# 下载与安装

## 相关资源

* SDK压缩包快速下载地址: 见帮助中心

## 环境依赖

* cmake (3.2~3.18)
* GCC 4.9 或更新版本
* 以下库及其相关头文件，可在对应linux发行版的包管理器中安装，括号中给出的是经过验证的版本。

libcurl 及 libcurl-devel (7.29.0)

openssl 及 openssl-devel (1.0.2k)

libuuid 及 libuuid-devel (2.23.2)

zlib 及 zlib-devel (1.2.7)

tips:

对于 CentOS 1810 minimal安装，可用如下命令搭建依赖环境

yum install epel-release

yum install centos-release-scl

yum install devtoolset-9-gcc-c++

yum install cmake3

yum install libcurl-devel openssl-devel libuuid-devel

启用 devtoolset-9

source /opt/rh/devtoolset-9/enable

## SDK安装使用

* 从源码编译安装

从SDK压缩包下载链接下载源码包，并按照环境依赖准备好环境，然后执行以下命令安装。

tar -xvf sdk-cpp.tar.gz

mkdir build

cd build

cmake3 ../sdk-cpp

make -j `nproc`

make install

* 使用SDK

例如编译名为Create.cpp的源文件，使用如下命令。

g++ -std=c++11 -laws-cpp-sdk-core -laws-cpp-sdk-s3 Create.cpp -o Create

## 连接

Aws::Client::ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

Aws::Auth::AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false)

## 全局错误码及类定义

S3Error类方法定义如下:

//获取本次http请求的返回状态码，HttpResponseCode的定义参考HttpResponseCode错误码定义

Aws::Http::HttpResponseCode GetResponseCode()

//获取本次请求错误类型，S3Errors的定义参考S3Errors错误码定义

const Aws::S3::S3Errors& GetError()

//获取本次请求错误码详细信息

const Aws::String& GetMessage()

HttpResponseCode枚举定义如下:

请求可能会返回相关Http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 |

S3Errors枚举定义如下:

enum class S3Errors

{

//From Core//

//////////////////////////////////////////////////////////////////////////////////////////

INCOMPLETE\_SIGNATURE = 0,

INTERNAL\_FAILURE = 1,

INVALID\_ACTION = 2,

INVALID\_CLIENT\_TOKEN\_ID = 3,

INVALID\_PARAMETER\_COMBINATION = 4,

INVALID\_QUERY\_PARAMETER = 5,

INVALID\_PARAMETER\_VALUE = 6,

MISSING\_ACTION = 7, // SDK should never allow

MISSING\_AUTHENTICATION\_TOKEN = 8, // SDK should never allow

MISSING\_PARAMETER = 9, // SDK should never allow

OPT\_IN\_REQUIRED = 10,

REQUEST\_EXPIRED = 11,

SERVICE\_UNAVAILABLE = 12,

THROTTLING = 13,

VALIDATION = 14,

ACCESS\_DENIED = 15,

RESOURCE\_NOT\_FOUND = 16,

UNRECOGNIZED\_CLIENT = 17,

MALFORMED\_QUERY\_STRING = 18,

SLOW\_DOWN = 19,

REQUEST\_TIME\_TOO\_SKEWED = 20,

INVALID\_SIGNATURE = 21,

SIGNATURE\_DOES\_NOT\_MATCH = 22,

INVALID\_ACCESS\_KEY\_ID = 23,

REQUEST\_TIMEOUT = 24,

NETWORK\_CONNECTION = 99,

UNKNOWN = 100,

BUCKET\_ALREADY\_EXISTS= static\_cast<int>(Aws::Client::CoreErrors::SERVICE\_EXTENSION\_START\_RANGE) + 1,

BUCKET\_ALREADY\_OWNED\_BY\_YOU,

INVALID\_OBJECT\_STATE,

NO\_SUCH\_BUCKET,

NO\_SUCH\_KEY,

NO\_SUCH\_UPLOAD,

OBJECT\_ALREADY\_IN\_ACTIVE\_TIER,

OBJECT\_NOT\_IN\_ACTIVE\_TIER

};

# 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

Aws::S3::Model::CreateBucketOutcome Aws::S3::S3Client::CreateBucket(const Aws::S3::Model::CreateBucketRequest& request) const

#### 参数说明

* request:类型Aws::S3::Model::CreateBucketRequest，创建Bucket请求接口参数，定义的方法如下:

//设置要创建的Bucket名称，必须设置，Aws::String可通过标准std::string赋值

void SetBucket(const Aws::String& value)

//设置Bucket的ACL规则,BucketCannedACL是个enum class类型，从小到大分别是NOT\_SET,private\_，public\_read，public\_read\_write和authenticated\_read，分别对应数字0到4

void SetACL(const Aws::S3::Model::BucketCannedACL& value)

