol \*string `type:"\*string" enum:"Protocol"`

}

type RoutingRule struct {

Condition \*Condition `type:"structure"`

Redirect \*Redirect `type:"structure" required:"true"`

}

type Condition struct {

HttpErrorCodeReturnedEquals \*string `type:"\*string"`

KeyPrefixEquals \*string `type:"\*string"`

}

type Redirect struct {

HostName \*string `type:"\*string"`

HttpRedirectCode \*string `type:"\*string"`

Protocol \*string `type:"\*string" enum:"Protocol"`

ReplaceKeyPrefixWith \*string `type:"\*string"`

ReplaceKeyWith \*string `type:"\*string"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| GetBucketWebsiteOutput | 获取静态网站配置信息返回结果的容器 | struct |
| ErrorDocument | 错误文档配置容器 | struct |
| Key | 指定通用错误文档的对象键，当发生错误且未命中重定向规则中的错误码重定向时，将返回该对象键的内容 | \*string |
| IndexDocument | 索引文档配置容器 | struct |
| Suffix | 指定索引文档的对象键后缀。例如指定为index.html，那么当访问到存储桶的根目录时，会自动返回 index.html 的内容，或者当访问到article/目录时，会自动返回 article/index.html的内容 | \*string |
| RedirectAllRequestsTo | 重定向所有请求配置容器，该规则与其他规则互斥，也就是说使用了重定向规则就不能配置其他规则。 | struct |
| HostName | 要重定向的主机名 | \*string |
| Protocol | 重定向时使用的协议，默认使用原请求的协议 | \*string |
| RoutingRules | 重定向规则配置容器 | struct |
| Condition | 重定向规则的条件配置 | struct |
| HttpErrorCodeReturnedEquals | 指定重定向规则的错误码匹配条件，只支持配置4XX返回码，例如403或404 | \*string |
| KeyPrefixEquals | 指定重定向规则的对象键前缀匹配条件 | \*string |
| Redirect | 重定向格则容器，可以配置规则重定向其他主机、页面或其他协议，当发生错误时，也可以配置错误码。 | struct |
| HostName | 重定向机器名 | \*string |
| HttpRedirectCode | http返回码规则， | \*string |
| Protocol | 重定向请求要用的协议，默认使用原请求所是应用的协议。 | \*string |
| ReplaceKeyPrefixWith | 指定重定向规则的具体重定向目标的对象键，替换方式为替换原始请求中所匹配到的前缀部分，仅可在 Condition 为 KeyPrefixEquals 时设置 | \*string |
| ReplaceKeyWith | 指定重定向规则的具体重定向目标的对象键，替换方式为替换整个原始请求的对象键 | \*string |

示例

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

"github.com/aws/aws-sdk-go/aws/awserr"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

input := &s3.GetBucketWebsiteInput{

Bucket: aws.\*string("rgwuser01-testbucket03"),

}

result, err := svc.GetBucketWebsite(input)

if err != nil {

if aerr, ok := err.(awserr.Error); ok {

switch aerr.Code() {

default:

fmt.Println(aerr.Error())

}

} else {

// Print the error, cast err to awserr.Error to get the Code and

// Message from an error.

fmt.Println(err.Error())

}

return

}

fmt.Println(result)

}

1.16、Delete Bucket Website

功能说明

DELETE Bucket website 请求用于删除存储桶中的静态网站配置。

方法原型

func (c \*[S3](#S3)) DeleteBucketWebsite(input \*[DeleteBucketWebsiteInput](#DeleteBucketWebsiteInput)) (\*[DeleteBucketWebsiteOutput](#DeleteBucketWebsiteOutput), error)

参数说明

type DeleteBucketWebsiteInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | 要删除静态网站配置的存储桶名称 | \*string | 是 |

返回结果说明

type DeleteBucketWebsiteOutput struct {

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| DeleteBucketWebsiteOutput | 操作结果包含过滤的或者未导出的字段, | struct |

示例

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

"github.com/aws/aws-sdk-go/aws/awserr"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

input := &s3.DeleteBucketWebsiteInput{

Bucket: aws.\*string("rgwuser01-testbucket03"),

}

result, err := svc.DeleteBucketWebsite(input)

if err != nil {

if aerr, ok := err.(awserr.Error); ok {

switch aerr.Code() {

default:

fmt.Println(aerr.Error())

}

} else {

// Print the error, cast err to awserr.Error to get the Code and

// Message from an error.

fmt.Println(err.Error())

}

return

}

fmt.Println(result)

}

1.17、Put Bucket Request Payment

功能说明

设置bucket的请求支付配置。默认情况下，bucket所有者为bucket的下载 付费。此配置参数使bucket所有者能够指定请求下载的人为下载付费。

方法原型

func (c \*S3) PutBucketRequestPayment(input \*PutBucketRequestPaymentInput)(\*PutBucketRequestPaymentOutput, error)

参数说明

* input:类型是个struct，定义如下:

type PutBucketRequestPaymentInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"string" required:"true"`

RequestPaymentConfiguration \*RequestPaymentConfiguration `locationName:"RequestPaymentConfiguration" type:"structure" required:"true"

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket名称 | \*string | 是 |
| RequestPaymentConfiguration | 设置在bucket上的请求支付配置 | structure | 是 |

RequestPaymentConfiguration也是个struct类型，定义如下：

type RequestPaymentConfiguration struct {

Payer \*string `type:"\*string" required:"true" enum:"Payer"`

}

返回结果说明

* 正常输出类型为PutBucketRequestPaymentOutput，否则返回error，PutBucketRequestPaymentOutput是struct类型，定义如下:

type PutBucketRequestPaymentOutput struct {

// contains filtered or unexported fields

}

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

input := &s3.PutBucketRequestPaymentInput{

Bucket: aws.\*string("bucket1"),

RequestPaymentConfiguration: &s3.RequestPaymentConfiguration{

Payer: aws.\*string("Requester"),

},

}

result, err := svc.PutBucketRequestPayment(input)

if err != nil {

fmt.Println(err)

return

}

fmt.Println(result)

}

1.18、Get Bucket Request Payment

功能说明

返回bucket的请求支付配置。

方法原型

func (c \*S3) GetBucketRequestPayment(input

\*GetBucketRequestPaymentInput) (\*GetBucketRequestPaymentOutput, error)

参数说明

* input:类型是个struct，定义如下:

type GetBucketRequestPaymentInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"string" required:"true"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket名称 | \*string | 是 |

返回结果说明

* 正常输出类型为GetBucketRequestPaymentOutput，否则返回error，GetBucketRequestPaymentOutput是struct类型，定义如下:

type GetBucketRequestPaymentOutput struct {

Payer \*string `type:"\*string" enum:"Payer"`

}

|  |  |  |
| --- | --- | --- |
| **成员名称** | **描述** | **类型** |
| Payer | 返回的具体的付费者 | \*string |

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

input := &s3.GetBucketRequestPaymentInput{

Bucket: aws.\*string("bucket1"),

}

result, err := svc.GetBucketRequestPayment(input)

if err != nil {

fmt.Println(err)

return

}

fmt.Println(result)

}

1.19、Put Bucket Tagging

功能说明

为指定的Bucket设置标签。一个Bucket最多设置50个标签。该操作需要s3:PutBucketTagging权限，桶的所有者默认拥有该权限。该操作会覆盖原有标签。

方法原型

func (c \*S3) PutBucketTagging(input \*PutBucketTaggingInput) (\*PutBucketTaggingOutput, error)

参数说明

* input:类型是个struct，定义如下:

type PutBucketTaggingInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

Tagging \*Tagging `locationName:"Tagging" type:"structure" required:"true" xmlURI:"http://s3.amazonaws.com/doc/2006-03-01/"`

}

* Tagging:类型是个struct，定义如下:

type Tagging struct {

TagSet []\*Tag `locationNameList:"Tag" type:"list" required:"true"`

}

* Tag:类型是个struct，定义如下:

type Tag struct {

Key \*string `min:"1" type:"\*string" required:"true"`

Value \*string `type:"\*string" required:"true"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Key | key最大128字节 | \*string | 是 |
| Value | value最大256字节 | \*string | 是 |

返回结果说明

* 正常输出类型为PutBucketTaggingOutput，否则返回error，

PutBucketTaggingOutput是struct类型，定义如下:

type PutBucketTaggingOutput struct {

}

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

resp, err := svc.PutBucketTagging(&s3.PutBucketTaggingInput{

Bucket: aws.\*string("bucket-1"),

Tagging: &s3.Tagging{

TagSet: []\*s3.Tag{

{

Key: aws.\*string("key1"),

Value: aws.\*string("val1"),

},

},

},

})

if err != nil {

fmt.Println("failed", err)

} else {

fmt.Println("success", resp)

}

}

1.20、Get Bucket Tagging

功能说明

获取指定BUCKET的标签。该操作需要s3:GetBucketTagging权限，桶的拥有者默认具有该权限。

方法原型

func (c \*S3) GetBucketTagging(input \*GetBucketTaggingInput) (\*GetBucketTaggingOutput, error)

参数说明

* input:类型是个struct，定义如下:

type GetBucketTaggingInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket的名称 | \*string | 是 |

返回结果说明

* 正常输出类型为GetBucketTaggingOutput，否则返回error，

GetBucketTaggingOutput是struct类型，定义如下:

type GetBucketTaggingOutput struct {

TagSet []\*Tag `locationNameList:"Tag" type:"list" required:"true"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| TagSet | 标签列表 | []\*Tag |

Tag:类型是个struct，定义如下:

type Tag struct {

Key \*string `min:"1" type:"\*string" required:"true"`

Value \*string `type:"\*string" required:"true"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| Key | 标签的Key | \*string |
| Value | 标准的Value | \*string |

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

resp, err := svc.GetBucketTagging(&s3.GetBucketTaggingInput{

Bucket: aws.\*string("bucket-1"),

})

if err != nil {

fmt.Println("failed", err)

} else {

fmt.Println("success", resp)

}

}

1.21、Delete Bucket Tagging

功能说明

删除Bucket上的标签。该操作需要s3:PutBucketTagging权限，桶的拥有者默认具有该权限。

方法原型

func (c \*S3) DeleteBucketTagging(input \*DeleteBucketTaggingInput) (\*DeleteBucketTaggingOutput, error)

参数说明

* input:类型是个struct，定义如下:

type DeleteBucketTaggingInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket的名称 | \*string | 是 |

返回结果说明

* 正常输出类型为PutBucketTaggingOutput，否则返回error，

PutBucketTaggingOutput是struct类型，定义如下:

type DeleteBucketTaggingOutput struct {

}

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

resp, err := svc.DeleteBucketTagging(&s3.DeleteBucketTaggingInput{

Bucket: aws.\*string("bucket-1"),

})

if err != nil {

fmt.Println("failed", err)

} else {

fmt.Println("success", resp)

}

}

1.22、Put Bucket Encrytion

功能说明

put bucket encryption请求可以启用存储桶默认加密功能

方法原型

func (c \*S3) PutBucketEncryption(input \*PutBucketEncryptionInput) (\*PutBucketEncryptionOutput, error)

参数说明

* input:类型是个struct，具体定义属性如下:

type PutBucketEncryptionInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

ServerSideEncryptionConfiguration \*ServerSideEncryptionConfiguration `locationName:"ServerSideEncryptionConfiguration" type:"structure" required:"true" xmlURI:"http://s3.amazonaws.com/doc/2006-03-01/"`

}

ServerSideEncryptionConfiguration:类型是个struct，具体定义属性如下:

type ServerSideEncryptionConfiguration struct {

Rules []\*ServerSideEncryptionRule `locationName:"Rule" type:"list" flattened:"true" required:"true"`

}

ServerSideEncryptionRule:类型是个list，具体定义属性如下:

type ServerSideEncryptionRule struct {

ApplyServerSideEncryptionByDefault \*ServerSideEncryptionByDefault `type:"structure"`

}

ServerSideEncryptionByDefault:类型是个struct，具体定义属性如下:

type ServerSideEncryptionByDefault struct {

KMSMasterKeyID \*string `type:"\*string" sensitive:"true"`

SSEAlgorithm \*string `type:"\*string" required:"true" enum:"ServerSideEncryption"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket的名称 | \*string | 是 |
| ServerSideEncryptionConfiguration | 指定默认服务端加密配置 | \*struct | 是 |
| Rules | 一个特殊的服务端加密配置规则信息 | \*list | 是 |
| ApplyServerSideEncryptionByDefault | 指定默认的服务端加密会应用于新对象上传至存储桶时。若是在上传对象时请求中未指定任何加密信息，则存储桶默认加密将会应用 | \*struct | 否 |
| SSEAlgorithm | 服务端加密算法。取值为AES256或aws:kms | \*string | 是 |
| KMSMasterKeyID | 若加密算法选用的是aws:kms，则此项必填，按照cmkuuid:keyspec:userid模式配置，其中cmkuuid是CMKID，keyspec是指定生成的数据密钥长度，userid是用户id；若是AES256算法，则此项可不填，若填，则字符长度需为32 | \*string | 否 |

返回结果说明

* PutBucketEncryptionOutPut:类型是个struct，具体定义属性如下:

type PutBucketEncryptionOutPut struct {

}

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/awserr"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

en := &s3.PutBucketEncryptionInput{

Bucket: aws.\*string("bucket\_sdk"),

ServerSideEncryptionConfiguration: &s3.ServerSideEncryptionConfiguration{

Rules: []\*s3.ServerSideEncryptionRule{

{

ApplyServerSideEncryptionByDefault: &s3.ServerSideEncryptionByDefault{

SSEAlgorithm: aws.\*string("aws:kms");

KMSMasterKeyID: aws.\*string("6b1f657c-816b-4534-a41a-903e7a60e703:AES\_256:e3d16fba6ae84e33a1d386dd880696c0")

}

}

}

},

}

resp, err := svc.PutBucketEncryption(en)

if err != nil {

if aerr, ok := err.(awserr.Error); ok {

switch aerr.Code() {

default:

fmt.Println(aerr.Error())

}

} else {

fmt.Println(err.Error())

}

} else {

fmt.Println("success", resp)

}

}

1.23、Get Bucket Encrytion

功能说明

get bucket encryption请求可以返回存储桶默认加密配置。若是存储桶不存 在默认加密配置，则返回NoSuchEncryptionSetError错误。

方法原型

func (c \*S3) GetBucketEncryption(input \*GetBucketEncryptionInput) (\*GetBucketEncryptionOutput, error)

参数说明

* input:类型是个struct，具体定义属性如下:

type GetBucketEncryptionInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket的名称 | \*string | 是 |

返回结果说明

* GetBucketEncryptionOutput:类型是个struct，具体定义属性如下:

type GetBucketEncryptionOutput struct {

ServerSideEncryptionConfiguration \*ServerSideEncryptionConfiguration `type:"structure"`

}

ServerSideEncryptionConfiguration:类型是个struct，具体定义属性如下:

type ServerSideEncryptionConfiguration struct {

Rules []\*ServerSideEncryptionRule `locationName:"Rule" type:"list" flattened:"true" required:"true"`

}

ServerSideEncryptionRule:类型是个list，具体定义属性如下:

type ServerSideEncryptionRule struct {

ApplyServerSideEncryptionByDefault \*ServerSideEncryptionByDefault `type:"structure"`

}

ServerSideEncryptionByDefault:类型是个struct，具体定义属性如下:

type ServerSideEncryptionByDefault struct {

KMSMasterKeyID \*string `type:"\*string" sensitive:"true"`

SSEAlgorithm \*string `type:"\*string" required:"true" enum:"ServerSideEncryption"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| ServerSideEncryptionConfiguration | 默认的服务端加密配置 | \*struct |
| Rules | 一个特殊的服务端加密配置规则信息 | \*list |
| ApplyServerSideEncryptionByDefault | 指定默认的服务端加密会应用于新对象上传至存储桶时。若是在上传对象时请求中未指定任何加密信息，则存储桶默认加密将会应用 | \*struct |
| SSEAlgorithm | 服务端加密算法。取值为AES256或aws:kms | \*string |
| KMSMasterKeyID | 若加密算法是aws:kms，则返回cmkuuid:keyspec:userid模式配置，其中cmkuuid是CMKID，keyspec是指定生成的数据密钥长度，userid是用户id；若是AES256算法，则返回字符长度为32的明文密钥 | \*string |

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/awserr"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

en := &s3.GetBucketEncryptionInput{

Bucket: aws.\*string("bucket\_sdk"),

}

resp, err := svc.GetBucketEncryption(en)

if err != nil {

if aerr, ok := err.(awserr.Error); ok {

switch aerr.Code() {

default:

fmt.Println(aerr.Error())

}

} else {

fmt.Println(err.Error())

}

} else {

fmt.Println("success", resp)

}

}

1.24、Delete Bucket Encrytion

功能说明

delete bucket encryption请求删除存储桶默认加密配置

方法原型

func (c \*S3) DeleteBucketEncryption(input \*DeleteBucketEncryptionInput) (\*DeleteBucketEncryptionOutput, error)

参数说明

* input:类型是个struct，具体定义属性如下:

type DeleteBucketEncryptionInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket的名称 | \*string | 是 |