//是否开启Bucket的对象锁定功能

void SetObjectLockEnabledForBucket(bool value)

// Bucket 权限设置，详见[1.9](#_1.9、Put Bucket ACL)

void SetGrantFullControl(Aws::String&& value)

void SetGrantRead(const Aws::String& value)

void SetGrantWrite(const Aws::String& value)

// Bucket ACL 读写权限设置

void SetGrantReadACP(const Aws::String& value)

void SetGrantWriteACP(const Aws::String& value)

#### 返回结果及说明

* CreateBucketOutCome:类型Aws::S3::Model::CreateBucketOutCome，创建Bucket请求接口返回参数，定义的方法如下:

//本次请求是否成功

void IsSuccess()

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/CreateBucketRequest.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void create\_bucket(S3Client &client, const Aws::String &bucket\_name) {

// 设置请求参数

CreateBucketRequest request;

request.SetBucket(bucket\_name);

// 发出请求

auto outcome = client.CreateBucket(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: CreateBucket: " << "Http code: "<< (int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " <<

err.GetMessage() << std::endl;

} else {

std::cout << "successful created bucket: " << bucket\_name << "\n";

}

return;

}

int main(int argc, char\* argv[])

{

if(argc != 2)

{

std::cout << "pls input bucket\_name" << std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

create\_bucket(client, bucket\_name);

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model::DeleteBucketOutcome Aws::S3::S3Client::DeleteBucket(const Aws::S3::Model::DeleteBucketRequest &request) const

#### 参数说明

* request:类型Aws::S3::Model::DeleteBucketRequest，删除Bucket请求接口参数，定义的方法如下:

//设置要删除的Bucket名称

void SetBucket(const Aws::String& value)

#### 返回结果说明

* DeleteBucketOutcome:类型Aws::S3::Model::DeleteBucketOutcome，删除Bucket请求接口返回参数，定义的方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/DeleteBucketRequest.h>

#include <aws/core/Aws.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Client;

using namespace std;

void DeleteBucket(S3Client &client, const Aws::String &bucket\_name)

{

// 设置请求

Aws::S3::Model::DeleteBucketRequest request;

request.SetBucket(bucket\_name);

auto outcome = client.DeleteBucket(request);

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

cout << "ERROR: DeleteBucket: " << endl

<< "Http code: " << (int)err.GetResponseCode() << endl

<< "Error Type:" << (int)err.GetErrorType() << endl

<< "Error Msg: " << err.GetMessage() << endl

<< "Exception Name: " << err.GetExceptionName() << endl;

}

else

{

std::cout << "successfully delete bucket: " << bucket\_name << "\n";

}

}

int main(int argc, char \*argv[])

{

if (argc != 2)

{

std::cout << "pls input bucket\_name" << std::endl;

return -1;

}

const Aws::String bucket\_name = argv[1];

Aws::SDKOptions options;

Aws::InitAPI(options);

{

// 设置连接参数

Aws::Client::ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; //S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

Aws::Auth::AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

DeleteBucket(client, bucket\_name);

}

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model::HeadBucketOutcome HeadBucket(const Aws::S3::Model::HeadBucketRequest&request) const

#### 参数说明

* request:类型Aws::S3::Model::HeadBucketRequest，HeadBucket请求参数，定义方法如下:

//设置Bucket名称，必须设置，Aws::StringAws::String可通过标准std::string赋值

void SetBucket(const Aws::String& value)

#### 返回结果说明

* 类型Aws::S3::Model::HeadBucketOutcome:HeadBucket请求返回参数，定义方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/HeadBucketRequest.h>

#include <aws/core/Aws.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void head\_bucket(S3Client &client, const Aws::String &bucket\_name) {

// 设置请求参数

HeadBucketRequest request;

request.SetBucket(bucket\_name);

// 发出请求

auto outcome = client.HeadBucket(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: HeadBucket: " << "Http code: "<< (int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " <<

err.GetMessage() << "Exception name:" << err.GetExceptionName() << std::endl;

} else {

std::cout << "successful created bucket: " << bucket\_name << "\n";

}

return;

}

int main(int argc, char\* argv[])

{

if(argc != 2)

{

std::cout << "pls input bucket\_name" << std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

head\_bucket(client, bucket\_name);

Aws::ShutdownAPI(options);

}

## 4、List Bucket

#### 功能说明

List Bucket 可以列出该用户名下所有的bucket列表。

#### 方法原型

Aws::S3::Model::ListBucketsOutcome Aws::S3::S3Client::ListBuckets() const

#### 参数说明

无

#### 返回结果说明

* ListBucketsOutcome:类型Aws::S3::Model::ListBucketsOutcome，列出Bucket请求接口返回参数，定义的方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