返回结果说明

* DeleteBucketEncryptionOutput:类型是个struct，具体定义属性如下:

type DeleteBucketEncryptionOutput struct {

}

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/awserr"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

en := &s3.DeleteBucketEncryptionInput{

Bucket: aws.\*string("bucket\_sdk"),

}

resp, err := svc.DeleteBucketEncryption(en)

if err != nil {

if aerr, ok := err.(awserr.Error); ok {

switch aerr.Code() {

default:

fmt.Println(aerr.Error())

}

} else {

fmt.Println(err.Error())

}

} else {

fmt.Println("success", resp)

}

}

1.25、Put Bucket Object Lock Configuration

功能说明

put bucket object lock请求在指定的存储桶上增加对象锁定配置。默认规 则将会应用到每一个新放入桶中的对象。

方法原型

func (c \*S3) PutObjectLockConfiguration(input \*PutObjectLockConfigurationInput) (\*PutObjectLockConfigurationOutput, error)

参数说明

* input:类型是个struct，具体定义属性如下:

type PutObjectLockConfigurationInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

ObjectLockConfiguration \*ObjectLockConfiguration `locationName:"ObjectLockConfiguration" type:"structure" xmlURI:"http://s3.amazonaws.com/doc/2006-03-01/"`

}

ObjectLockConfiguration:类型是个struct，具体定义属性如下:

type ObjectLockConfiguration struct {

ObjectLockEnabled \*string `type:"\*string" enum:"ObjectLockEnabled"`

Rule \*ObjectLockRule `type:"structure"`

}

ObjectLockRule:类型是个struct，具体定义属性如下:

type ObjectLockRule struct {

DefaultRetention \*DefaultRetention `type:"structure"`

}

DefaultRetention:类型是个struct，具体定义属性如下:

type DefaultRetention struct {

Days \*int64 `type:"integer"`

Mode \*string `type:"\*string" enum:"ObjectLockRetentionMode"`

Years \*int64 `type:"integer"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket的名称 | \*string | 是 |
| ObjectLockConfiguration | 应用到指定存储桶的对象锁定配置 | \*struct | 否 |
| ObjectLockEnabled | 表示指定桶的对象锁定功能是否生效。当对存储桶设置了对象锁定配置，即为生效 | \*string | 否 |
| Rule | 指定对象的对象锁定规则，配置需要指定模式和时间。年或日只能指定一个，不能在配置中既指定年又指定日 | \*struct | 否 |
| DefaultRetention | 对象锁定规则中指定的默认模式和时间。存储桶配置同时需要模式和时间，年和日不能同时指定，只能指定一个 | \*struct | 否 |
| Mode | 存储桶默认的对象锁定保留期限模式。取值为GOVERNANCE或COMPLIANCE | \*string | 否 |
| Days | 保留期限日期，天。与年只能二选其一 | \*integer | 否 |
| Years | 保留期限日期，年。与天只能二选其一 | \*integer | 否 |

返回结果说明

* PutObjectLockConfigurationOutPut:类型是个struct，具体定义属性如下:

type PutObjectLockConfigurationOutPut struct {

}

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/awserr"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

ol := &s3.PutObjectLockConfigurationInput{

Bucket: aws.\*string("bucket\_sdk"),

ObjectLockConfiguration: &s3.ObjectLockConfiguration{

ObjectLockEnabled: aws.\*string("Enabled"),

Rule: &s3.ObjectLockRule{

DefaultRetention: &s3.DefaultRetention{

Mode: aws.\*string("COMPLIANCE"),

Years: aws.Int64(1),

},

},

},

}

resp, err := svc.PutObjectLockConfiguration(ol)

if err != nil {

if aerr, ok := err.(awserr.Error); ok {

switch aerr.Code() {

default:

fmt.Println(aerr.Error())

}

} else {

fmt.Println(err.Error())

}

} else {

fmt.Println("success", resp)

}

}

1.26、Get Bucket Object Lock Configuration

功能说明

get bucket object lock请求获取存储桶的对象锁定配置。默认的对象锁定功能将会应用到每一个新放入到存储桶中的对象。

方法原型

func (c \*S3) GetObjectLockConfiguration(input \*GetObjectLockConfigurationInput) (\*GetObjectLockConfigurationOutput, error)

参数说明

* input:类型是个struct，具体定义属性如下:

type GetObjectLockConfigurationInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket的名称 | \*string | 是 |

返回结果说明

* GetObjectLockConfigurationOutPut:类型是个struct，具体定义属性如下:

type GetObjectLockConfigurationOutPut struct {

ObjectLockConfiguration \*ObjectLockConfiguration `type:"structure"`

}

ObjectLockConfiguration:类型是个struct，具体定义属性如下:

type ObjectLockConfiguration struct {

ObjectLockEnabled \*string `type:"\*string" enum:"ObjectLockEnabled"`

Rule \*ObjectLockRule `type:"structure"`

}

ObjectLockRule:类型是个struct，具体定义属性如下:

type ObjectLockRule struct {

DefaultRetention \*DefaultRetention `type:"structure"`

}

DefaultRetention:类型是个struct，具体定义属性如下:

type DefaultRetention struct {

Days \*int64 `type:"integer"`

Mode \*string `type:"\*string" enum:"ObjectLockRetentionMode"`

Years \*int64 `type:"integer"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| ObjectLockConfiguration | 应用到指定存储桶的对象锁定配置 | \*struct |
| ObjectLockEnabled | 表示指定桶的对象锁定功能是否生效。当对存储桶设置了对象锁定配置，即为生效 | \*string |
| Rule | 指定对象的对象锁定规则，配置需要指定模式和时间。年或日只能指定一个，不能在配置中既指定年又指定日 | \*struct |
| DefaultRetention | 对象锁定规则中指定的默认模式和时间。存储桶配置同时需要模式和时间，年和日不能同时指定，只能指定一个 | \*struct |
| Mode | 存储桶默认的对象锁定保留期限模式。取值为GOVERNANCE或COMPLIANCE | \*string |
| Days | 保留期限日期，天 | \*integer |
| Years | 保留期限日期，年 | \*integer |

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/awserr"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

ol := &s3.GetObjectLockConfigurationInput{

Bucket: aws.\*string("bucket\_sdk"),

}

resp, err := svc.GetObjectLockConfiguration(ol)

if err != nil {

if aerr, ok := err.(awserr.Error); ok {

switch aerr.Code() {

default:

fmt.Println(aerr.Error())

}

} else {

fmt.Println(err.Error())

}

} else {

fmt.Println("success", resp)

}

}

1.27、Put Bucket Logging

功能说明

put bucket logging请求设置日志转存参数。所有的日志将会保留到和源存储桶属于同一拥有者的目标存储桶中。桶的拥有者可以设置桶的日志状态。桶的拥有者对所有的日志具有FULL\_CONTROL权限，可以通过Grantee授权其他用户，其中Permissions参数指定了用户对日志的访问权限。

方法原型

func (c \*S3) PutBucketLogging(input \*PutBucketLoggingInput) (\*PutBucketLoggingOutput, error)

参数说明

* input:类型是个struct，具体定义属性如下:

type PutBucketLoggingInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

BucketLoggingStatus \*BucketLoggingStatus `locationName:"BucketLoggingStatus" type:"structure" required:"true" xmlURI:"http://s3.amazonaws.com/doc/2006-03-01/"`

}

BucketLoggingStatus:类型是个struct，具体定义属性如下:

type ObjectLockConfiguration struct {

LoggingEnabled \*LoggingEnabled `type:"structure"`

}

LoggingEnabled:类型是个struct，具体定义属性如下:

type LoggingEnabled struct {

TargetBucket \*string `type:"\*string" required:"true"`

TargetGrants []\*TargetGrant `locationNameList:"Grant" type:"list"`

TargetPrefix \*string `type:"\*string" required:"true"`

}

TargetGrant:类型是个list，具体定义属性如下:

type TargetGrant struct {

Grantee \*Grantee `type:"structure" xmlPrefix:"xsi" xmlURI:"http://www.w3.org/2001/XMLSchema-instance"`

Permission \*string `type:"\*string" enum:"BucketLogsPermission"`

}

Grantee:类型是个struct，具体定义属性如下:

type TargetGrant struct {

DisplayName \*string `type:"\*string"`

EmailAddress \*string `type:"\*string"`

ID \*string `type:"\*string"`

Type \*string `locationName:"xsi:type" type:"\*string" xmlAttribute:"true" required:"true" enum:"Type"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket的名称 | \*string | 是 |
| BucketLoggingStatus | 日志状态信息。此值为空表示关闭日志转存功能 | \*struct | 是 |
| LoggingEnabled | 描述日志存储位置和日志对象前缀 | \*struct | 否 |
| TargetBucket | 日志存储位置。可以将日志存放到任意用户拥有的桶中，包含源存储桶。用户可以配置多个源桶的日志均投放到同一个目标存储桶中，在这种情况下，用户可以使用TargetPrefix区分日志来自哪个源存储桶。 | \*string | 是 |
| TargetGrants | 授权信息 | \*list | 否 |
| Grantee | 授权许可 | \*struct | 否 |
| DisplayName | 展示名字 | \*string | 否 |
| EmailAddress | 邮件地址 | \*string | 否 |
| ID | 授权用户ID | \*string | 否 |
| Type | 授权类型 | \*string | 是 |
| URI | 授权组URI | \*string | 否 |
| Permission | 日志访问许可 | \*string | 否 |
| TargetPrefix | 日志对象前缀。若多个源存储桶的日志均写到同一个目标存储桶中，则可以通过目标前缀来区分日志来自哪一个源存储桶 | \*string | 是 |

返回结果说明

* PutBucketLoggingOutPut:类型是个struct，具体定义属性如下:

type PutBucketLoggingOutPut struct {

}

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/awserr"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

en := &s3.PutBucketLoggingInput{

Bucket: aws.\*string("bucket\_sdk"),

BucketLoggingStatus: &s3.BucketLoggingStatus{

LoggingEnabled: &s3.LoggingEnabled{

TargetBucket: aws.\*string("target\_bucket\_sdk"),

TargetGrants: []\*s3.TargetGrant{

{

Grantee: &s3.Grantee{

EmailAddress: aws.\*string("john@example.com"),

Type: aws.\*string("AmazonCustomerByEmail"),

},

Permission: aws.\*string("FULL\_CONTROL"),

},

},

TargetPrefix: aws.\*string("log/"),

},

},

}

resp, err := svc.PutBucketLogging(en)

if err != nil {

if aerr, ok := err.(awserr.Error); ok {

switch aerr.Code() {

default:

fmt.Println(aerr.Error())

}

} else {

fmt.Println(err.Error())

}

} else {

fmt.Println("success", resp)

}

}

1.28、Get Bucket Logging

功能说明

get bucket logging请求获取存储桶的日志转存配置

方法原型

func (c \*S3) GetBucketLogging(input \*GetBucketLoggingInput) (\*GetBucketLoggingOutput, error)

参数说明

* input:类型是个struct，具体定义属性如下:

type GetBucketLoggingInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket的名称 | \*string | 是 |

返回结果说明

* GetBucketLoggingOutPut:类型是个struct，具体定义属性如下:

type GetBucketLoggingOutPut struct {

LoggingEnabled \*LoggingEnabled `type:"structure"`

}

LoggingEnabled:类型是个struct，具体定义属性如下:

type LoggingEnabled struct {

TargetBucket \*string `type:"\*string" required:"true"`

TargetGrants []\*TargetGrant `locationNameList:"Grant" type:"list"`

TargetPrefix \*string `type:"\*string" required:"true"`

}

TargetGrant:类型是个list，具体定义属性如下:

type TargetGrant struct {

Grantee \*Grantee `type:"structure" xmlPrefix:"xsi" xmlURI:"http://www.w3.org/2001/XMLSchema-instance"`

Permission \*string `type:"\*string" enum:"BucketLogsPermission"`

}

Grantee:类型是个struct，具体定义属性如下:

type TargetGrant struct {

DisplayName \*string `type:"\*string"`

EmailAddress \*string `type:"\*string"`

ID \*string `type:"\*string"`

Type \*string `locationName:"xsi:type" type:"\*string" xmlAttribute:"true" required:"true" enum:"Type"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| LoggingEnabled | 描述日志存储位置和日志对象前缀 | \*struct |
| TargetBucket | 日志存储位置。可以将日志存放到任意用户拥有的桶中，包含源存储桶。用户可以配置多个源桶的日志均投放到同一个目标存储桶中，在这种情况下，用户可以使用TargetPrefix区分日志来自哪个源存储桶。 | \*string |
| TargetGrants | 授权信息 | \*list |
| Grantee | 授权许可 | \*struct |
| DisplayName | 展示名字 | \*string |
| EmailAddress | 邮件地址 | \*string |
| ID | 授权用户ID | \*string |
| Type | 授权类型 | \*string |
| URI | 授权组URI | \*string |
| Permission | 日志访问许可 | \*string |
| TargetPrefix | 日志对象前缀。若多个源存储桶的日志均写到同一个目标存储桶中，则可以通过目标前缀来区分日志来自哪一个源存储桶 | \*string |

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/awserr"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

ol := &s3.GetBucketLoggingInput{

Bucket: aws.\*string("bucket\_sdk"),

}

resp, err := svc.GetBucketLogging(ol)

if err != nil {

if aerr, ok := err.(awserr.Error); ok {

switch aerr.Code() {

default:

fmt.Println(aerr.Error())

}

} else {

fmt.Println(err.Error())

}

} else {

fmt.Println("success", resp)

}

}

1.29、Put Bucket CORS

功能说明

Put Bucket CORS 接口用来请求设置 Bucket 的跨域资源共享权限。

方法原型

func (c \*[S3](#S3)) PutBucketCors(input \*[PutBucketCorsInput](#PutBucketCorsInput)) (\*[PutBucketCorsOutput](#PutBucketCorsOutput), error)

参数说明

type PutBucketCorsInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

CORSConfiguration \*CORSConfiguration `locationName:"CORSConfiguration" type:"structure" required:"true" xmlURI:"http://s3.amazonaws.com/doc/2006-03-01/"`

}

type CORSConfiguration struct {

CORSRules []\*CORSRule `locationName:"CORSRule" type:"list" flattened:"true" required:"true"`

}

type CORSRule struct {

AllowedHeaders []\*string `locationName:"AllowedHeader" type:"list" flattened:"true"`

AllowedMethods []\*string `locationName:"AllowedMethod" type:"list" flattened:"true" required:"true"`

AllowedOrigins []\*string `locationName:"AllowedOrigin" type:"list" flattened:"true" required:"true"`

ExposeHeaders []\*string `locationName:"ExposeHeader" type:"list" flattened:"true"`

ID \*string `type:"\*string"`

MaxAgeSeconds \*int64 `type:"integer"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | 要设置跨域的bucket名称 | \*string | 是 |
| CORSConfiguration | 描述Amazon S3桶中对象的跨源访问配置的参数容器 | struct | 是 |
| CORSRules | 为指定bucket配置的所有跨域规则的集合，允许配置100条规则 | list | 是 |
| ID | 跨域规则的ID,最大长度255 | \*string | 否 |
| AllowedHeaders | 允许浏览器发送 CORS 请求时携带的自定义 HTTP 请求头部，不区分英文大小写，单条 CORSRule 可以配置多个 AllowedHeader。 | list | 否 |
| AllowedMethods | 允许该源执行的HTTP方法列表，包括GET , PUT , HEAD , POST , and DELETE。单挑规则可以配置多个方法。 | list | 是 |
| AllowedOrigins | 允许能够访问该bucket的一个或多个源,支持 \* 通配符，表示所有域名都允许访问，不推荐。  一条CORSRule可以配置多个allowedorigins | list | 是 |
| ExposeHeaders | 允许浏览器获取的 CORS 请求响应中的头部，不区分英文大小写,单条 CORSRule 可以配置多个 ExposeHeader。 | list | 否 |
| MaxAgeSeconds | 跨域资源共享配置的有效时间，单位为秒，对应 CORS 请求响应中的 Access-Control-Max-Age 头部，单条 CORSRule 只能配置一个 MaxAgeSeconds | \*int64 | 否 |

返回结果说明

type [PutBucketCorsOutput](#PutBucketCorsOutput)struct {

// contains filtered or unexported fields

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| [PutBucketCorsOutput](#PutBucketCorsOutput) | 返回的结果，无需特殊处理 | struct |

示例

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

"github.com/aws/aws-sdk-go/aws/awserr"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

input := &s3.PutBucketCorsInput{

Bucket: aws.\*string("rgwuser01-testbucket03"),

CORSConfiguration: &s3.CORSConfiguration{

CORSRules: []\*s3.CORSRule{

{

AllowedHeaders: []\*string{

aws.\*string("\*"),

},

AllowedMethods: []\*string{

aws.\*string("PUT"),

aws.\*string("POST"),

aws.\*string("DELETE"),

},

AllowedOrigins: []\*string{

aws.\*string("http://www.example.com"),

},

ExposeHeaders: []\*string{

aws.\*string("x-amz-server-side-encryption"),

},

MaxAgeSeconds: aws.Int64(3000),

},

{

AllowedHeaders: []\*string{

aws.\*string("Authorization"),

},

AllowedMethods: []\*string{

aws.\*string("GET"),

},

AllowedOrigins: []\*string{

aws.\*string("\*"),

},

MaxAgeSeconds: aws.Int64(3000),

},

},

},

}

result, err := svc.PutBucketCors(input)

if err != nil {

if aerr, ok := err.(awserr.Error); ok {

switch aerr.Code() {

default:

fmt.Println(aerr.Error())

}

} else {

// Print the error, cast err to awserr.Error to get the Code and

// Message from an error.

fmt.Println(err.Error())

}

return

}

fmt.Println(result)

}

1.30、Get Bucket CORS

功能说明

Get Bucket CORS 接口用来请求获取 Bucket 的跨域资源共享权限配置。

方法原型

func (c \*[S3](#S3)) GetBucketCors(input \*[GetBucketCorsInput](#GetBucketCorsInput)) (\*[GetBucketCorsOutput](#GetBucketCorsOutput), error)

参数说明

type GetBucketCorsInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | 要获取cors配置的bucket名称 | \*string | 否 |

返回结果说明

type GetBucketCorsOutput struct {

CORSRules []\*CORSRule `locationName:"CORSRule" type:"list" flattened:"true"`

}

type CORSRule struct {

AllowedHeaders []\*string `locationName:"AllowedHeader" type:"list" flattened:"true"`

AllowedMethods []\*string `locationName:"AllowedMethod" type:"list" flattened:"true" required:"true"`

AllowedOrigins []\*string `locationName:"AllowedOrigin" type:"list" flattened:"true" required:"true"`

ExposeHeaders []\*string `locationName:"ExposeHeader" type:"list" flattened:"true"`

ID \*string `type:"\*string"`

MaxAgeSeconds \*int64 `type:"integer"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| GetBucketCorsOutput | 返回的跨域配置信息的容器 | struct |
| CORSRules | 为指定bucket配置的所有跨域规则的集合，允许配置100条规则 | list |
| ID | 跨域规则的ID,最大长度255 | \*string |
| AllowedHeaders | 允许浏览器发送 CORS 请求时携带的自定义 HTTP 请求头部，不区分英文大小写，单条 CORSRule 可以配置多个 AllowedHeader。 | list |
| AllowedMethods | 允许该源执行的HTTP方法列表，包括GET , PUT , HEAD , POST , and DELETE | list |
| AllowedOrigins | 允许能够访问该bucket的一个或多个源 | list |
| ExposeHeaders | 允许浏览器获取的 CORS 请求响应中的头部，不区分英文大小写,单条 CORSRule 可以配置多个 ExposeHeader。 | list |
| MaxAgeSeconds | 跨域资源共享配置的有效时间，单位为秒，对应 CORS 请求响应中的 Access-Control-Max-Age 头部，单条 CORSRule 只能配置一个 MaxAgeSeconds | \*int64 |

示例

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

"github.com/aws/aws-sdk-go/aws/awserr"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

input := &s3.GetBucketCorsInput{

Bucket : aws.\*string("rgwuser01-testbucket03"),

}

result ,err := svc.GetBucketCors(input)

if err != nil {

if aerr, ok := err.(awserr.Error); ok {

switch aerr.Code() {

default:

fmt.Println(aerr.Error())

}

} else {

// Print the error, cast err to awserr.Error to get the Code and

// Message from an error.

fmt.Println(err.Error())

}

return

}

fmt.Println(result)

}

1.31、Delete Bucket CORS

功能说明

Delete Bucket CORS 接口用来删除 Bucket 的跨域资源共享权限配置。

方法原型

func (c \*S3) DeleteBucketCors(input \*DeleteBucketCorsInput) (\*DeleteBucketCorsOutput, error)

参数说明

type DeleteBucketCorsInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | 要删除cors配置的bucket名称 | \*string | 是 |

返回结果说明

type [PutBucketCorsOutput](#PutBucketCorsOutput)struct {

// contains filtered or unexported fields

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| [DeleteBucketCorsOutput](#PutBucketCorsOutput) | 返回的结果，无需特殊处理 | struct |

示例

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

"github.com/aws/aws-sdk-go/aws/awserr"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

input := &s3.DeleteBucketCorsInput{

Bucket : aws.\*string("rgwuser01-testbucket03"),

}

result ,err := svc.DeleteBucketCors(input)

if err != nil {

if aerr, ok := err.(awserr.Error); ok {

switch aerr.Code() {

default:

fmt.Println(aerr.Error())

}

} else {

// Print the error, cast err to awserr.Error to get the Code and

// Message from an error.

fmt.Println(err.Error())

}

return

}

fmt.Println(result)

}

1.32、Put Bucket Versioning

功能说明

Put Bucket Versioning 接口实现启用或者暂停Bucket的版本控制功能。

方法原型

func (c \*S3) PutBucketVersioning(input \*PutBucketVersioningInput) (\*PutBucketVersioningOutput, error)

参数说明

* input:类型是个class，具体定义属性如下:

type PutBucketVersioningInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

VersioningConfiguration \*VersioningConfiguration `locationName:"VersioningConfiguration" type:"structure" required:"true"

}

VersioningConfiguration是个struct,定义属性如下:

type VersioningConfiguration struct {

Status \*string `type:"\*string" enum:"BucketVersioningStatus"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket的名称 | \*string | 是 |
| VersioningConfiguration | 是否开启Bucket的版本控制功能，枚举值：Suspended，Enabled。 | \*string | 是 |

返回结果说明

* PutBucketVersioningOutput:类型是个struct，具体定义属性如下:

type PutBucketVersioningOutput struct {

// contains filtered or unexported fields

}

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

input := &s3.PutBucketVersioningInput{

Bucket: aws.\*string("bucket\_name"),

VersioningConfiguration: &s3.VersioningConfiguration{

Status:aws.\*string("Enabled"),

},

}

result, err := svc.PutBucketVersioning(input)

if err != nil {

fmt.Println("requeset fail ")

fmt.Println(err)

return

}

fmt.Println("requeset success ")

fmt.Println(result)

}

1.33、Get Bucket Versioning

功能说明

Get Bucket Versioning 接口实现获得Bucket的版本控制配置。

方法原型

func (c \*S3) GetBucketVersioning(input \*GetBucketVersioningInput) (\*GetBucketVersioningOutput, error)

参数说明

* input:类型是个struct，具体定义方法如下:

type GetBucketVersioningInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket的名称 | \*string | 是 |

返回结果说明

* GetBucketVersioningOutput:类型是个struct，具体定义属性如下:

type GetBucketVersioningOutput struct {

Status \*string `type:"\*string" enum:"BucketVersioningStatus"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **返回结果描述** | **类型** |
| Status | Bucket的版本控制配置，假如没有设置过该配置，该字段不会返回 | \*string |

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

input := &s3.GetBucketVersioningInput {

Bucket: aws.\*string("bucket\_version"),

}

result, err := svc.GetBucketVersioning(input)

if err != nil {

fmt.Println("requeset fail ")

fmt.Println(err)

return

}

fmt.Println("requeset success ")

fmt.Println(\*result.Status)

}

1.34、Put Bucket Notification Configuration

功能说明

设置桶的指定事件通知。使用此API，可以替换现有的通知配置。配置定义了您希望ZOS发布的事件类型，以及当ZOS检测到指定类型的事件时，您希望ZOS发布事件通知的目的地。默认情况下，桶没有配置事件通知。也就是说，通知配置将是一个空的NotificationConfiguration。在ZOS接收到这个请求之后，它首先验证通知的目的地是否存在，以及bucket所有者是否具有发送测试通知的权限。

默认情况下，只有桶的所有者可以配置桶的通知。但是，桶的所有者可以使用桶策略授予其他用户设置s3:PutBucketNotification权限的权限。

方法原型

func (c \*S3) PutBucketNotificationConfiguration(input \*PutBucketNotificationConfigurationInput) (\*PutBucketNotificationConfigurationOutput, error)

参数说明

* input:类型是个struct，具体定义属性如下:

type PutBucketNotificationConfigurationInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

NotificationConfiguration \*NotificationConfiguration `locationName:"NotificationConfiguration" type:"structure" required:"true" xmlURI:"http://s3.amazonaws.com/doc/2006-03-01/"`

}

NotificationConfiguration:类型是个struct，具体定义属性如下:

type NotificationConfiguration struct {

TopicConfigurations []\*TopicConfiguration `locationName:"TopicConfiguration" type:"list" flattened:"true"`

}

TopicConfiguration:类型是个struct，具体定义属性如下:

type TopicConfiguration struct {

Events []\*string `locationName:"Event" type:"list" flattened:"true" required:"true"`

Filter \*NotificationConfigurationFilter `type:"structure"`

Id \*string `type:"\*string"`

TopicArn \*string `locationName:"Topic" type:"\*string" required:"true"`

}

NotificationConfigurationFilter:类型是个struct，具体定义属性如下:

type NotificationConfigurationFilter struct {

Key \*KeyFilter `locationName:"S3Key" type:"structure"`

}

KeyFilter:类型是个struct，具体定义属性如下:

type KeyFilter struct {

FilterRules []\*FilterRule `locationName:"FilterRule" type:"list" flattened:"true"`

}

FilterRule:类型是个struct，具体定义属性如下:

type FilterRule struct {

Name \*string `type:"\*string" enum:"FilterRuleName"`

Value \*string `type:"\*string"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket的名称 | \*string | 是 |
| NotificationConfiguration | 通知的配置信息，含有TopicConfiguration | \*NotificationConfiguration | 是 |
| TopicConfigurations | 主题类型的通知配置信息，含有Id,TopicArn和Event | []\*TopicConfiguration | 是 |
| Events | 触发通知的事件类型。可以同时支持多个事件。支持的事件有's3:ObjectCreated:\*','s3:ObjectCreated:Put','s3:ObjectCreated:Post','s3:ObjectCreated:Copy','s3:ObjectCreated:CompleteMultipartUpload','s3:ObjectRemoved:\*','s3:ObjectRemoved:Delete','s3:ObjectRemoved:DeleteMarkerCreated' | \*string | 是 |
| Filter | 过滤器 | \*NotificationConfigurationFilter | 否 |

返回结果说明

返回结无有意义字段，无需处理。

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

accessKey := "test"

secretKey := "test"

endPoint := "192.168.198.110:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(accessKey, secretKey, ""),

Endpoint: aws.\*string(endPoint),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

resp, err := svc.PutBucketNotificationConfiguration(&s3.PutBucketNotificationConfigurationInput{

Bucket: aws.\*string("java-zos"),

NotificationConfiguration: &s3.NotificationConfiguration{

TopicConfigurations: []\*s3.TopicConfiguration{

{

Events: []\*string{

aws.\*string("s3:ObjectCreated:\*"),

aws.\*string("s3:ObjectRemoved:\*"),

},

Id: aws.\*string("java-zos-test"),

TopicArn: aws.\*string("arn:aws:sns:default::Kafka\_Topic"),

},

},

},

})

if err != nil {

fmt.Println("failed", err)

} else {

fmt.Println("success", resp)

}

}

1.35、Get Bucket Notification Configuration

功能说明

返回桶的通知配置。如果桶上没有通知，则返回一个不包含TopicConfigurations元素的响应。当指定Notification名称时，返回该Notification的配置信息；当不指定Notification名称时，返回该桶下所有Notification的配置信息。

默认情况下，必须是桶的所有者才能读取桶的通知配置。但是，桶的所有者可以使用桶策略授予其他用户使用s3:GetBucketNotification权限读取该配置的权限。

方法原型

func (c \*S3) GetBucketNotificationConfiguration(input \*GetBucketNotificationConfigurationRequest) (\*NotificationConfiguration, error)

参数说明

* input:类型是个struct，具体定义属性如下:

type GetBucketNotificationConfigurationRequest struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket的名称 | \*string | 是 |

返回结果说明

* NotificationConfiguration:类型是个struct，具体定义属性如下:

type NotificationConfiguration struct {

TopicConfigurations []\*TopicConfiguration `locationName:"TopicConfiguration" type:"list" flattened:"true"`

}

TopicConfiguration:类型是个struct，具体定义属性如下:

type TopicConfiguration struct {

Events []\*string `locationName:"Event" type:"list" flattened:"true" required:"true"`

Filter \*NotificationConfigurationFilter `type:"structure"`

Id \*string `type:"\*string"`

TopicArn \*string `locationName:"Topic" type:"\*string" required:"true"`

}

NotificationConfigurationFilter:类型是个struct，具体定义属性如下:

type NotificationConfigurationFilter struct {

Key \*KeyFilter `locationName:"S3Key" type:"structure"`

}

KeyFilter:类型是个struct，具体定义属性如下:

type KeyFilter struct {

FilterRules []\*FilterRule `locationName:"FilterRule" type:"list" flattened:"true"`

}

FilterRule:类型是个struct，具体定义属性如下:

type FilterRule struct {

Name \*string `type:"\*string" enum:"FilterRuleName"`

Value \*string `type:"\*string"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| TopicConfigurations | TopicConfiguration的列表 | []\*TopicConfiguration |
| Events | 通知的事件 | \*string |
| Filter | 通知的过滤器 | \*NotificationConfigurationFilter |
| Id | 通知的Id | \*string |
| TopicArn | Topic的Arn标识 | \*string |
| Key | 过滤器 | \*KeyFilter |
| FilterRules | 过滤器规则列表 | []\*FilterRule |
| Name | 规则的键 | \*string |
| Value | 规则的值 | \*string |

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

accessKey := "test"

secretKey := "test"

endPoint := "192.168.198.110:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(accessKey, secretKey, ""),

Endpoint: aws.\*string(endPoint),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

resp, err := svc.GetBucketNotificationConfiguration(&s3.GetBucketNotificationConfigurationRequest{

Bucket: aws.\*string("java-zos"),

})

if err != nil {

fmt.Println("failed", err)

} else {

fmt.Println("success", resp)

}

}

2、Object 操作

2.1、Get Object

功能说明

Get Object 请求可以将一个文件（Object）下载至本地。该操作需要对目标 Object 具有读权限或目标 Object 对所有人都开放了读权限（公有读）。

方法原型

func (c \*S3) GetObject(input \*GetObjectInput) (\*GetObjectOutput, error)

参数说明

* GetObjectInput是个 struct， 具体定义字段如下

type GetObjectInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string"

required:"true"`

IfMatch \*string `location:"header" locationName:"If-Match" type:"\*string"`

IfModifiedSince \*time.Time `location:"header" locationName:"If-Modified-Since" type:"timestamp"`

IfNoneMatch \*string `location:"header" locationName:"If-None-Match"

type:"\*string"`

IfUnmodifiedSince \*time.Time `location:"header" locationName:"If-Unmodified-Since" type:"timestamp"`

Key \*string `location:"uri" locationName:"Key" min:"1" type:"\*string"

required:"true"`

Range \*string `location:"header" locationName:"Range" type:"\*string"`

VersionId \*string `location:"query\*string" locationName:"versionId"

type:"\*string"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket的名称 | \*string | 是 |
| IfMatch | 当对象的Etag和IfMatch中的值一致时返回对象，否则返回412错误 | \*string | 否 |
| IfModifiedSince | 只有指定时间之后有修改记录的对象才会返回对象，否则返回304错误 | \*string | 否 |
| IfNoneMatch | 只有对象的Etag不满足指定字符串时才会返回对象，否则返回304错误 | \*string | 否 |
| IfUnmodifiedSince | 只有指定时间之后没有修改记录的对象才会返回，否则返回412错误 | \*string | 否 |
| Key | Object的名称 | \*string | 是 |
| Range | 指定下载对象的字节范围，例如 bytes=0-9 表示下载开头10个byte | \*string | 否 |
| VersionId | 对象的某个特定版本的版本号，没有该版本则返回404错误 | \*string | 否 |

返回结果说明

type GetObjectOutput struct {

Body io.ReadCloser `type:"blob"`

DeleteMarker \*bool `location:"header" locationName:"x-amz-delete-marker" type:"boolean"`

ETag \*string `location:"header" locationName:"ETag" type:"\*string"`

LastModified \*time.Time `location:"header" locationName:"Last-Modified" type:"timestamp"`

Metadata map[\*string]\*string `location:"headers" locationName:"x-amz-meta-" type:"map"`

ObjectLockLegalHoldStatus \*string `location:"header" locationName:"x-amz-object-lock-legal-hold" type:"\*string" enum:"ObjectLockLegalHoldStatus"`

ObjectLockMode \*string `location:"header" locationName:"x-amz-object-lock-mode" type:"\*string" enum:"ObjectLockMode"`

ObjectLockRetainUntilDate \*time.Time `location:"header" locationName:"x-amz-object-lock-retain-until-date" type:"timestamp" timestampFormat:"iso8601"`

StorageClass \*string `location:"header" locationName:"x-amz-storage-class" type:"\*string" enum:"StorageClass"`

TagCount \*int64 `location:"header" locationName:"x-amz-tagging-count" type:"integer"`