//获取本次请求返回的结果数据

Aws::S3::Model::ListBucketsResult &GetResult()

Aws::S3::Model::ListBucketsResult 的定义如下:

// 获取Bucket列表

const Aws::Vector<Aws::S3::Model::Bucket> &Aws::S3::Model::ListBucketsResult::GetBuckets() const

// 返回的Bucket的Owner

const Aws::S3::Model::Owner &Aws::S3::Model::ListBucketsResult::GetOwner() const

Aws::S3::Model::Bucket 定义方法如下:

// 获取Bucket名称

const Aws::String& GetName() const

// 获取Bucket 的创建时间（当修改Bucket属性时该日期会随之改变）

const Aws::Utils::DateTime& GetCreationDate() const

Aws::S3::Model::Owner 定义方法如下：

// 获取Owner 的名称

const Aws::String& GetDisplayName() const

// 获取Owner 的 ID

const Aws::String& GetID() const

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/ListBucketsResult.h>

#include <aws/core/Aws.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Client;

using namespace std;

void ListBuckets(S3Client &client)

{

// 设置请求

auto outcome = client.ListBuckets();

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

cout << "ERROR: ListBucket: " << endl

<< "Http code: " << (int)err.GetResponseCode() << endl

<< "Error Type:" << (int)err.GetErrorType() << endl

<< "Error Msg: " << err.GetMessage() << endl

<< "Exception Name: " << err.GetExceptionName() << endl;

}

else

{

auto buckets = outcome.GetResult().GetBuckets();

auto owner = outcome.GetResult().GetOwner();

cout << "Owner:" << endl

<< "\t"

<< "display\_name: " << owner.GetDisplayName() << "\tID: " <<

owner.GetID() << endl;

cout << "Buckets:" << endl;

for (auto iter = buckets.begin(); iter != buckets.end(); ++iter)

{

cout << "\t" << iter->GetName() << "\t" << iter->GetCreationDate().ToLocalTimeString(Aws::Utils::DateFormat::ISO\_8601) << endl;

}

}

}

int main(int argc, char \*argv[])

{

Aws::SDKOptions options;

Aws::InitAPI(options);

{

// 设置连接参数

Aws::Client::ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; //S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

Aws::Auth::AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

ListBuckets(client);

}

Aws::ShutdownAPI(options);

}

## 1.5、List Objects

#### 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

Aws::S3::Model::ListObjectsOutcome Aws::S3::S3Client::ListObjects(const Aws::S3::Model::ListObjectsRequest &request) const

#### 参数说明

* request:类型Aws::S3::Model::ListObjectsResult，定义方法如下:

//设置要列出的Bucket名称

void SetBucket(const Aws::String& value)

//设置分组结果时使用的字符

void SetDelimiter(const Aws::String& value)

//设置返回的key的编码方式，合法值为 EncodingType::url

void SetEncodingType(const EncodingType& value)

//设置列出key时的起始key

void SetMarker(const Aws::String& value)

// 设置一次返回的key的数量

void SetMaxKeys(int value)

// 设置列出指定前缀的key

void SetPrefix(const Aws::String& value)

#### 返回结果说明

* 类型Aws::S3::Model::ListObjectsOutcome，定义方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

// 获取返回结果

Aws::S3::Model::ListObjectsResult& GetResult()

类型ListObjectsResult，定义方法如下:

// 获取返回的每个Object的元数据

const Aws::Vector<Object>& GetContents()

// 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； 可以使用 const Aws::String& CommonPrefix::GetPrefix()获取Prefix

const Aws::Vector<CommonPrefix>& GetCommonPrefixes()

// 获取请求时指定的delimiter

const Aws::String& GetDelimiter()

// 获取编码Object Key的方式

const EncodingType& GetEncodingType()

// 获取是否返回了所有满足要求的Key

bool GetIsTruncated()

// 若请求时设置了Marker,在返回结果中将包含该Marker

const Aws::String& GetMarker()

// 获取指定的MaxKeys

int GetMaxKeys()

// 获取BucketName

const Aws::String& GetName()

// 若指定了delimiter且仅返回了部分结果，则可通过设置后续请求的Marker为NextMarker来获取剩余的结果，若没有返回NextMarker且只返回了部分结果，则可使用返回的最后一个Key作为后续请求的Marker来获取剩余结果

const Aws::String& GetNextMarker

// 获取请求时指定的Prefix

const Aws::String& GetPrefix()

类型Aws::S3::Model::Object，定义方法如下:

//获取Entity Tag

const Aws::String& GetETag()

// 获取 Key

const Aws::String& GetKey()

// 获取修改时间

const Aws::Utils::DateTime& GetLastModified()

// 获取所有者

const Owner& GetOwner()

// 获取Object Size

long long GetSize()

// 获取Object存储级别

const ObjectStorageClass& GetStorageClass()

#### 示例

include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/ListObjectsRequest.h>

#include <aws/core/Aws.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Client;

using namespace std;

void ListObjects(S3Client &client, const Aws::String &bucket)

{

// 设置请求

ListObjectsRequest request;

request.WithBucket(bucket)

.WithEncodingType(EncodingType::url)

.WithMaxKeys(2);

auto outcome = client.ListObjects(request);

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

cout << "ERROR: ListObjects: " << endl

<< "Http code: " << (int)err.GetResponseCode() << endl

<< "Error Type:" << (int)err.GetErrorType() << endl

<< "Error Msg: " << err.GetMessage() << endl

<< "Exception Name: " << err.GetExceptionName() << endl;

}

else

{

auto result = outcome.GetResult();

auto contents = result.GetContents();

cout << "Bucket:" << result.GetName() << endl;

for (auto content = contents.begin(); content != contents.end();

++content)

{

cout << "\tKey: " << content->GetKey() << "\tSize: " << content->GetSize() << "\tOwner: " << content->GetOwner().GetDisplayName()

<< "\tStorageClass: " << ObjectStorageClassMapper::GetNameForObjectStorageClass(content->GetStorageClass())

<< "\tETag: " << content->GetETag() << " \tLastModified:

" << content->GetLastModified().ToLocalTimeString(Aws::Utils::DateFormat::ISO\_8601) << endl;

}

cout << "nextMarker:" << result.GetNextMarker() << endl;

}

}

int main(int argc, char \*argv[])

{

if (argc != 2)

{

cout << "pls input bucket\_name" << endl;

return -1;

}

const Aws::String bucket = argv[1];

Aws::SDKOptions options;

Aws::InitAPI(options);

{

// 设置连接参数

Aws::Client::ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

Aws::Auth::AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

ListObjects(client, bucket);

}

Aws::ShutdownAPI(options);

}

## 1.6、Put Bucket Policy

#### 功能说明

Put Bucket Policy 请求用于为某个Bucket设置桶策略。

#### 方法原型

Aws::S3::Model::PutBucketPolicyOutcome S3Client::PutBucketPolicy(

const Aws::S3::Model::PutBucketPolicyRequest& request) const

#### 参数说明

* request: Aws::S3::Model::PutBucketPolicyRequest类型，该类定义的方法有:

// 指定为哪一个bucket设置policy，必须要设置

void SetBucket(const Aws::String& value)

// 设置用于请求的请求体，请求体为具体的policy规则

void SetBody(const std::shared\_ptr<Aws::IOStream>& body)

request body中的bucket policy在请求的请求体中，bucket policy格式为json，其中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 | vector | 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" | map | 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 | vector | 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。 | map | 否 |

bucket policy的json示例如下：

{

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

"Statement":[

{

"Sid": "id-1",

"Effect": "Allow",

"Principal": {"\*"},

"Action": [ "s3:PutObject","s3:PutObjectAcl"],

"Resource": ["arn:aws:s3:::bucketName/\*"]

}

]

}

#### 返回结果说明

* 返回结果类型为Aws::S3::Model::PutBucketPolicyOutCome，该类型定义的方法有:

// 本次请求是否成功

void IsSuccess(const Aws::String& value)

// 获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

#include <aws/s3/model/PutBucketPolicyRequest.h>

#include <aws/core/http/Scheme.h>

int main(int argc, char\* argv[])

{

// 设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

Aws::Client::ClientConfiguration cfg;

cfg.endpointOverride = "http://192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

Aws::Auth::AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

// 创建S3Client

Aws::S3::S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

const Aws::String bucket\_name = "bucket1";

Aws::S3::Model::PutBucketPolicyRequest request;

request.SetBucket(bucket\_name);

std::shared\_ptr<Aws::IOStream> policy = std::make\_shared<std::stringstream>("{\"Version\":\"2012-10-17\",\"Id\":\"S3PolicyId1\",\"Statement\":[{\"Sid\":\"deny\_get\",\"Effect\":\"Deny\",\"Principal\":\"\*\",\"Action\":\"s3:GetObject\",\"Resource\":\"arn:aws:s3:::bucket1/\*\"}]}");

request.SetBody(policy);

auto outcome = client.PutBucketPolicy(request);

if (outcome.IsSuccess())

{

std::cout << "Bucket policy set successfully." << std::endl;

} else

{

auto err = outcome.GetError();

std::cout << "PutBucketPolicy ERROR" << ", Http code: "<< (int)err.GetResponseCode() <<