VersionId \*string `location:"header" locationName:"x-amz-version-id" type:"\*string"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| Body | 返回对象的内容 | io.ReadCloser |
| DeleteMarker | 获取的对象是否是DeleteMarker | \*bool |
| LastModified | 对象的创建时间 | \*time.Time |
| ETag | 对象的ETag | \*string |
| Metadata | 和对象一起存储的元数据信息 | map[\*string]\*string |
| ObjectLockLegalHoldStatus | 获取Object LegalHold 的设置 | \*string |
| ObjectLockMode | 获取Object Retention 的模式 | \*string |
| ObjectLockRetainUntilDate | 获取Retention 过期日期 | \*time.Time |
| StorageClass | 获取对象的存储级别 | \*string |
| TagCount | 获取对象的tag数量 | \*int64 |
| VersionId | 获取Object 的VersionId | \*string |

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

input := &s3.GetObjectInput{

Bucket: aws.\*string("-sdk-bucket"),

Key: aws.\*string("go-sdk-object"),

}

result, err := svc.GetObject(input)

if err != nil {

fmt.Println("request fail")

fmt.Println(err)

return

}

fmt.Println("request success")

fmt.Println(result)

}

2.2、Head Object

功能说明

Head Object 请求可以获取对应 Object 的元数据，Head 的权限与 Get 的权限一致。

方法原型

func (c \*S3) HeadObject(input \*HeadObjectInput) (\*HeadObjectOutput, error)

参数说明

* input:类型是个struct，具体定义属性如下:

type HeadObjectInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

IfMatch \*string `location:"header" locationName:"If-Match" type:"\*string"`

IfModifiedSince \*time.Time `location:"header" locationName:"If-Modified-Since" type:"timestamp"`

IfNoneMatch \*string `location:"header" locationName:"If-None-Match" type:"\*string"`

IfUnmodifiedSince \*time.Time `location:"header" locationName:"If-Unmodified-Since" type:"timestamp"`

Key \*string `location:"uri" locationName:"Key" min:"1" type:"\*string" required:"true"`

VersionId \*string `location:"query\*string" locationName:"versionId" type:"\*string"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket的名称 | \*string | 是 |
| IfMatch | 当文件的Etag和IfMatch中的值一致时返回文件元数据，否则返回412错误 | \*string | 否 |
| IfModifiedSince | 只有指定时间之后有修改记录的文件才会返回元数据，否则返回304错误 | \*string | 否 |
| IfNoneMatch | 只有文件的Etag不满足指定字符串时才会返回元数据，否则返回304错误 | \*string | 否 |
| IfUnmodifiedSince | 只有指定时间之后没有修改记录的文件才会返回元数据，否则返回412错误 | \*string | 否 |
| Key | Object的名称 | \*string | 是 |
| VersionId | 文件的某个特定版本的版本号，没有该版本则返回404错误 | \*string | 否 |

返回结果说明

* HeadOjectOutput:是个struct，具体定义属性如下:

type HeadObjectOutput struct {

ContentLength \*int64 `location:"header" locationName:"Content-Length" type:"long"`

ETag \*string `location:"header" locationName:"ETag" type:"\*string"`

LastModified \*time.Time `location:"header" locationName:"Last-Modified" type:"timestamp"`

StorageClass \*string `location:"header" locationName:"x-amz-storage-class" type:"\*string" enum:"StorageClass"`

VersionId \*string `location:"header" locationName:"x-amz-version-id" type:"\*string"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| ContenLength | 文件大小，单位字节 | Int |
| Etag | 文件最新版本的etag | \*string |
| StorageClass | 文件的存储级别 | \*string |
| LastModified | 文件上次修改时间 | timestamp |
| VersionId | 文件最新版本 | \*string |

示例

package main

import (

"fmt"

"time"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

t := time.Now()

input := &s3.HeadObjectInput{

Bucket: aws.\*string("bucket\_version"),

Key: aws.\*string("rgw\_kv\_common.cc"),

IfModifiedSince:&t,

IfMatch:aws.\*string("ae1e94178c42c5c0a95925d457fefd42"),

}

result, err := svc.HeadObject(input)

if err != nil {

fmt.Println("requeset fail ")

fmt.Println(err)

return

}

fmt.Println("requeset success ")

fmt.Println(result)

}

2.3、Put Object

功能说明

Put Object 请求可以将一个文件（Oject）上传至指定 Bucket。

方法原型

func (c \*S3) PutObject(input \*PutObjectInput) (\*PutObjectOutput, error)

参数说明

* PutObjectInput: 类型是个struct， 具体定义如下

type PutObjectInput struct {

ACL \*string `location:"header" locationName:"x-amz-acl" type:"\*string" enum:"ObjectCannedACL"`

Append \*bool `location:"query\*string" locationName:"append" type:"boolean"`

AppendPosition \*int64 `location:"query\*string" locationName:"position" type:"integer"`

Body io.ReadSeeker `type:"blob"`

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

ContentMD5 \*string `location:"header" locationName:"Content-MD5" type:"\*string"`

GrantFullControl \*string `location:"header" locationName:"x-amz-grant-full-control" type:"\*string"`

GrantRead \*string `location:"header" locationName:"x-amz-grant-read" type:"\*string"`

GrantReadACP \*string `location:"header" locationName:"x-amz-grant-read-acp" type:"\*string"`

GrantWriteACP \*string `location:"header" locationName:"x-amz-grant-write-acp" type:"\*string"`

Key \*string `location:"uri" locationName:"Key" min:"1" type:"\*string" required:"true"`

Metadata map[\*string]\*string `location:"headers" locationName:"x-amz-meta-" type:"map"`

ObjectLockLegalHoldStatus \*string `location:"header" locationName:"x-amz-object-lock-legal-hold" type:"\*string" enum:"ObjectLockLegalHoldStatus"`

ObjectLockMode \*string `location:"header" locationName:"x-amz-object-lock-mode" type:"\*string" enum:"ObjectLockMode"`

ObjectLockRetainUntilDate \*time.Time `location:"header" locationName:"x-amz-object-lock-retain-until-date" type:"timestamp" timestampFormat:"iso8601"`

Tagging \*string `location:"header" locationName:"x-amz-tagging" type:"\*string"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| ACL | Bucket的ACL可选类型有private、public-read、public-read-write和authenticated-read | \*string | 否 |
| Append | 追加模式上传对象，仅支持设为 true | \*bool | 否 |
| AppendPosition | 追加模式下，指定追加的起始字节位置 | \*int64 | 否 |
| Body | 对象的数据 | io.ReadSeeker | 否 |
| Bucket | 存入bucket的名称 | \*string | 是 |
| ContentMD5 | 对象数据的MD5 | \*string | 否 |
| GrantFullControl | 被授权用户可以对对象进行read, write, read ACP, and write ACP操作  以下Grant\*参数，格式都是”id=xxxx”或”emailAddress=xxxx”或者”uri=xxxx”以及他们的组合（用逗号连接） | \*string | 否 |
| GrantRead | 被授权用户可以对对象进行读操作 | \*string | 否 |
| GrantReadACP | 被授权用户可以读取对象的ACL | \*string | 否 |
| GrantWriteACP | 被授权用户可以修改对象的ACL | \*string | 否 |
| Key | Object的名字 | \*string | 是 |
| Metadata | key value形式的Object Metadata | map[\*string]\*string | 否 |
| ObjectLockLegalHoldStatus | 表示指定对象是否设置依法保留配置。取值为ON或OFF | \*string | 否 |
| ObjectLockMode | 表示指定对象的保留期限模式。取值为取值为GOVERNANCE或COMPLIANCE | \*string | 否 |
| ObjectLockRetainUntilDate | 对象锁定保留期限过期日期 | \*time.Time | 否 |
| Tagging | 标签集容器，内部为标签列表，最多50个标签，每个标签都是键值对，key最大128字节，value最大256字节，value可以为空，key和value均为utf-8编码 | \*string | 否 |

返回结果说明

* PutObjectOutput 类型是个struct，具体定义如下：

type PutObjectOutput struct {

AppendPosition \*int64 `location:"header" locationName:"x-rgw-next-append-position" type:"integer"`

ETag \*string `location:"header" locationName:"ETag" type:"\*string"`

VersionId \*string `location:"header" locationName:"x-amz-version-id"

type:"\*string"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| AppendPosition | 下一次请求时的追加位置 | \*int64 |
| ETag | 上传对象的Etag | \*string |
| VersionId | 上传对象的VersionId | \*string |

示例

package main

import (

"fmt"

"io"

"os"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

file, err := os.OpenFile("PutObject.go", os.O\_RDONLY, 0666)

if err != nil {

fmt.Println("Open file failed")

return

}

readSeek := io.ReadSeekCloser(file)

input := &s3.PutObjectInput{

Bucket: aws.\*string("java-sdk-bucket"),

Key: aws.\*string("go-sdk-object"),

ACL: aws.\*string(s3.BucketCannedACLPrivate),

Body: readSeek,

}

result, err := svc.PutObject(input)

if err != nil {

fmt.Println("request fail")

fmt.Println(err)

return

}

fmt.Println("request success")

fmt.Println(result)

}

2.4、Delete Object

功能说明

Delete Object 请求可以将一个文件（Object）删除。

方法原型

func (c \*S3) DeleteObject(input \*DeleteObjectInput) (\*DeleteObjectOutput

, error)

参数说明

* input：类型DeleteObjectInput，具体定义如下

type DeleteObjectInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

BypassGovernanceRetention \*bool `location:"header" locationName:"x-amz-bypass-governance-retention" type:"boolean"`

Key \*string `location:"uri" locationName:"Key" min:"1" type:"\*string" required:"true"`

VersionId \*string `location:"query\*string" locationName:"versionId" type:"\*string"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket的名称 | \*string | 是 |
| BypassGovernanceRetention | 执行操作时是否绕过Governance模式锁的限制 | \*bool | 否 |
| Key | 要删除的对象的名称 | \*string | 是 |
| VersionId | 要删除的对象的VersionId | \*string | 否 |

返回结果说明

* DeleteObjectOutput 是一个struct，具体定义如下

type DeleteObjectOutput struct {

DeleteMarker \*bool `location:"header" locationName:"x-amz-delete-marker" type:"boolean"`

VersionId \*string `location:"header" locationName:"x-amz-version-id" type:"\*string"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| DeleteMarker | 指明删除的对象是否是DeleteMarker | \*bool |
| VersionId | 如果创建了DeleteMarker，则可通过此字段获取DeleteMarker的VersionId | \*string |

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

input := &s3.DeleteObjectInput{

Bucket: aws.\*string("java-sdk-bucket"),

Key: aws.\*string("go-sdk-object"),

}

result, err := svc.DeleteObject(input)

if err != nil {

fmt.Println("request fail")

fmt.Println(err)

return

}

fmt.Println("request success")

fmt.Println(result)

}

2.5、Delete Objects

功能说明

Delete Objects 请求实现批量删除文件，最大支持单次删除 1000 个文件。对于返回结果，COS 提供 Verbose 和 Quiet 两种结果模式。Verbose 模式将返回每个 Object 的删除结果；Quiet 模式只返回报错的 Object 信息。

方法原型

func (c \*S3) DeleteObjects(input \*DeleteObjectsInput) (\*DeleteObjectsOutput, error)

参数说明

* intput:类型是DeleteObjectsInput，具体定义如下

type DeleteObjectsInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

BypassGovernanceRetention \*bool `location:"header" locationName:"x-amz-bypass-governance-retention" type:"boolean"`

Delete \*Delete `locationName:"Delete" type:"structure" required:"true" xmlURI:"http:

}

Delete是一个struct，具体定义如下：

type Delete struct {

Objects []\*ObjectIdentifier `locationName:"Object" type:"list" flattened:"true" required:"true"`

Quiet \*bool `type:"boolean"`

}

ObjectIdentifier 是要给struct，具体定义如下：

type ObjectIdentifier struct {

Key \*string `min:"1" type:"\*string" required:"true"`

VersionId \*string `type:"\*string"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket的名称 | \*string | 是 |
| BypassGovernanceRetention | 执行操作时是否绕过Governance模式锁的限制 | \*bool | 否 |
| Delete | 要删除对象的配置 | \*Delete | 是 |
| Objects | Delete集合中描述对象信息的集合 | []\*ObjectIdentifier | 是 |
| Quiet | 设置是否开启quiet模式 | \*bool | 否 |
| Key | 要删除的对象的名称 | \*string | 是 |
| VersionId | 要删除的对象的VersionId | \*string | 否 |

返回结果说明

* DeleteObjectsOutput是一个struct,具体定义如下：

type DeleteObjectsOutput struct {

Deleted []\*DeletedObject `type:"list" flattened:"true"`

Errors []\*Error `locationName:"Error" type:"list" flattened:"true"`

}

DeletedObject是一个struct，具体定义如下：

type DeletedObject struct {

DeleteMarkerVersionId \*string `type:"\*string"`

Key \*string `min:"1" type:"\*string"`

VersionId \*string `type:"\*string"`

}

Errors是一个struct，具体定义如下：

type Error struct {

Code \*string `type:"\*string"`

Key \*string `min:"1" type:"\*string"`

Message \*string `type:"\*string"`

VersionId \*string `type:"\*string"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| Deleted | 成功删除的对象集合 | []\*DeletedObject |
| Errors | 删除失败的对象集合 | []\*Error |
| DeleteMarkerVersionId | 若创建了DeleteMarker，则包含DeleteMarker的VersionId | \*string |
| Key | 删除成功的对象的名称 | \*string |
| VersionId | 删除成功的对象的VersionId | \*string |
| Code | 删除失败的错误码 | \*string |
| Key | 删除失败的对象名称 | \*string |
| Message | 删除失败的描述信息 | \*string |
| VersionId | 删除失败的对象的VersionId | \*string |

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

var objects []\*s3.ObjectIdentifier

objects = append(objects, &s3.ObjectIdentifier{Key: aws.\*string("object1")})

objects = append(objects, &s3.ObjectIdentifier{Key: aws.\*string("object2")})

objects = append(objects, &s3.ObjectIdentifier{Key: aws.\*string("object3")})

quiet := true

delete := &s3.Delete{

Objects: objects,

Quiet: &quiet,

}

input := &s3.DeleteObjectsInput{

Bucket: aws.\*string("java-sdk-bucket"),

Delete: delete,

}

result, err := svc.DeleteObjects(input)

if err != nil {

fmt.Println("request fail")

fmt.Println(err)

return

}

fmt.Println("request success")

fmt.Println(result)

}

2.6、Put Object ACL

功能说明

设置Object的ACL，控制对Object的访问权限。该操作需要用户具有WRITE\_ACP权限。

有三种方式设置ACL，三种方式不可同时使用，每次只能给一种参数赋值。其中，通过ACL参数方式进行操作，是设置预定义的固定的ACL，不能针对特定用户进行授权，且该参数实现的效果，也可以借由另外两种方式实现，该参数使用请求头进行传递；AccessControlPolicy参数方式和Grant\*参数方式则可以针对特定用户进行授权，AccessControlPolicy方式通过请求体传递，而Grant\*方式通过请求头传递。三种方式都会覆盖原有ACL属性，包括对象所有者自身的权限，如需保留原有ACL属性，应将需要保留的原ACL添加到本次操作的授权中（ACL参数方式会默认将对象所有者权限设为FULL\_CONTROL，而另外两种方式则不会保留任何原ACL属性）。

方法原型

func (c \*S3) PutObjectAcl(input \*PutObjectAclInput) (\*PutObjectAclOutput, error)

参数说明

* input:类型是个struct，具体定义属性如下:

type PutObjectAclInput struct {

ACL \*string `location:"header" locationName:"x-amz-acl" type:"\*string" enum:"ObjectCannedACL"`

AccessControlPolicy \*AccessControlPolicy `locationName:"AccessControlPolicy" type:"structure" xmlURI:"http://s3.amazonaws.com/doc/2006-03-01/"`

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

GrantFullControl \*string `location:"header" locationName:"x-amz-grant-full-control" type:"\*string"`

GrantRead \*string `location:"header" locationName:"x-amz-grant-read" type:"\*string"`

GrantReadACP \*string `location:"header" locationName:"x-amz-grant-read-acp" type:"\*string"`

GrantWrite \*string `location:"header" locationName:"x-amz-grant-write" type:"\*string"`

GrantWriteACP \*string `location:"header" locationName:"x-amz-grant-write-acp" type:"\*string"`

Key \*string `location:"uri" locationName:"Key" min:"1" type:"\*string" required:"true"`

VersionId \*string `location:"query\*string" locationName:"versionId" type:"\*string"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket的名称 | \*string | 是 |
| Key | 对象名称 | \*string | 是 |
| VersionId | 对象版本号 | \*string | 否 |
| ACL | 预定义的固定ACL，  取值范围  s3.ObjectCannedACLPrivate,  s3.ObjectCannedACLPublicRead,  s3.ObjectCannedACLPublicReadWrite,  s3.ObjectCannedACLAuthenticatedRead, | \*string | ACL参数方式则必须，其他两种方式，则不能使用 |
| AccessControlPolicy | 包含授权列表和对象所有者参数 | \*AccessControlPolicy | 该方式下必须，其他两种方式下则不能使用 |
| GrantFullControl | 被授权用户可以对对象进行read, write, read ACP, and write ACP操作  以下Grant\*参数，格式都是”id=xxxx”或”emailAddress=xxxx”或者”uri=xxxx”以及他们的组合（用逗号连接） | \*string | 否 |
| GrantRead | 被授权用户可以对对象进行读操作 | \*string | 否 |
| GrantWrite | 被授权用户可以对对象进行写操作，删除或覆盖写该对象 | \*string | 否 |
| GrantReadACP | 被授权用户可以读取对象的ACL | \*string | 否 |
| GrantWriteACP | 被授权用户可以修改对象的ACL | \*string | 否 |