", Error Type:" << (int)err.GetErrorType() << ", Error Msg: "

<< err.GetMessage() << std::endl;

}

Aws::ShutdownAPI(options);

return 0;

}

## 1.7、Get Bucket Policy

#### 功能说明

Get Bucket Policy 请求用于获取某个Bucket的桶策略。

#### 方法原型

Aws::S3::Model::GetBucketPolicyOutcome S3Client::GetBucketPolicy(

const Aws::S3::Model::GetBucketPolicyRequest& request) const

#### 参数说明

* request: Aws::S3::Model::GetBucketPolicyRequest类型，该类定义的方法有:

// 指定获取哪一个bucket的policy，必须要设置

void SetBucket(const Aws::String& value)

#### 返回结果说明

* 返回结果类型为Aws::S3::Model::GetBucketPolicyOutCome，该类型定义的方法有:

// 本次请求是否成功

void IsSuccess(const Aws::String& value)

// 获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

// 获取bucket policy result接口

const GetBucketPolicyResult& GetResult() const

GetBucketPolicyResult类封装了GetPolicy接口，用于获取bucket policy：

Aws::IOStream& GetPolicy()

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

#include <aws/s3/model/GetBucketPolicyRequest.h>

#include <aws/core/http/Scheme.h>

int main(int argc, char\* argv[])

{

// 设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

Aws::Client::ClientConfiguration cfg;

cfg.endpointOverride = "http://192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

Aws::Auth::AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

// 创建S3Client

Aws::S3::S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

const Aws::String bucket\_name = "bucket1";

Aws::S3::Model::GetBucketPolicyRequest request;

request.SetBucket(bucket\_name);

auto outcome = client.GetBucketPolicy(request);

if (outcome.IsSuccess())

{

Aws::StringStream policy\_stream;

Aws::String line;

outcome.GetResult().GetPolicy() >> line;

policy\_stream << line;

std::cout << "Bucket Policy: " << policy\_stream.str()

<< std::endl;

} else

{

auto err = outcome.GetError();

std::cout << "PutBucketPolicy ERROR" << ", Http code: "<< (int)err.GetResponseCode() <<

", Error Type:" << (int)err.GetErrorType() << ", Error Msg: "

<< err.GetMessage() << std::endl;

}

Aws::ShutdownAPI(options);

return 0;

}

## 1.8、Delete Bucket Policy

#### 功能说明

Delete Bucket Policy 请求用于删除某个Bucket的桶策略。

#### 方法原型

Aws::S3::Model::DeleteBucketPolicyOutcome S3Client::DeleteBucketPolicy(const Aws::S3::Model::DeleteBucketPolicyRequest& request) const

#### 参数说明

* request: Aws::S3::Model::DeleteBucketPolicyRequest类型，该类定义的方法有:

// 指定删除哪一个bucket的policy，必须要设置

void SetBucket(const Aws::String& value)

#### 返回结果说明

* 返回结果类型为Aws::S3::Model::DeleteBucketPolicyResponse，该类型定义的方法有:

// 本次请求是否成功

void IsSuccess(const Aws::String& value)

// 获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

#include <aws/s3/model/DeleteBucketPolicyRequest.h>

#include <aws/core/http/Scheme.h>

int main(int argc, char\* argv[])

{

// 设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

Aws::Client::ClientConfiguration cfg;

cfg.endpointOverride = "http://192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

Aws::Auth::AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

// 创建S3Client

Aws::S3::S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

const Aws::String bucket\_name = "bucket1";

Aws::S3::Model::DeleteBucketPolicyRequest request;

request.SetBucket(bucket\_name);

auto outcome = client.DeleteBucketPolicy(request);

if (outcome.IsSuccess())

{

std::cout << "Bucket policy delete successfully." << std::endl;

} else

{

auto err = outcome.GetError();

std::cout << "PutBucketPolicy ERROR" << ", Http code: "<< (int)err.GetResponseCode() <<

", Error Type:" << (int)err.GetErrorType() << ", Error Msg: "

<< err.GetMessage() << std::endl;

}

Aws::ShutdownAPI(options);

return 0;

}

## 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

Aws::S3::Model::PutBucketAclOutcome S3Client::PutBucketAcl(const PutBucketAclRequest& request) const

#### 参数说明

* request: Aws::S3::Model::PutBucketAclRequest类型，该类定义的方法有:

//指定为哪一个bucket设置ACL，必须要设置

void SetBucket(const Aws::String& value)

// 以下三组参数，对应三种方式，不能同时使用

// 设置ACL，BucketCannedACL取值范围为

// private\_,

// public\_read,

// public\_read\_write,

// authenticated\_read

void SetACL(const BucketCannedACL& value)

// 设置AccessControlPolicy

void SetAccessControlPolicy(const AccessControlPolicy& value)

// 设置Grant\*形式的参数,字符串格式为key=value，比如

// "id=xxxx, emailAdddress=xxxx, uri=xxxx"

// 可以组合多个key=value

// uri取值为http://acs.amazonaws.com/groups/global/AllUsers

// 或者http://acs.amazonaws.com/groups/global/AuthenticatedUsers

void SetGrantFullControl(const Aws::String& value)

void SetGrantRead(const Aws::String& value)

void SetGrantReadACP(const Aws::String& value)

void SetGrantWrite(const Aws::String& value)

void SetGrantWriteACP(const Aws::String& value)

AccessControlPolicy类型，该类定义的方法有:

//指定桶所有者

void SetOwner(const Owner& value)

// 设置授权列表

void SetGrants(const Aws::Vector<Grant>& value)

Owner类型，该类定义的方法有:

//指定桶所有者用户ID

void SetID(const Aws::String& value)

Grant类型，该类定义的方法有:

//指定被授权用户

void SetGrantee(const Grantee& value)

// 指定被授权权限，取值范围为

// FULL\_CONTROL,

// WRITE,

// WRITE\_ACP,

// READ,

// READ\_ACP

void SetPermission(const Permission& value)

Grantee类型，该类定义的方法有:

//指定被授权用户类型，取值范围

// CanonicalUser,

// AmazonCustomerByEmail,

// Group

void SetType(const Type& value)

// 指定被授权用户ID，如果用户类型为CanonicalUser，需要指定该字段

void SetID(const Aws::String& value)

// 指定被授权用户邮箱，如果用户类型为AmazonCustomerByEmail，需要指定该字段

void SetEmailAddress(Aws::String&& value)

// 指定被授权组，如果用户类型为Group，需要指定该字段

// 取值为http://acs.amazonaws.com/groups/global/AllUsers

// 或者http://acs.amazonaws.com/groups/global/AuthenticatedUsers

void SetURI(const Aws::String& value)

#### 返回结果说明

* 返回结果类型为Aws::S3::Model::PutBucketAclOutCome，该类型定义的方法有:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/PutBucketAclRequest.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::S3::Model::TypeMapper;

using namespace Aws::S3::Model::PermissionMapper;

using namespace Aws::Auth;

using namespace std;

void put\_bucket\_acl(S3Client &client, const Aws::String &bucket\_name)

{

PutBucketAclRequest request;

request.SetBucket(bucket\_name);

// request.SetACL(BucketCannedACL::public\_read);

/\*

Owner owner;

owner.SetID("test-1");

Grantee grantee;

grantee.SetType(Type::CanonicalUser);

grantee.SetID("test-3");

// grantee.SetURI();

// grantee.SetEmailAddress();

Grant grant;

grant.SetGrantee(grantee);

grant.SetPermission(Permission::FULL\_CONTROL);

Aws::Vector<Grant> grants;

grants.push\_back(grant);

AccessControlPolicy policy;

policy.SetOwner(owner);

policy.SetGrants(grants);

request.SetAccessControlPolicy(policy);

\*/

request.SetGrantRead(

"id=test-2,uri=http://acs.amazonaws.com/groups/global/AllUsers");

auto outcome = client.PutBucketAcl(request);

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: " << "Http code: "<<

(int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " <<

err.GetMessage() << std::endl;

} else {

std::cout << "successful : " << endl;

}

return;

}

int main(int argc, char\* argv[])

{

if(argc != 2)

{

std::cout << "pls input bucket\_name" << std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

put\_bucket\_acl(client, bucket\_name);

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model::GetBucketAclOutcome S3Client::GetBucketAcl(const GetBucketAclRequest& request) const

#### 参数说明

* request: Aws::S3::Model::GetBucketAclRequest类型，该类定义的方法有:

//指定bucket

void SetBucket(const Aws::String& value)

#### 返回结果说明

* 返回结果类型为Aws::S3::Model::GetBucketAclOutCome，该类型定义的方法有:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

// 获取bucket acl result接口

const GetBucketAclResult& GetResult() const

GetBucketAclResult，该类型定义的方法有:

// 获取桶所有者

const Owner& GetOwner() const

// 获取授权列表

const Aws::Vector<Grant>& GetGrants() const

Owner，该类型定义的方法有:

// 获取桶所有者display name

const Aws::String& GetDisplayName() const

// 获取桶所有者用户ID

const Aws::String& GetID() const

Grant，该类型定义的方法有:

// 获取被授权用户

const Grantee& GetGrantee() const

// 获取被授权权限

const Permission& GetPermission()