AccessControlPolicy:类型是个struct，具体定义属性如下:

type AccessControlPolicy struct {

Grants []\*Grant `locationName:"AccessControlList" locationNameList:"Grant" type:"list"`

Owner \*Owner `type:"structure"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Grants | 授权列表 | []\*Grant | 该方式下必须 |
| Owner | 对象所有者 | \*Owner | 该方式下必须 |

Grant:类型是个struct，具体定义属性如下:

type Grant struct {

Grantee \*Grantee `type:"structure" xmlPrefix:"xsi" xmlURI:"http://www.w3.org/2001/XMLSchema-instance"`

Permission \*string `type:"\*string" enum:"Permission"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Grantee | 被授权用户 | \*Grantee | 该方式下必须 |
| Permission | 向被授权用户授予的权限，  取值范围  s3.PermissionFullControl  s3.PermissionWrite  s3.PermissionWriteAcp  s3.PermissionRead  s3.PermissionReadAcp | \*string | 该方式下必须 |

Grantee:类型是个struct，具体定义属性如下:

type Grantee struct {

DisplayName \*string `type:"\*string"`

EmailAddress \*string `type:"\*string"`

ID \*string `type:"\*string"`

Type \*string `locationName:"xsi:type" type:"\*string" xmlAttribute:"true" required:"true" enum:"Type"`

URI \*string `type:"\*string"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Type | 被授权用户类型，  取值范围  s3.TypeCanonicalUser  s3.TypeAmazonCustomerByEmail  s3.TypeGroup | \*string | 该方式下必须 |
| ID | 被授权用户ID | \*string | Type为'CanonicalUser'，则该字段必须 |
| DisplayName | 被授权用户display name | \*string | 否 |
| EmailAddress | 被授权用户邮箱 | \*string | 如果Type为'AmazonCustomerByEmail'，则该字段必须 |
| URI | 被授权组URI  取值范围为  所有用户：http://acs.amazonaws.com/groups/global/AllUsers  所有认证用户：http://acs.amazonaws.com/groups/global/AuthenticatedUsers | \*string | 如果Type为'Group'，则该字段必须 |

Owner:类型是个struct，具体定义属性如下:

type Owner struct {

DisplayName \*string `type:"\*string"`

ID \*string `type:"\*string"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| ID | 对象所有者ID | \*string | 该方式下必须 |
| DisplayName | 对象所有者display name | \*string | 否 |

返回结果说明

* 正常输出类型为PutObjectAclOutput，否则返回error，

PutObjectAclOutput是struct类型，定义如下:

type PutObjectAclOutput struct {

\_ struct{} `type:"structure"`

}

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

resp, err := svc.PutObjectAcl(&s3.PutObjectAclInput{

Bucket: aws.\*string("bucket-1"),

Key: aws.\*string("test-file"),

// ACL: aws.\*string(s3.ObjectCannedACLPublicRead),

// GrantReadACP: aws.\*string("id=test-2,emailAddress=abc@abc.com"),

AccessControlPolicy: &s3.AccessControlPolicy{

Owner: &s3.Owner{

ID: aws.\*string("test-1"),

},

Grants: []\*s3.Grant{

{

Grantee: &s3.Grantee{

Type: aws.\*string(s3.TypeCanonicalUser),

ID: aws.\*string("test-2"),

},

Permission: aws.\*string(s3.PermissionFullControl),

},

{

Grantee: &s3.Grantee{

Type: aws.\*string(s3.TypeAmazonCustomerByEmail),

EmailAddress: aws.\*string("def@def.com"),

},

Permission: aws.\*string(s3.PermissionFullControl),

},

},

},

})

if err != nil {

fmt.Println("failed", err)

} else {

fmt.Println("success", resp)

}

}

2.7、Get Object ACL

功能说明

获取指定Object的 ACL。该操作需要READ\_ACP权限。该功能返回的结果与Put Object ACL参数一致，但是需要注意的是，如果以邮箱类型授权，返回结果中将会以对应被授权用户ID形式出现，即Type不会是AmazonCustomerByEmail，而是CanonicalUser。

方法原型

func (c \*S3) GetObjectAcl(input \*GetObjectAclInput) (\*GetObjectAclOutput

, error)

参数说明

* input:类型是个struct，定义如下:

type GetObjectAclInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

Key \*string `location:"uri" locationName:"Key" min:"1" type:"\*string" required:"true"`

VersionId \*string `location:"query\*string" locationName:"versionId" type:"\*string"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket的名称 | \*string | 是 |
| Key | 对象名称 | \*string | 是 |
| VersionId | 对象版本号 | \*string | 否 |

返回结果说明

* 正常输出类型为GetObjectAclOutput，否则返回error，

GetObjectAclOutput是struct类型，定义如下:

type GetObjectAclOutput struct {

Grants []\*Grant `locationName:"AccessControlList" locationNameList:"Grant" type:"list"`

Owner \*Owner `type:"structure"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| Grants | 授权列表 | []\*Grant |
| Owner | Object所有者 | \*Owner |

Grant:类型是个struct，具体定义属性如下:

type Grant struct {

Grantee \*Grantee `type:"structure" xmlPrefix:"xsi" xmlURI:"http://www.w3.org/2001/XMLSchema-instance"`

Permission \*string `type:"\*string" enum:"Permission"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| Grantee | 被授权用户 | \*Grantee |
| Permission | 被授权权限 | \*string |

Grantee:类型是个struct，具体定义属性如下:

type Grantee struct {

DisplayName \*string `type:"\*string"`

EmailAddress \*string `type:"\*string"`

ID \*string `type:"\*string"`

Type \*string `locationName:"xsi:type" type:"\*string" xmlAttribute:"true" required:"true" enum:"Type"`

URI \*string `type:"\*string"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| DisplayName | 被授权用户的展示名 | \*string |
| EmailAddress | 被授权用户邮箱 | \*string |
| ID | 被授权用户ID | \*string |
| Type | 被授权用户类型 | \*string |
| URI | 被授权组uri | \*string |

Owner:类型是个struct，具体定义属性如下:

type Owner struct {

DisplayName \*string `type:"\*string"`

ID \*string `type:"\*string"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| DisplayName | Owner 的展示名 | \*string |
| ID | Owner的 ID | \*string |

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

resp, err := svc.GetObjectAcl(&s3.GetObjectAclInput{

Bucket: aws.\*string("bucket-1"),

Key: aws.\*string("test-file"),

})

if err != nil {

fmt.Println("failed", err)

} else {

fmt.Println("success", resp)

}

}

2.8、Put Object Tagging

功能说明

将提供的标签集设置为存储桶中已存在的对象。标签是一个键值对。请注意，Amazon S3 将标签的最大数量限制为每个对象 10 个标签。要使用此操作，您必须具有执行 s3:PutObjectTagging 操作的权限。 默认情况下，Bucket 拥有者拥有此权限，并且可以将此权限授予其他人。要放置任何其他版本的标签，请使用 versionId 查询参数。 您还需要 s3:PutObjectVersionTagging 操作的权限。

方法原型

func (c \*S3) PutObjectTagging(input \*PutObjectTaggingInput) (\*PutObjectTaggingOutput, error)

参数说明

* input:类型是个struct，定义如下:

type PutObjectTaggingInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

Key \*string `location:"uri" locationName:"Key" min:"1" type:"\*string" required:"true"`

Tagging \*Tagging `locationName:"Tagging" type:"structure" required:"true" xmlURI:"http://s3.amazonaws.com/doc/2006-03-01/"`

VersionId \*string `location:"query\*string" locationName:"versionId" type:"\*string"`

}

Tagging:类型是个struct，定义如下:

type Tagging struct {

TagSet []\*Tag `locationNameList:"Tag" type:"list" required:"true"`

}

Tag:类型是个struct，定义如下:

type Tag struct {

Key \*string `min:"1" type:"\*string" required:"true"`

Value \*string `type:"\*string" required:"true"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket的名称 | \*string | 是 |
| Key | 对象名称 | \*string | 是 |
| VersionId | 对象版本号 | \*string | 否 |
| Tagging | 标签集容器 | \*Tagging | 是 |
| TagSet | 标签集合 | []\*Tag | 是 |
| Key | 标签键 | \*string | 是 |
| Value | 标签值 | \*string | 是 |

返回结果说明

* 正常输出类型为PutObjectTaggingOutput，否则返回error，

PutObjectTaggingOutput是struct类型，定义如下:

type PutObjectTaggingOutput struct {

\_ struct{} `type:"structure"`

VersionId \*string `location:"header" locationName:"x-amz-version-id" type:"\*string"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| VersionId | 对象版本号 | \*string |

示例

package main

import (

"crypto/tls"

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/service/s3"

"net/http"

)

func main() {

accessKey := "Q6PG14FVG2P9DLESOP44"

secretKey := "r3mdCea11J4subPBdmHQnh1JhPlRV3P0vU29yVsI"

endPoint := "https://124.236.120.250:10443"

customTransport := &http.Transport{

TLSClientConfig: &tls.Config{InsecureSkipVerify: true},

}

customHTTPClient := &http.Client{Transport: customTransport}

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(accessKey, secretKey, ""),

Endpoint: aws.\*string(endPoint),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

HTTPClient: customHTTPClient,

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

resp, err := svc.PutObjectTagging(&s3.PutObjectTaggingInput{

Bucket: aws.\*string("bucket-002"),

Key: aws.\*string("object-002"),

VersionId: aws.\*string("yGuRzcUi-2RxqngoTLzdt1jLyKR-p1w"),

Tagging: &s3.Tagging{

TagSet: []\*s3.Tag{

{

Key: aws.\*string("key1"),

Value: aws.\*string("val1"),

},

},

},

})

if err != nil {

fmt.Println("failed", err)

} else {

fmt.Println("success", resp)

}

}

2.9、Get Object Tagging

功能说明

返回对象的标签集。要使用此操作，您必须具有执行s3:GetObjectTagging操作的权限。默认情况下，操作返回有关对象当前版本的信息。对于多版本的存储桶，您的存储桶中可以有一个对象的多个版本。此时，要检索任何其他版本的标签，请使用 versionId 查询参数。同时，您还需要s3:GetObjectVersionTagging操作的权限。

默认情况下，存储桶拥有者具有此权限，并且可以将此权限授予其他人。

方法原型

func (c \*S3) GetObjectTagging(input \*GetObjectTaggingInput) (\*GetObjectTaggingOutput, error)

参数说明

* input:类型是个struct，定义如下:

type GetObjectTaggingInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

Key \*string `location:"uri" locationName:"Key" min:"1" type:"\*string" required:"true"`

VersionId \*string `location:"query\*string" locationName:"versionId" type:"\*string"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket的名称 | \*string | 是 |
| Key | 对象名称 | \*string | 是 |
| VersionId | 对象版本号 | \*string | 否 |

返回结果说明

* 正常输出类型为GetObjectTaggingOutput，否则返回error，

GetObjectTaggingOutput是struct类型，定义如下:

type GetObjectTaggingOutput struct {

TagSet []\*Tag `locationNameList:"Tag" type:"list" required:"true"`

VersionId \*string `location:"header" locationName:"x-amz-version-id" type:"\*string"`

}

Tag:类型是个struct，定义如下:

type Tag struct {

Key \*string `min:"1" type:"\*string" required:"true"`

Value \*string `type:"\*string" required:"true"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| TagSet | 标签集合 | []\*Tag |
| VersionId | 对象版本号 | \*string |
| Key | 标签的Key | \*string |
| Value | 标准的Value | \*string |

示例

package main

import (

"crypto/tls"

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/service/s3"

"net/http"

)

func main() {

accessKey := "Q6PG14FVG2P9DLESOP44"

secretKey := "r3mdCea11J4subPBdmHQnh1JhPlRV3P0vU29yVsI"

endPoint := "124.236.120.250:10443"

customTransport := &http.Transport{

TLSClientConfig: &tls.Config{InsecureSkipVerify: true},

}

customHTTPClient := &http.Client{Transport: customTransport}

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(accessKey, secretKey, ""),

Endpoint: aws.\*string(endPoint),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(false),

HTTPClient: customHTTPClient,

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

resp, err := svc.GetObjectTagging(&s3.GetObjectTaggingInput{

Bucket: aws.\*string("bucket-002"),

Key: aws.\*string("object-002"),

VersionId: aws.\*string("yGuRzcUi-2RxqngoTLzdt1jLyKR-p1w"),

})

if err != nil {

fmt.Println("failed", err)

} else {

fmt.Println("success", resp)

}

}

2.10、Delete Object Tagging

功能说明

从指定的对象中删除整个标记集。要使用此操作，您必须具有执行s3:DeleteObjectTagging操作的权限。要删除特定对象版本的标签，请在请求中添加versionId查询参数。您将需要s3:DeleteObjectVersionTagging操作的权限。

方法原型

func (c \*S3) DeleteObjectTagging(input \*DeleteObjectTaggingInput) (\*DeleteObjectTaggingOutput, error)

参数说明

* input:类型是个struct，定义如下:

type DeleteObjectTaggingInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

Key \*string `location:"uri" locationName:"Key" min:"1" type:"\*string" required:"true"`

VersionId \*string `location:"query\*string" locationName:"versionId" type:"\*string"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket的名称 | \*string | 是 |
| Key | 对象名称 | \*string | 是 |
| VersionId | 对象版本号 | \*string | 否 |

返回结果说明

* 正常输出类型为DeleteObjectTaggingOutput，否则返回error，

DeleteObjectTaggingOutput是struct类型，定义如下:

type DeleteObjectTaggingOutput struct {

\_ struct{} `type:"structure"`

VersionId \*string `location:"header" locationName:"x-amz-version-id" type:"\*string"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| VersionId | 对象版本号 | \*string |

示例

package main

import (

"crypto/tls"

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/service/s3"

"net/http"

)

func main() {

accessKey := "Q6PG14FVG2P9DLESOP44"

secretKey := "r3mdCea11J4subPBdmHQnh1JhPlRV3P0vU29yVsI"

endPoint := "124.236.120.250:10443"

customTransport := &http.Transport{

TLSClientConfig: &tls.Config{InsecureSkipVerify: true},

}

customHTTPClient := &http.Client{Transport: customTransport}

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(accessKey, secretKey, ""),

Endpoint: aws.\*string(endPoint),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(false),

HTTPClient: customHTTPClient,

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

resp, err := svc.DeleteObjectTagging(&s3.DeleteObjectTaggingInput{

Bucket: aws.\*string("bucket-002"),

Key: aws.\*string("object-002"),

VersionId: aws.\*string("yGuRzcUi-2RxqngoTLzdt1jLyKR-p1w"),

})

if err != nil {

fmt.Println("failed", err)

} else {

fmt.Println("success", resp)

}

}

2.11、Put Object Legal Hold

功能说明

put object legal hold请求在指定对象上使用依法保留配置

方法原型

func (c \*S3) PutObjectLegalHold(input \*PutObjectLegalHoldInput)

(\*PutObjectLegalHoldOutput, error)

参数说明

* input:类型是个struct，具体定义属性如下:

type PutObjectLegalHoldInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

Key \*string `location:"uri" locationName:"Key" min:"1" type:"\*string" required:"true"`

LegalHold \*ObjectLockLegalHold `locationName:"LegalHold" type:"structure" xmlURI:"http://s3.amazonaws.com/doc/2006-03-01/"`

VersionId \*string `location:"query\*string" locationName:"versionId" type:"\*string"`

}

ObjectLockLegalHold:类型是个struct，具体定义属性如下:

type ObjectLockLegalHold struct {

Status \*string `type:"\*string" enum:"ObjectLockLegalHoldStatus"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket的名称 | \*string | 是 |
| Key | 进行依法保留设置的对象名称 | \*string | 是 |
| LegalHold | 指定对象依法保留配置 | \*struct | 否 |
| Status | 表示指定对象是否设置依法保留配置。取值为ON或OFF | \*string | 否 |
| VersionId | 配置依法保留的对象的版本ID | \*string | 否 |

返回结果说明

* PutObjectLegalHoldOutPut:类型是个struct，具体定义属性如下:

type PutObjectLegalHoldOutPut struct {

}

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/awserr"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

lh := &s3.PutObjectLegalHoldInput{

Bucket: aws.\*string("bucket\_sdk"),

Key: aws.\*string("tuanjie.jpg"),

VersionId: aws.\*string("8qzYJQ9ehEskQdA43qCuHKugA5KYtrL"),

LegalHold: &s3.ObjectLockLegalHold{

Status: aws.\*string("OFF"),

},

}

resp, err := svc.PutObjectLegalHold(lh)

if err != nil {

if aerr, ok := err.(awserr.Error); ok {

switch aerr.Code() {

default:

fmt.Println(aerr.Error())

}

} else {

fmt.Println(err.Error())

}

} else {

fmt.Println("success", resp)