Grantee，该类型定义的方法有:

// 获取被授权用户类型

const Type& GetType() const

// 获取被授权用户ID

const Aws::String& GetID() const

// 获取被授权用户display name

const Aws::String& GetDisplayName() const

// 获取被授权用户邮箱

const Aws::String& GetEmailAddress() const

// 获取被授权组uri

const Aws::String& GetURI() const

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/GetBucketAclRequest.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::S3::Model::TypeMapper;

using namespace Aws::S3::Model::PermissionMapper;

using namespace Aws::Auth;

using namespace std;

void get\_bucket\_acl(S3Client &client, const Aws::String &bucket\_name)

{

GetBucketAclRequest request;

request.SetBucket(bucket\_name);

auto outcome = client.GetBucketAcl(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: " << "Http code: "<<

(int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " <<

err.GetMessage() << std::endl;

} else {

std::cout << "successful : " << endl;

auto result = outcome.GetResult();

auto owner = result.GetOwner();

std::cout << "Owner ID = " << owner.GetID() << endl;

std::cout << "Owner DisplayName = " << owner.GetDisplayName() <<

endl;

auto grants = result.GetGrants();

for (auto it = grants.cbegin(); it != grants.cend(); ++it) {

auto grantee = it->GetGrantee();

auto type = grantee.GetType();

std::cout << "Type = " << GetNameForType(type) << endl;

if (type == Type::Group) {

auto uri = grantee.GetURI();

std::cout << "URI = " << uri << endl;

} else {

auto id = grantee.GetID();

auto display\_name = grantee.GetDisplayName();

auto email = grantee.GetEmailAddress();

std::cout << "ID = " << id << endl;

std::cout << "DisplayName = " << display\_name << endl;

std::cout << "Email = " << email << endl;

}

auto permission = it->GetPermission();

std::cout << "Permission = " << GetNameForPermission(permission) << endl;

}

}

return;

}

int main(int argc, char\* argv[])

{

if(argc != 2)

{

std::cout << "pls input bucket\_name" << std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

get\_bucket\_acl(client, bucket\_name);

Aws::ShutdownAPI(options);

}

## 1.11、Put Bucket Lifecycle Configuration

#### 功能说明

Put Bucket Lifecycle Configuration接口用来设置 Bucket 的生命周期规则。

#### 方法原型

Aws::S3::Model::PutBucketLifecycleConfigurationOutcome Aws::S3::S3Client::PutBucketLifecycleConfiguration(const Aws::S3::Model::PutBucketLifecycleConfigurationRequest& request) const

#### 参数说明

* request:类型Aws::S3::Model::PutBucketLifecycleConfigurationRequest，创建Bucket Lifecycle请求接口参数，定义的方法如下:

//设置要创建lifecycle规则的Bucket名称，Aws::String可通过标准std::string赋值，必须设置

void SetBucket(const Aws::String& value)

//设置Bucket的lifecycle规则容器, 必须设置，BucketLifecycleConfiguration是个//class类型

void SetLifecycleConfiguration(const BucketLifecycleConfiguration& value)

Aws::S3::Model::BucketLifecycleConfiguration的定义方法如下：

//设置组成BucketLifecycleConfiguration规则，LifecycleRule是一个class类型，//必须设置

void void SetRules(const Aws::Vector<LifecycleRule>& value)

//增加一条lifecycle规则

BucketLifecycleConfiguration& AddRules(const LifecycleRule& value)

Aws::S3::Model::LifecycleRule的定义方法如下：

//设置当前是否应用生命周期规则，ExpirationStatus类型是enum class,从小到大分别是NOT\_SET,Enabled,Disabled，必须设置

void SetStatus(const ExpirationStatus& value)

//设置生命周期规则的特征ID

void SetID(const Aws::String& value)

//设置对象的到期删除时间

void SetExpiration(const LifecycleExpiration& value)

//设置历史版本对象到期删除时间

void SetNoncurrentVersionExpiration(const NoncurrentVersionExpiration& value)

//设置生命周期规则的过滤条件void SetFilter(const LifecycleRuleFilter& value)

//设置对象到期转存规则

void SetTransitions(const Aws::Vector<Transition>& value)

//增加一条转存规则

LifecycleRule& AddTransitions(const Transition& value)

//设置历史版本对象到期转存规则

void SetNoncurrentVersionTransitions(const Aws::Vector<NoncurrentVersionTransition>& value)

//增加一条历史版本对象到期转存规则

LifecycleRule& AddNoncurrentVersionTransitions(const NoncurrentVersionTransition& value)

//设置一次分段上传最多持续时间

void SetAbortIncompleteMultipartUpload(const AbortIncompleteMultipartUpload& value)