}

}

2.12、Get Object Legal Hold

功能说明

get object legal hold请求获取指定对象的当前依法保留状态

方法原型

func (c \*S3) GetObjectLegalHold(input \*GetObjectLegalHoldInput) (\*GetObjectLegalHoldOutput, error)

参数说明

* input:类型是个struct，具体定义属性如下:

type GetObjectLegalHoldInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

Key \*string `location:"uri" locationName:"Key" min:"1" type:"\*string" required:"true"`

VersionId \*string `location:"query\*string" locationName:"versionId" type:"\*string"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket的名称 | \*string | 是 |
| Key | 进行依法保留设置的对象名称 | \*string | 是 |
| VersionId | 配置依法保留的对象的版本ID | \*string | 否 |

返回结果说明

* GetObjectLegalHoldOutPut:类型是个struct，具体定义属性如下:

type GetObjectLegalHoldOutPut struct {

LegalHold \*ObjectLockLegalHold `type:"structure"`

}

ObjectLockLegalHold:类型是个struct，具体定义属性如下:

type ObjectLockLegalHold struct {

Status \*string `type:"\*string" enum:"ObjectLockLegalHoldStatus"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| LegalHold | 指定对象依法保留配置 | \*struct |
| Status | 表示指定对象是否设置依法保留配置。取值为ON或OFF | \*string |

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/awserr"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

lh := &s3.GetObjectLegalHoldInput{

Bucket: aws.\*string("bucket\_sdk"),

Key: aws.\*string("tuanjie.jpg"),

VersionId: aws.\*string("8qzYJQ9ehEskQdA43qCuHKugA5KYtrL"),

}

resp, err := svc.GetObjectLegalHold(lh)

if err != nil {

if aerr, ok := err.(awserr.Error); ok {

switch aerr.Code() {

default:

fmt.Println(aerr.Error())

}

} else {

fmt.Println(err.Error())

}

} else {

fmt.Println("success", resp)

}

}

2.13、Put Object Retention

功能说明

put object retention请求设置对象保留期限配置。

方法原型

func (c \*S3) PutObjectRetention(input \*PutObjectRetentionInput) (\*PutObjectRetentionOutput, error)

参数说明

* input:类型是个struct，具体定义属性如下:

type PutObjectRetentionInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

Key \*string `location:"uri" locationName:"Key" min:"1" type:"\*string" required:"true"`

BypassGovernanceRetention \*bool `location:"header" locationName:"x-amz-bypass-governance-retention" type:"boolean"`

Retention \*ObjectLockRetention `locationName:"Retention" type:"structure" xmlURI:"http://s3.amazonaws.com/doc/2006-03-01/"`

VersionId \*string `location:"query\*string" locationName:"versionId" type:"\*string"`

}

ObjectLockRetention:类型是个struct，具体定义属性如下:

type ObjectLockRetention struct {

Mode \*string `type:"\*string" enum:"ObjectLockRetentionMode"`

RetainUntilDate \*time.Time `type:"timestamp" timestampFormat:"iso8601"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket的名称 | \*string | 是 |
| Key | 进行依法保留设置的对象名称 | \*string | 是 |
| Retention | 对象保留期限配置元素 | \*struct | 否 |
| Mode | 表示指定对象的保留期限模式。取值为取值为GOVERNANCE或COMPLIANCE | \*string | 否 |
| RetainUntilDate | 对象锁定保留期限过期日期 | \*timestamp | 否 |
| VersionId | 配置依法保留的对象的版本ID | \*string | 否 |
| BypassGovernanceRetention | 表示是否这个操作应该绕过监管模式 | \*boolean | 否 |

返回结果说明

* PutObjectRetentionOutPut:类型是个struct，具体定义属性如下:

type PutObjectRetentionOutPut struct {

}

示例

package main

import (

"time"

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/awserr"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

retentiontime, \_ := time.Parse(time.RFC3339,"2021-06-16T17:36:05Z")

lh := &s3.PutObjectRetentionInput{

Bucket: aws.\*string("bucket\_sdk"),

Key: aws.\*string("tuanjie.jpg"),

VersionId: aws.\*string("8qzYJQ9ehEskQdA43qCuHKugA5KYtrL"),

Retention: &s3.ObjectLockRetention{

Mode: aws.\*string("COMPLIANCE"),

RetainUntilDate: aws.Time(retentiontime),

},

}

resp, err := svc.PutObjectRetention(lh)

if err != nil {

if aerr, ok := err.(awserr.Error); ok {

switch aerr.Code() {

default:

fmt.Println(aerr.Error())

}

} else {

fmt.Println(err.Error())

}

} else {

fmt.Println("success", resp)

}

}

2.14、Get Object Retention

功能说明

get object retention获取对象的保留期限设置

方法原型

func (c \*S3) GetObjectRetention(input \*GetObjectRetentionInput) (\*GetObjectRetentionOutput, error)

参数说明

* input:类型是个struct，具体定义属性如下:

type GetObjectRetentionInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

Key \*string `location:"uri" locationName:"Key" min:"1" type:"\*string" required:"true"`

VersionId \*string `location:"query\*string" locationName:"versionId" type:"\*string"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | Bucket的名称 | \*string | 是 |
| Key | 进行依法保留设置的对象名称 | \*string | 是 |
| VersionId | 配置依法保留的对象的版本ID | \*string | 否 |

返回结果说明

* GetObjectRetentionOutPut:类型是个struct，具体定义属性如下:

type GetObjectLegalHoldOutPut struct {

Retention \*ObjectLockRetention `type:"structure"`

}

ObjectLockRetention:类型是个struct，具体定义属性如下:

type ObjectLockRetention struct {

Mode \*string `type:"\*string" enum:"ObjectLockRetentionMode"`

RetainUntilDate \*time.Time `type:"timestamp" timestampFormat:"iso8601"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| Retention | 对象保留期限配置元素 | \*struct |
| Mode | 表示指定对象的保留期限模式。取值为GOVERNANCE或COMPLIANCE | \*string |
| RetainUntilDate | 对象锁定保留期限过期日期 | \*timestamp |

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/awserr"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

lh := &s3.GetObjectRetentionInput{

Bucket: aws.\*string("bucket\_sdk"),

Key: aws.\*string("tuanjie.jpg"),

VersionId: aws.\*string("8qzYJQ9ehEskQdA43qCuHKugA5KYtrL"),

}

resp, err := svc.GetObjectRetention(lh)

if err != nil {

if aerr, ok := err.(awserr.Error); ok {

switch aerr.Code() {

default:

fmt.Println(aerr.Error())

}

} else {

fmt.Println(err.Error())

}

} else {

fmt.Println("success", resp)

}

}

2.17、CopyObject

功能说明

CopyObject 请求实现将一个文件从源路径复制到目标路径。

方法原型

func (c \*S3) CopyObject(input \*CopyObjectInput) (\*CopyObjectOutput, error)

参数说明

* input：类型CopyObjectInput，是个struct，具体定义如下

type CopyObjectInput struct {

ACL \*string `location:"header" locationName:"x-amz-acl" type:"\*string" enum:"ObjectCannedACL"`

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

CopySource \*string `location:"header" locationName:"x-amz-copy-source" type:"\*string" required:"true"`

CopySourceIfMatch \*string `location:"header" locationName:"x-amz-copy-source-if-match" type:"\*string"`

CopySourceIfModifiedSince \*time.Time `location:"header" locationName:"x-amz-copy-source-if-modified-since" type:"timestamp"`

CopySourceIfNoneMatch \*string `location:"header" locationName:"x-amz-copy-source-if-none-match" type:"\*string"`

CopySourceIfUnmodifiedSince \*time.Time `location:"header" locationName:"x-amz-copy-source-if-unmodified-since" type:"timestamp"`

GrantFullControl \*string `location:"header" locationName:"x-amz-grant-full-control" type:"\*string"`

GrantRead \*string `location:"header" locationName:"x-amz-grant-read" type:"\*string"`

GrantReadACP \*string `location:"header" locationName:"x-amz-grant-read-acp" type:"\*string"`

GrantWriteACP \*string `location:"header" locationName:"x-amz-grant-write-acp" type:"\*string"`

Key \*string `location:"uri" locationName:"Key" min:"1" type:"\*string" required:"true"`

Metadata map[\*string]\*string `location:"headers" locationName:"x-amz-meta-" type:"map"`

MetadataDirective \*string `location:"header" locationName:"x-amz-metadata-directive" type:"\*string" enum:"MetadataDirective"`

ObjectLockLegalHoldStatus \*string `location:"header" locationName:"x-amz-object-lock-legal-hold" type:"\*string" enum:"ObjectLockLegalHoldStatus"`

ObjectLockMode \*string `location:"header" locationName:"x-amz-object-lock-mode" type:"\*string" enum:"ObjectLockMode"`

ObjectLockRetainUntilDate \*time.Time `location:"header" locationName:"x-amz-object-lock-retain-until-date" type:"timestamp" timestampFormat:"iso8601"`

Tagging \*string `location:"header" locationName:"x-amz-tagging" type:"\*string"`

TaggingDirective \*string `location:"header" locationName:"x-amz-tagging-directive" type:"\*string" enum:"TaggingDirective"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| ACL | 复制目的对象的ACL | \*string | 否 |
| Bucket | 复制目的Bucket | \*string | 是 |
| CopySource | 复制源，格式为 "Bucket/Key" | \*string | 是 |
| CopySourceIfMatch | 仅当指定的Etag和Copy Source指定的object的Etag匹配时才复制 | \*string | 否 |
| CopySourceIfModifiedSince | 仅当CopySource在指定时间后更新过才复制 | \*time.Time | 否 |
| CopySourceIfNoneMatch | 仅当指定的Etag和Copy Source指定的object的Etag不匹配时才复制 | \*string | 否 |
| CopySourceIfUnmodifiedSince | 仅当CopySource在指定时间后未更新过才复制 | \*time.Time | 否 |
| GrantFullControl | 被授权用户可以对对象进行read, write, read ACP, and write ACP操作  以下Grant\*参数，格式都是”id=xxxx”或”emailAddress=xxxx”或者”uri=xxxx”以及他们的组合（用逗号连接） | \*string | 否 |
| GrantRead | 被授权用户可以对对象进行读操作 | \*string | 否 |
| GrantReadACP | 被授权用户可以读取对象的ACL | \*string | 否 |
| GrantWriteACP | 被授权用户可以修改对象的ACL | \*string | 否 |
| Key | 复制目的Object的名字 | \*string | 是 |
| Metadata | 复制目的对象的元数据 | map[\*string]\*string | 否 |
| MetadataDirective | Metadata复制选项，'COPY' 复制源的Metadata或 'REPLACE'使用新指定的Metadata覆盖 | \*string | 否 |
| ObjectLockLegalHoldStatus | 设置复制的Object 的LegalHold 开关 | \*string | 否 |
| ObjectLockMode | 设置复制的Object 的 Retention 模式 | \*string | 否 |
| ObjectLockRetainUntilDate | 设置复制的Object 的 Retention 保留过期时间 | \*time.Time | 否 |
| Tagging | 设置复制对象的Tag | \*string | 否 |
| TaggingDirective | Tagging 复制选项 'COPY' 复制源的Tag或 'REPLACE'使用新指定的Tag覆盖 | \*string | 否 |

返回结果说明

* CopyObjectOutput是个struct，具体定义如下

type CopyObjectOutput struct {

CopyObjectResult \*CopyObjectResult `type:"structure"`

CopySourceVersionId \*string `location:"header" locationName:"x-amz-copy-source-version-id" type:"\*string"`

VersionId \*string `location:"header" locationName:"x-amz-version-id" type:"\*string"`

}

CopyObjectResult 是个struct，具体定义如下

type CopyObjectResult struct {

ETag \*string `type:"\*string"`

LastModified \*time.Time `type:"timestamp"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| CopyObjectResult | 复制结果 | \*CopyObjectResult |
| CopySourceVersionId | 复制源对象的VersionId | \*string |
| VersionId | 复制目的对象的VersionId | \*string |
| ETag | 复制目的对象的Etag | \*string |
| LastModified | 复制目的对象的修改时间 | \*time.Time |

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

input := &s3.CopyObjectInput{

Bucket: aws.\*string("java-sdk-bucket"),

Key: aws.\*string("go-copy-object"),

CopySource: aws.\*string("java-sdk-bucket/txt\_copy"),

}

result, err := svc.CopyObject(input)

if err != nil {

fmt.Println("request fail")

fmt.Println(err)

return

}

fmt.Println("request success")

fmt.Println(result)

}

2.18、UploadPartCopy

功能说明

Copy 请求实现将一个文件从源路径复制到目标路径，支持拷贝大于5GB的文件。

方法原型

func (c \*S3) UploadPartCopy(input \*UploadPartCopyInput) (\*UploadPartCopyOutput, error)

参数说明

* input：类型是UploadPartCopyInput， 是一个struct，具体定义如下

type UploadPartCopyInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

CopySource \*string `location:"header" locationName:"x-amz-copy-source" type:"\*string" required:"true"`

CopySourceIfMatch \*string `location:"header" locationName:"x-amz-copy-source-if-match" type:"\*string"`

CopySourceIfModifiedSince \*time.Time `location:"header" locationName:"x-amz-copy-source-if-modified-since" type:"timestamp"`

CopySourceIfNoneMatch \*string `location:"header" locationName:"x-amz-copy-source-if-none-match" type:"\*string"`

CopySourceIfUnmodifiedSince \*time.Time `location:"header" locationName:"x-amz-copy-source-if-unmodified-since" type:"timestamp"`

CopySourceRange \*string `location:"header" locationName:"x-amz-copy-source-range" type:"\*string"`

Key \*string `location:"uri" locationName:"Key" min:"1" type:"\*string" required:"true"`

PartNumber \*int64 `location:"query\*string" locationName:"partNumber" type:"integer" required:"true"`

UploadId \*string `location:"query\*string" locationName:"uploadId" type:"\*string" required:"true"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | 复制目的Bucket | \*string | 是 |
| CopySource | 复制源，格式为 "Bucket/Key" | \*string | 是 |
| CopySourceIfMatch | 仅当指定的Etag和Copy Source指定的object的Etag匹配时才复制 | \*string | 否 |
| CopySourceIfModifiedSince | 仅当CopySource在指定时间后更新过才复制 | \*time.Time | 否 |
| CopySourceIfNoneMatch | 仅当指定的Etag和Copy Source指定的object的Etag不匹配时才复制 | \*string | 否 |
| CopySourceIfUnmodifiedSince | 仅当CopySource在指定时间后未更新过才复制 | \*time.Time | 否 |
| CopySourceRange | 复制对象的字节范围，格式为 "bytes={start}-{end}" | \*string | 否 |
| Key | 复制目的Object的名字 | \*string | 是 |
| PartNumber | 设置本次复制的分段编号 | \*string | 是 |
| UploadId | 设置分段上传ID，由3.1返回 | \*string | 是 |

返回结果说明

* UploadPartCopyOutput 是一个struct,具体定义如下

type UploadPartCopyOutput struct {

CopyPartResult \*CopyPartResult `type:"structure"`

CopySourceVersionId \*string `location:"header" locationName:"x-amz-copy-source-version-id" type:"\*string"`

}

CopyPartResult 是一个struct，具体定义如下

type CopyPartResult struct {

ETag \*string `type:"\*string"`

LastModified \*time.Time `type:"timestamp"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| CopyPartResult | 复制结果 | \*CopyPartResult |
| CopySourceVersionId | 复制源对象的VersionId | \*string |
| ETag | 复制目的对象的Etag | \*string |
| LastModified | 复制目的对象的修改时间 | \*time.Time |

示例

package main

import (

"fmt"

"strconv"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/service/s3"

)

func min(a int64, b int64) int64 {

if a < b {

return a

}

return b

}

func getCompletedPart(resultList []\*s3.CopyPartResult) []\*s3.CompletedPart {

var result []\*s3.CompletedPart

for index, val := range resultList {

var partNumber int64 = int64(index + 1)

result = append(result, &s3.CompletedPart{

PartNumber: &partNumber,

ETag: val.ETag,

})

}

return result

}

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

sourceBucket := aws.\*string("java-sdk-bucket")

sourceKey := aws.\*string("zos-sdk-cpp.tar.gz")

destBucket := aws.\*string("java-sdk-bucket")

destKey := aws.\*string("dest-obj")

sourceBucketKey := \*sourceBucket + "/" + \*sourceKey

// 获取复制对象信息

metadataInput := &s3.HeadObjectInput{

Bucket: sourceBucket,

Key: sourceKey,

}

metaResult, err := svc.HeadObject(metadataInput)

if err != nil {

fmt.Println("request fail")

fmt.Println(err)

return

}

length := metaResult.ContentLength

// 5M 一个分段复制

var partSize int64 = 1024 \* 1024 \* 5

var bytePosition int64 = 0

var partNum int64 = 1

// 初始化分段上传

initInput := &s3.CreateMultipartUploadInput{

Bucket: destBucket,

Key: destKey,

}

initResult, err := svc.CreateMultipartUpload(initInput)

if err != nil {

fmt.Println("request fail")

fmt.Println(err)

return

}

uploadId := initResult.UploadId

// 分段复制

var copyResultList []\*s3.CopyPartResult

for {

if bytePosition >= \*length {

break

}

lastByte := min(\*length-1, bytePosition+partSize-1)

strRange := "bytes=" + strconv.FormatInt(bytePosition, 10) + "-" + strconv.FormatInt(lastByte, 10)

input := &s3.UploadPartCopyInput{

Bucket: destBucket,

Key: destKey,

CopySource: &sourceBucketKey,

PartNumber: &partNum,

UploadId: uploadId,

CopySourceRange: &strRange,

}

result, err := svc.UploadPartCopy(input)

if err != nil {

fmt.Println("request fail")

fmt.Println(err)

break

}

copyResultList = append(copyResultList, result.CopyPartResult)

partNum++

bytePosition += partSize

}

// 完成分段上传

completeInput := &s3.CompleteMultipartUploadInput{

Bucket: destBucket,

Key: destKey,

UploadId: uploadId,

MultipartUpload: &s3.CompletedMultipartUpload{

Parts: getCompletedPart(copyResultList),

},

}

completeResult, err := svc.CompleteMultipartUpload(completeInput)

if err != nil {

fmt.Println("request fail")

fmt.Println(err)

return

}

fmt.Println("request success")

fmt.Println(completeResult)

}

2.19、Metadata Search

功能说明

搜索请求实现通过指定文件属性筛选得到想要的文件。

方法原型

func (c \*S3) MdSearch(input \*MdSearchInput) (\*MdSearchOutput, error)

参数说明

* input:类型\*s3.MdSearchInput，获取Bucket Lifecycle请求接口参数，定义的方法如下:

type MdSearchInput struct {

Marker \*int64 `location:"query\*string" locationName:"Marker" type:"integer"`

Max\_keys \*int64 `location:"query\*string" locationName:"max-keys" type:"integer"`

Qmode \*string `location:"query\*string" locationName:"qmode" type:"\*string"`

Qorder \*string `location:"query\*string" locationName:"qorder" type:"\*string"`

Qorderkey \*string `location:"query\*string" locationName:"qorderkey" type:"\*string"`

Query \*string `location:"query\*string" locationName:"query" type:"\*string"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Query | 搜索请求,格式[(]<arg> <op> <value> [)][<and|or> …]  arg的可选值包括bucket、name、instance（版本）、size、mtime、storage\_class、content\_type、versioned\_epoch;  op的可选值包括’<, *<=, ==, >=, >*’  e.g. query=’(name==hudie)and(bucket=bucket1)’ | \*string | 是 |
| Qmode | 搜索模式。可选值term(精确),fuzzy(模糊),wild(通配符),regex(正则)  默认值：term | \*string | 否 |
| Qorder | 结果排序规则。  asc :升序（默认）  desc: 降序 | \*string | 否 |
| Qorderkey | 搜索结果排序字段  默认：name | \*string | 否 |
| Max\_keys | 返回搜索结果数量  默认：100， 不能超过10000 | \*int64 | 否 |
| Marker | 返回搜索结果开始的位置，即从第几个结果开始返回 | \*int64 | 否 |

返回结果说明

{

Marker: \*string,

Contents: [{

ContentType: \*string,

LastModified: \*time.Time,

Bucket: \*string,

Instance: \*string,

ETag: \*string,

VersionedEpoch: \*int64,

StorageClass: \*string,

Key: \*string,

CustomMetadata: {},

Owner: {

DisplayName: \*string,

ID: \*string

},

Size: \*int64

}],

IsTruncated: \*string

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| Marker | 返回搜索结果开始的位置 | \*string |
| Contents | 搜索结果的内容 | array |
| IsTruncated | 是否有截断 | \*bool |

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/awserr"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://127.0.0.1:8002"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

input := &s3.MdSearchInput{

Query: aws.\*string("key==hudie\*"),

Qmode: aws.\*string("wild"),

}

result, err := svc.MdSearch(input)

if err != nil {

if aerr, ok := err.(awserr.Error); ok {

switch aerr.Code() {

default:

fmt.Println(aerr.Error())

}

} else {

fmt.Println(err.Error())

}

}

fmt.Print(result)

}

3、分块上传操作

3.1、Create Multipart Upload

功能说明

Create Multipart Upload 请求实现初始化分片上传，成功执行此请求以后会返回 Upload ID 用于后续的 Upload Part 请求。

方法原型

func (c \*S3) CreateMultipartUpload(input \*CreateMultipartUploadInput) (\*CreateMultipartUploadOutput, error)

参数说明

* input:类型是个struct，具体定义属性如下:

type CreateMultipartUploadInput struct {

ACL \*string `location:"header" locationName:"x-amz-acl" type:"\*string" enum:"ObjectCannedACL"`

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

Key \*string `location:"uri" locationName:"Key" min:"1" type:"\*string" required:"true"`

Tagging \*string `location:"header" locationName:"x-amz-tagging" type:"\*string"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| ACL | 要上传文件的ACL规则，可选类型有private、public-read、public-read-write和authenticated-read | \*string | 否 |
| Bucket | 文件要上传的Bucket名称 | \*string | 是 |
| Key | 要上传的文件名称 | \*string | 是 |
| Tagging | 文件的标签，字符串要类似于这种Key1=Value1，而且只能有一个=符号，=之前为key，之后为value，所有即使有多个=也会解析为value | \*string | 否 |

返回结果说明

* CreateMultipartUploadOutput:类型是个struct，具体定义属性如下:

type CreateMultipartUploadOutput struct {

Bucket \*string `locationName:"Bucket" type:"\*string"`

Key \*string `min:"1" type:"\*string"`

UploadId \*string `type:"\*string"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| Bucket | 文件保存的Bucket名称 | \*string |
| UploadId | 分段上传ID | \*string |
| Key | 文件在Bucket中保存的名称 | \*string |

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

input := &s3.CreateMultipartUploadInput{

Bucket: aws.\*string("bucket1"),

Key: aws.\*string("30M"),

Tagging: aws.\*string("tag-key=tag-value"),

}

result, err := svc.CreateMultipartUpload(input)

if err != nil {

fmt.Println("requeset fail ")

fmt.Println(err)

return

}

fmt.Println("requeset success ")

fmt.Println(result)

}

3.2、Upload Part

功能说明

Upload Part 请求实现在初始化以后的分块上传，支持的块的数量为 1 到 10000，除了最后一块，其他每块大小都必须大于或等于5M。在每次请求 Upload Part 时，需要携带 partNumber 和 uploadID，partNumber 为块的编号，支持乱序上传。

方法原型

func (c \*S3) UploadPart(input \*UploadPartInput) (\*UploadPartOutput, error)

参数说明

* input:类型是个struct，具体定义属性如下:

type UploadPartInput struct {

Body io.ReadSeeker `type:"blob"`

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"``

Key \*string `location:"uri" locationName:"Key" min:"1" type:"\*string" required:"true"`

PartNumber \*int64 `location:"query\*string" locationName:"partNumber" type:"integer" required:"true"`

UploadId \*string `location:"query\*string" locationName:"uploadId" type:"\*string" required:"true"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Body | 要上传的文件数据 | io.ReadSeeker | 是 |
| Bucket | 文件要上传的Bucket名称 | \*string | 是 |
| Key | 要上传的文件名称 | \*string | 是 |
| PartNumber | 当前分段上传的编号 | \*string | 是 |
| UploadId | 分段上传的ID，在[3.1](#_3.1、Create_Multipart_Upload)中返回的UploadId | \*string | 是 |

返回结果说明

* UploadPartOutput:类型是个struct，具体定义属性如下:

type UploadPartOutput struct {

ETag \*string `location:"header" locationName:"ETag" type:"\*string"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| Etag | 当前分段数据的Etag | \*string |

示例

package main

import (

"fmt"

"os"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

file, err := os.Open("/root/go\_test/file-name")

if err != nil {

fmt.Println("read file fail ")

os.Exit(1)

}

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

input := &s3.UploadPartInput{

Body: file,

Bucket: aws.\*string("bucket-name"),

Key: aws.\*string("key-name"),

PartNumber: aws.Int64(1),

UploadId: aws.\*string("2~YT2JbF7LxbIWSb1vZUDQoyOr\_ueZxzG"),

}

result, err := svc.UploadPart(input)

if err != nil {

fmt.Println("requeset fail ")

fmt.Println(err)

return

}

fmt.Println("requeset success ")

fmt.Println(result)

}

3.3、Complete Multipart Upload

功能说明

Complete Multipart Upload 用来实现完成整个分块上传。当您已经使用 Upload Parts 上传所有块以后，你可以用该 API 完成上传。在使用该 API 时，您必须在 Body 中给出每一个块的 PartNumber 和 ETag，用来校验块的准确性。

方法原型

func (c \*S3) CompleteMultipartUpload(input \*CompleteMultipartUploadInput

) (\*CompleteMultipartUploadOutput, error)

参数说明

* input:类型是个struct，具体定义属性如下:

type CompleteMultipartUploadInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

Key \*string `location:"uri" locationName:"Key" min:"1" type:"\*string" required:"true"`

MultipartUpload \*CompletedMultipartUpload `locationName:"CompleteMultipartUpload" type:"structure"

UploadId \*string `location:"query\*string" locationName:"uploadId" type:"\*string" required:"true"`

}

CompletedMultipartUpload是个struct，具体定义属性如下:

type CompletedMultipartUpload struct {

Parts []\*CompletedPart `locationName:"Part" type:"list" flattened:"true"`

}

CompletedPart是个struct，具体定义属性如下:

type CompletedPart struct {

ETag \*string `type:"\*string"`

PartNumber \*int64 `type:"integer"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | 文件要上传的Bucket名称 | \*string | 是 |
| Key | 要上传的文件名称 | \*string | 是 |
| MultipartUpload | 所有成功上传的分段的分段信息，包括分段对应Etag和分段号 | CompletedMultipartUpload | 是 |
| Parts | 各个分段信息数据 | CompletedPart | 是 |
| Etag | 该分段对应Etag | \*string | 是 |
| PartNumber | 分段对应的分段编号 | \*string | 是 |
| UploadId | 分段上传的ID，在[3.1](#_3.1、Create_Multipart_Upload)中返回的UploadId | \*string | 是 |

返回结果说明

* CompleteMultipartUploadOutput:类型是个struct，具体定义属性如下:

type CompleteMultipartUploadOutput struct {

Bucket \*string `type:"\*string"`

ETag \*string `type:"\*string"`

Key \*string `min:"1" type:"\*string"`

Location \*string `type:"\*string"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| Bucket | 保存文件的Bucket名称 | \*string |
| Bucket | 整体文件的Etag | \*string |
| Key | 文件在集群中保存的名字 | \*string |
| Location | 文件具体位置 | Dict |

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

input := &s3.CompleteMultipartUploadInput{

Bucket: aws.\*string("bucket-name"),

Key: aws.\*string("key-name"),

MultipartUpload: &s3.CompletedMultipartUpload{

Parts: []\*s3.CompletedPart{

{

ETag: aws.\*string("\"5f363e0e58a95f06cbe9bbc662c5dfb6\""),

PartNumber: aws.Int64(1),

},

{

ETag: aws.\*string("\"da6a0d097e307ac52ed9b4ad551801fc\""),

PartNumber: aws.Int64(2),

},

{

ETag: aws.\*string("\"e6f9e88e31a7725d438502267ca44d8e\""),

PartNumber: aws.Int64(3),

},

},

},

UploadId: aws.\*string("2~YT2JbF7LxbIWSb1vZUDQoyOr\_ueZxzG"),

}

result, err := svc.CompleteMultipartUpload(input)

if err != nil {

fmt.Println("requeset fail ")

fmt.Println(err)

return

}

fmt.Println("requeset success ")

fmt.Println(result)

}

3.4、List Parts

功能说明

List Parts 用来查询特定分段上传中的已上传的分段的信息。

方法原型

func (c \*S3) ListParts(input \*ListPartsInput) (\*ListPartsOutput, error)

参数说明

* input:类型是个struct，具体定义属性如下:

type ListPartsInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

Key \*string `location:"uri" locationName:"Key" min:"1" type:"\*string" required:"true"`

MaxParts \*int64 `location:"query\*string" locationName:"max-parts" type:"integer"`

PartNumberMarker \*int64 `location:"query\*string" locationName:"part-number-marker" type:"integer"`

UploadId \*string `location:"query\*string" locationName:"uploadId" type:"\*string" required:"true"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | 文件要上传的Bucket名称 | \*string | 是 |
| Key | 上传文件在集群中保存的文件名称 | \*string | 是 |
| MaxParts | 最多返回的分段数目 | Int | 否 |
| PartNumberMarker | list的分段编号的起始编号，只有分段编号大于这个数字的分段信息才会返回 | Int | 否 |
| UploadId | 分段上传的ID，在[3.1](#_3.1、Create_Multipart_Upload)中返回的UploadId | \*string | 是 |

返回结果说明

* ListPartsOutput:类型是个struct，具体定义属性如下:

type ListPartsOutput struct {

Bucket \*string `type:"\*string"`

IsTruncated \*bool `type:"boolean"`

Key \*string `min:"1" type:"\*string"`

MaxParts \*int64 `type:"integer"`

NextPartNumberMarker \*int64 `type:"integer"`

Owner \*Owner `type:"structure"`

PartNumberMarker \*int64 `type:"integer"`

Parts []\*Part `locationName:"Part" type:"list" flattened:"true"`

StorageClass \*string `type:"\*string" enum:"StorageClass"`

UploadId \*string `type:"\*string"`

}

Owner是个struct，具体定义属性如下:

type Owner struct {

DisplayName \*string `type:"\*string"`

ID \*string `type:"\*string"`

}

Part类型是个struct，具体定义属性如下:

type Part struct {

ETag \*string `type:"\*string"`

LastModified \*time.Time `type:"timestamp"`

PartNumber \*int64 `type:"integer"`

Size \*int64 `type:"integer"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| Bucket | 文件上传的Bcuket名称 | \*string |
| IsTruncated | 本次请求数据是否是截断返回 | boolean |
| Key | 分段上传的文件在集群中保存的名字 | \*string |
| MaxParts | 本次请求最多可返回的分段数 | int |
| NextPartNumberMarker | 下一次list的时候的分段起始编号，主要用于截断返回时(也就是已上传的分段数目大于当前返回的分段数目)，作为下一次list的分段起始编号 | Int |
| Owner | 分段上传的文件所属用户 | Owner |
| DisplayName | 用户的DisplayName | \*string |
| ID | 用户的ID | \*string |
| PartNumberMarker | 本次请求的分段起始编号 | int |
| Parts | 已经上传的分段信息 | Part |
| Etag | 该分段数据对应的Etag | \*string |
| LastModified | 该分段上次修改时间 | Time |
| PartNumber | 该分段的分段号 | Int |
| Size | 该分段大小 | Int |
| StorageClass | 分段传的文件的存储级别 | \*string |
| UploadId | 分段上传的UploadId | \*string |

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

input := &s3.ListPartsInput{

Bucket: aws.\*string("bucket1"),

Key: aws.\*string("30M"),

MaxParts: aws.Int64(2),

PartNumberMarker:aws.Int64(0),

UploadId: aws.\*string("2~YT2JbF7LxbIWSb1vZUDQoyOr\_ueZxzG"),

}

result, err := svc.ListParts(input)

if err != nil {

fmt.Println("requeset fail ")

fmt.Println(err)

return

}

fmt.Println("requeset success ")

fmt.Println(result)

}

3.5、Abort Multipart Upload

功能说明

Abort Multipart Upload 用来实现舍弃一个分块上传并删除已上传的块。当您调用 Abort Multipart Upload 时，如果有正在使用这个 Upload Parts 上传块的请求，则 Upload Parts 会返回失败。

方法原型

func (c \*S3) AbortMultipartUpload(input \*AbortMultipartUploadInput) (\*AbortMultipartUploadOutput, error)

参数说明

* input类型是个struct，具体定义属性如下:

type AbortMultipartUploadInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

Key \*string `location:"uri" locationName:"Key" min:"1" type:"\*string" required:"true"`

UploadId \*string `location:"query\*string" locationName:"uploadId" type:"\*string" required:"true"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | 文件要上传的Bucket名称 | \*string | 是 |
| Key | 上传文件在集群中保存的文件名称 | \*string | 是 |
| UploadId | 分段上传的ID，在[3.1](#_3.1、Create_Multipart_Upload)中返回的UploadId | \*string | 是 |

返回结果说明

* AbortMultipartUploadOutput:类型是个struct，具体定义属性如下:

type AbortMultipartUploadOutput struct {

}

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

input := &s3.AbortMultipartUploadInput{

Bucket: aws.\*string("bucket-name"),

Key: aws.\*string("key-name"),

UploadId: aws.\*string("2~D3v1TwFDVHnVwvdlG8i4gEn33ni4p75"),

}

result, err := svc.AbortMultipartUpload(input)

if err != nil {

fmt.Println("requeset fail ")

fmt.Println(err)

return

}

fmt.Println("requeset success ")

fmt.Println(result)

}

3.6、List Multipart Uploads

功能说明

List Multiparts Uploads 用来查询正在进行中的分段上传，也就是已经 Created但是还没有Aaborted或者Completed的分段上传数据，单次最多列出 1000 个正在进行中的分段上传。

方法原型

func (c \*S3) ListMultipartUploads(input \*ListMultipartUploadsInput) (\*ListMultipartUploadsOutput, error)

参数说明

* input:类型是个struct，具体定义属性如下:

type ListMultipartUploadsInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

KeyMarker \*string `location:"query\*string" locationName:"key-marker" type:"\*string"`

MaxUploads \*int64 `location:"query\*string" locationName:"max-uploads" type:"integer"`

Prefix \*string `location:"query\*string" locationName:"prefix" type:"\*string"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | 文件要上传的Bucket名称 | \*string | 是 |
| KeyMarker | 只有Key大于KeyMarker的分段上传数据才会返回 | \*string | 否 |
| MaxUploads | 单次最多返回的分段上传数据，大小是1-1000，超过1000的数据会被视为1000 | Int | 否 |
| Prefix | Key的前缀，只有以Prefix为开头的Key才会被返回 | \*string | 否 |

返回结果说明

* ListMultipartUploadsOutput:类型是个strict，具体定义属性如下:

type ListMultipartUploadsOutput struct {

Bucket \*string `type:"\*string"`

IsTruncated \*bool `type:"boolean"`

KeyMarker \*string `type:"\*string"`

MaxUploads \*int64 `type:"integer"`

NextKeyMarker \*string `type:"\*string"`

Prefix \*string `type:"\*string"`

Uploads []\*MultipartUpload `locationName:"Upload" type:"list" flattened:"true"`

}

MultipartUpload是个struct，具体定义属性如下:

type MultipartUpload struct {

Initiated \*time.Time `type:"timestamp"`

Initiator \*Initiator `type:"structure"`

Key \*string `min:"1" type:"\*string"`

Owner \*Owner `type:"structure"`

StorageClass \*string `type:"\*string" enum:"StorageClass"`

UploadId \*string `type:"\*string"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **描述** | **类型** |
| Bucket | 分段上传数据所在Bucket | \*string |
| IsTruncated | 本次请求是否是截断返回 | Boolean |
| KeyMarker | 本次请求的KeyMarker | \*string |
| MaxUploads | 本次请求最多可返回的分段上传数据 | Int |
| NextKeyMarker | 下次请求的KeyMarker | \*string |
| Prefix | 本次请求的KeyMarker | \*string |
| Uploads | 分段上传数据 | MultipartUpload |
| Initiated | 分段上传的初始化时间 | Time |
| Initiator | 分段上传的创建者，类型是个struct,有 DisplayName \*string和ID \*string两个属性 | Struct |
| Key | 分段上传的Key | \*string |
| Owner | 分段上传的所有者,类型是个struct，有 DisplayName \*string和ID \*string两个属性 | Owner |
| StorageClass | 分段上传数据的存储级别 | \*string |
| UploadId | 分段上传的UploadId | \*string |

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

input := &s3.ListMultipartUploadsInput{

Bucket: aws.\*string("bucket-name"),

KeyMarker: aws.\*string("30M"),

MaxUploads: aws.Int64(5),

Prefix: aws.\*string("30M"),

}

result, err := svc.ListMultipartUploads(input)

if err != nil {

fmt.Println("requeset fail ")

fmt.Println(err)

return

}

fmt.Println("requeset success ")

fmt.Println(result)

}

4、图片处理

4.1、Post请求处理图片

功能说明

该功能是对存储桶中的图片进行处理，并将处理后的图片持久化到指定的存储桶中,支持处理图片格式JPG、PNG、WEBP、BMP、TIFF。

方法原型

func (c \*S3) ProcessObject(input \*ProcessObjectInput) (\*ProcessObjectOutput, error)

参数说明

* input:类型\*s3.ProcessObjectInput，图片处理持久化的请求接口参数，定义的方法如下:

type ProcessObjectInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

Key \*string `location:"uri" locationName:"Key" min:"1" type:"\*string" required:"true"`

ProcessSource \*string `location:"header" locationName:"x-zos-process-source" type:"\*string" required:"true"`

ZosProcess \*string `locationName:"x-zos-process" type:"\*string" required:"true"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| ProcessSource | 源图片的路径，格式{Bucket}/{Object}[?Versionid=] | \*string | 是 |
| Bucket | 指定处理后的图片要存放的存储桶 | \*string | 是 |
| Key | 处理后的图片要保存的名称 | \*string | 是 |
| ZosProces | 图片处理的方式，包含缩放，裁剪，旋转，水印，格式转换，获取信息功能 | \*string | 是 |

ZosProcess的定义方法为：

* 图片缩放(e.g. image/resize,w\_300,h\_200,m\_fixed)

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数用途** | **取值** | **是否必须** |
| w | 指定目标缩放图宽度 | [1,4096] | 使用按百分比缩放可不指定宽高 |
| h | 指定目标缩放图高度 | [1,4096] | 使用按百分比缩放可不指定宽高 |
| m | 指定缩放模式 | lfit（默认值）：等比缩放，目标缩放图为指定w和h矩形框内的最大图形。  mfit：等比缩放，目标缩放图为延伸出指定w和h矩形框外的最小图形。  fill : 将原图等比缩放为延伸出指定w与h的矩形框外的最小图片，之后将超出的部分进行居中裁剪。  pad: 将原图等比缩放为指定w和h矩形框内最大的图形，然后使用color指定的颜色将矩形框内剩余部分进行填充。  fixed: 固定宽高，强制缩放。 | 否 |
| color | 缩放模式为pad时，指定填充颜色 | RGB颜色值，默认FFFFFF(白色) | 否（仅当m为pad模式时使用） |
| p | 按百分比进行缩放 | [1,1000]  小于100缩小，大于100放大 | 否 |
| limit | 指定目标缩放图大于原图时是否缩放 | 1(默认)：目标缩放图大于原图时返回原图  0：按指定参数缩放 | 否 |

* 图片裁剪(e.g. image/crop,w\_100,h\_100,x\_10,y\_10,g\_se)

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数用途** | **取值** | **是否必须** |
| w | 指定裁剪宽度。 | [0,图片宽度]  默认为最大值。 | 否 |
| h | 指定裁剪高度。 | [0,图片高度]  默认为最大值。 | 否 |
| x | 指定裁剪起点横坐标（默认左上角为原点）。 | [0,图片边界] | 否 |
| y | 指定裁剪起点纵坐标（默认左上角为原点）。 | [0,图片边界] | 否 |
| g | 设置裁剪的原点位置。原点按照九宫格的形式分布，一共有九个位置可以设置，为每个九宫格的左上角顶点。 | nw：左上(默认)  north：中上  ne：右上  west：左中  center：中部  east：右中  sw：左下  south：中下  se：右下 | 否 |

* 图片旋转(e.g. image/rotate,45)

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数用途** | **取值** | **是否必须** |
| [value] | 图片按顺时针旋转的角度。 | [0,360]  默认值：0，表示不旋转。 | 是 |

* 水印
* 图片水印(e.g. image/watermark,image\_aHVkaWUuanBnP3gtem9zLXByb2Nlc3M9aW1hZ2UvcmVzaXplLHBfMzAvcm90YXRlLDE4MA==,g\_north,t\_40)
* 文字水印(e.g. [image/watermark,text\_Q2hpbmF0ZWxlY29t,type\_heiti,color\_FF0000,size\_40,g\_se](http://192.168.16.130:7480/public/hudie.jpg?x-zos-process=image/watermark,text_Q2hpbmF0ZWxlY29t,type_heiti,color_FF0000,size_40,g_se),t\_80)

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数用途** | **取值** | **是否必须** |
| t | 图片水印或文字水印的透明度 | [0, 100] | 否 |
| x | 文字水印距离图片边界的水平距离 | [0, 4096]  默认值：10 | 否 |
| y | 文字水印距离图片边界的垂直距离 | [0, 4096]  默认值：10 | 否 |
| text | 指定文字水印内容 | Base64编码后的字符串，编码结果字符串中‘/’要替换为‘\_’ | 否 |
| color | 指定文字水印的颜色 | RGB颜色值。  默认：FFFFFF（白色） | 否 |
| size | 指定文字水印的字体大小 | 默认值：40 | 否 |
| type | 指定文字水印的字体类型 | 如Airal, Helvetica, 支持中文字体包括yahei(微软雅黑)，heiti(黑体)，kaishu（楷书）,youyuan(幼圆) | 否 |
| rotate | 指定文字水印顺时针旋转角度 | [0, 360]  默认值：0 | 否 |
| image | 指定图片水印名称，水印图片需要和原图存放在相同存储空间 | 水印图片可以进行预处理（e.g. 水印图片缩放为30%并旋转180度，hudie.jpg?x-zos-process=image/resize,p\_30/rotate,180 ），需要转换成base64编码，编码结果字符串中‘/’要替换为‘\_’ | 否 |
| g | 指定水印在图片中的位置 | nw：左上(默认)  north：中上  ne：右上  west：左中  center：中部  east：右中  sw：左下  south：中下  se：右下 | 否 |

* 格式转化(e.g. image/format,png)

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数用途** | **取值** | **是否必须** |
| [value] | 将原图转换成指定格式 | Jpg、png、webp、bmp、tiff | 是 |

* 获取图片信息(e.g. image/info)

返回结果说明

{

ETag: \*string

VersionId: \*string

}

|  |  |  |
| --- | --- | --- |
| **参数名称** | **参数描述** | **类型** |
| VersionId | 对象的版本号 | \*string |
| ETag | 对象的ETag | \*string |

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/awserr"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

input := &s3.ProcessObjectInput{

Bucket: aws.\*string("verbucks"),

Key: aws.\*string("hudie2\_mini.jpg"),

ProcessSource: aws.\*string("src/hudie.jpg"),

ZosProcess: aws.\*string("image/resize,w\_200"),

}

result, err := svc.ProcessObject(input)

if err != nil {

if aerr, ok := err.(awserr.Error); ok {

switch aerr.Code() {

default:

fmt.Println(aerr.Error())

}

} else {

fmt.Println(err.Error())

}

}

fmt.Println(result)

}

4.2、Get请求获取图片

功能说明

图片处理是在get\_object基础进行的扩展，用于对存储桶中的图片文件在线处理，支持处理图片格式JPG、PNG、WEBP、BMP、TIFF。

方法原型

func (c \*S3) GetObject(input \*GetObjectInput) (\*GetObjectOutput,error)

参数说明

* input:类型\*s3.GetObjectInput，图片处理的请求接口参数，定义的方法如下:

type GetObjectInput struct {

Bucket \*string `location:"uri" locationName:"Bucket" type:"\*string" required:"true"`

Key \*string `location:"uri" locationName:"Key" min:"1" type:"\*string" required:"true"`

ZosProcess \*string `location:"query\*string" locationName:"x-zos-process" type:"\*string"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Bucket | 存储桶的名称 | \*string | 是 |
| Key | 存储对象的名称 | \*string | 是 |
| ZosProces | 图片处理的方式，包含缩放，裁剪，旋转，水印，格式转换，获取信息功能，定义方法同5.1参数说明 | \*string | 是 |

返回结果说明

{

AcceptRanges: \*string

Body: io.ReadCloser

ContentLength: \*int64

ContentRange: \*string

ContentType: \*string

ETag: \*string

LastModified: \*time.Time

Metadata: map[\*string]\*string

StorageClass: \*string

VersionId: \*string

}

|  |  |  |
| --- | --- | --- |
| **参数名称** | **参数描述** | **类型** |
| AcceptRanges | 标识定义范围的单位 | \*string |
| Body | 返回对象的内容 | io.ReadCloser |
| ContentLength | 响应体的长度 | \*int64 |
| ContentRange | 指明对象在返回的响应中包含的部分 | \*string |
| ContentType | 描述对象的格式 | \*string |
| LastModified | 对象的创建时间 | \*time.Time |
| ETag | 对象的ETag | \*strng |
| Metadata | 和对象一起存储的元数据信息 | map[\*string]\*string |
| VersionId | 对象的版本信息 | \*string |
| StorageClass | 对象的存储级别 | \*string |

示例

package main

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/awserr"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/service/s3"

)

func main() {

ak := "9L6CF1NST0D4ATG7HFS4"

sk := "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"

end\_point := "http://192.168.218.130:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(ak, sk, ""),

Endpoint: aws.\*string(end\_point),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

S3ForcePathStyle: aws.Bool(true),

})

svc := s3.New(sess)

input := &s3.GetObjectInput{

Bucket: aws.\*string("verbucks"),

Key: aws.\*string("hudie2\_mini.jpg"),

ZosProcess: aws.\*string("image/resize,w\_200"),

}

result, err := svc.GetObject(input)

if err != nil {

if aerr, ok := err.(awserr.Error); ok {

switch aerr.Code() {

default:

fmt.Println(aerr.Error())

}

} else {

fmt.Println(err.Error())

}

}

fmt.Println(result)

}

5、Topic操作

5.1、Create Topic

功能说明

创建可以向其发布通知的主题。此操作是幂等的，因此如果请求者已经拥有一个该指定名称的主题，则返回该主题的ARN，而不创建新主题。

方法原型

func (c \*SNS) CreateTopic(input \*CreateTopicInput) (\*CreateTopicOutput, error)

参数说明

* input:类型是个struct，具体定义属性如下:

type CreateTopicInput struct {

Attributes map[\*string]\*string `type:"map"`

Name \*string `type:"\*string" required:"true"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| Name | Topic的名称 | \*string | 是 |
| Attributes | Topic的属性，根据不同的通知后端（Kafka、RabbitMQ、HTTP/S），具有不同的键值 | map | 是 |

返回结果说明

* CreateTopicOutPut:类型是个struct，具体定义属性如下:

type CreateTopicOutput struct {

TopicArn \*string `type:"\*string"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **参数描述** | **类型** |
| TopicArn | Topic的Arn标识 | \*string |

示例

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/service/sns"

)

func main() {

accessKey := "test"

secretKey := "test"

endPoint := "192.168.198.110:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(accessKey, secretKey, ""),

Endpoint: aws.\*string(endPoint),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

})

svc := sns.New(sess)

attributes := make(map[\*string]\*string)

attributes["push-endpoint"] = "kafka://11.50.132.160:9092"

attributes["kafka-ack-level"] = "broker"

resp, err := svc.CreateTopic(&sns.CreateTopicInput{

Name: aws.\*string("Kafka\_Topic"),

Attributes: aws.\*stringMap(attributes),

})

if err != nil {

fmt.Println("failed", err)

} else {

fmt.Println("success", resp)

}

}

5.2、Delete Topic

功能说明

删除主题。此操作是幂等（Idempotence）操作，因此删除不存在的主题不会导致错误。

方法原型

func (c \*SNS) DeleteTopic(input \*DeleteTopicInput) (\*DeleteTopicOutput, error)

参数说明

* input:类型是个struct，具体定义属性如下:

type DeleteTopicInput struct {

TopicArn \*string `type:"\*string" required:"true"`

}

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数描述** | **类型** | **是否必须** |
| TopicArn | Topic的Arn标识 | \*string | 是 |

返回结果说明

返回结果无特殊意义字段，无需说明。

示例

import (

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/service/sns"

)

func main() {

accessKey := "test"

secretKey := "test"

endPoint := "192.168.198.110:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(accessKey, secretKey, ""),

Endpoint: aws.\*string(endPoint),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

})

svc := sns.New(sess)

resp, err := svc.DeleteTopic(&sns.DeleteTopicInput{

TopicArn: aws.\*string("arn:aws:sns:default::Kafka\_Topic"),

})

if err != nil {

fmt.Println("failed", err)

} else {

fmt.Println("success", resp)

}

}

5.3、List Topics

功能说明

返回所有主题的所有属性。当不含主题时，返回空Topics数组元素。

方法原型

func (c \*SNS) ListTopics(input \*ListTopicsInput) (\*ListTopicsOutput, error)

参数说明

* input:类型是个struct，具体定义属性如下:

type ListTopicsInput struct {}

返回结果说明

* ListTopicsOutPut:类型是个struct，具体定义属性如下:

type ListTopicsOutput struct {

// A list of topic ARNs.

Topics []\*Topic `type:"list"`

}

Topic:类型是个struct，具体定义属性如下:

type Topic struct {

// The topic's ARN.

TopicArn \*string `type:"\*string"`

}

|  |  |  |
| --- | --- | --- |
| **返回结果** | **参数描述** | **类型** |
| Topics | Topic的列表 | []\*Topic |
| TopicArn | Topic的Arn标识 | \*string |

示例

import (

"crypto/tls"

"fmt"

"github.com/aws/aws-sdk-go/aws"

"github.com/aws/aws-sdk-go/aws/credentials"

"github.com/aws/aws-sdk-go/aws/session"

"github.com/aws/aws-sdk-go/service/sns"

"net/http"

)

func main() {

accessKey := "test"

secretKey := "test"

endPoint := "192.168.198.110:7480"

sess, \_ := session.NewSession(&aws.Config{

Credentials: credentials.NewStaticCredentials(accessKey, secretKey, ""),

Endpoint: aws.\*string(endPoint),

Region: aws.\*string("default"),

DisableSSL: aws.Bool(true),

})

svc := sns.New(sess)

resp, err := svc.ListTopics(nil)

if err != nil {

fmt.Println("failed", err)

} else {

fmt.Println("success")

for \_, t := range resp.Topics {

fmt.Println(\*t.TopicArn)

}

}

}