Aws::S3::Model::LifecycleExpiration的定义方法如下：

//设置对象的到期删除时间，日期为ISO8601格式，必须为UTC午夜0时

void SetDate(const Aws::Utils::DateTime& value)

//设置对象受规则约束的天数，与SetDate只能设置一个

void SetDays(int value)

//设置是否删除“删除标记”

void SetExpiredObjectDeleteMarker(bool value)

Aws::S3::Model::LifecycleRuleFilter的定义方法如下：

//设置前缀过滤条件

void SetPrefix(const Aws::String& value)

//设置过滤标签

void SetTag(const Tag& value)

//设置多个tag条件

void SetAnd(const LifecycleRuleAndOperator& value)

Aws::S3::Model::Tag的定义方法如下：

//设置标签key

void SetKey(const Aws::String& value)

//设置标签的value

void SetValue(const Aws::String& value)

Aws::S3::Model::LifecycleRuleAndOperator的定义方法如下：

//设置过滤条件的前缀

void SetPrefix(const Aws::String& value)

//生命周期规则对拥有所有标签的对象才会生效

void SetTags(const Aws::Vector<Tag>& value)

Aws::S3::Model::Transition的定义方法如下：

//设置转存的存储级别,TransitionStorageClass是个enum class,有3个级别，//STANDARD(标准型)，INFREQUENT-ACCESS(低频型)，ARCHIVE(归档型，该级别暂未启用)

void SetStorageClass(const TransitionStorageClass& value)

//设置对象的转存时间，日期为ISO8601格式，必须为UTC午夜0时

void SetDate(const Aws::Utils::DateTime& value)

//设置对象受规则约束的天数，与SetDate只能设置一个

void SetDays(int value)

Aws::S3::Model::NoncurrentVersionTransition的定义方法如下：

//设置历史版本受规则约束的天数

void SetNoncurrentDays(int value)

//设置转存的存储级别,TransitionStorageClass是个enum class,有3个级别，//STANDARD(标准型)，INFREQUENT-ACCESS(低频型)，ARCHIVE(归档型，该级别暂未启用)

void SetStorageClass(const TransitionStorageClass& value)

Aws::S3::Model::NoncurrentVersionExpiration的定义方法如下：

//设置历史版本受规则约束的天数

void SetNoncurrentDays(int value)

Aws::S3::Model:: AbortIncompleteMultipartUpload的定义方法如下：

//设置分段上传最大持续天数

void SetDaysAfterInitiation(int value)

#### 返回结果及说明

* PutBucketLifecycleConfigurationOutcome:类型Aws::S3::Model::PutBucketLifecycleConfigurationOutcome，创建lifecycle请求接口返回参数，定义的方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/PutBucketLifecycleConfigurationRequest.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void put\_bucket\_lifecycle(S3Client &client, const Aws::String &bucket\_name) {

// 设置请求参数

PutBucketLifecycleConfigurationRequest request;

BucketLifecycleConfiguration lifecycle;

LifecycleRuleFilter filter;

LifecycleRule rule;

LifecycleExpiration expire;

filter.SetPrefix("");

expire.SetDays(365);

rule.SetID("first rule");

rule.SetStatus(ExpirationStatus::Enabled);

rule.SetFilter(filter);

rule.SetExpiration(expire);

lifecycle.AddRules(rule);

request.SetBucket(bucket\_name);

request.SetLifecycleConfiguration(lifecycle);

// 发出请求

auto outcome = client.PutBucketLifecycleConfiguration(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: Put Bucket Lifecycle: " << "Http code: "<<

(int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " <<

err.GetMessage() << std::endl;

} else {

std::cout << "successful put bucket lifecycle: " << bucket\_name <<

"\n";

}

return;

}

int main(int argc, char\* argv[])

{

if(argc != 2)

{

std::cout << "pls input bucket\_name" << std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:80"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

put\_bucket\_lifecycle(client, bucket\_name);

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model::GetBucketLifecycleConfigurationOutcome GetBucketLifecycleConfiguration(const Model::GetBucketLifecycleConfigurationRequest& request) const

#### 参数说明

* request:类型Aws::S3::Model::GetBucketLifecycleConfigurationRequest，获取Bucket Lifecycle请求接口参数，定义的方法如下:

//设置要获取lifecycle规则的Bucket名称，Aws::String可通过标准std::string赋值，必须设置

void SetBucket(const Aws::String& value)

#### 返回结果及说明

* GetBucketLifecycleConfigurationOutcome:类型Aws::S3::Model::GetBucketLifecycleConfigurationOutcome，获取lifecycle请求接口返回参数，定义的方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

//获取请求结果

const GetBucketLifecycleConfigurationResult& GetResult

//获取本次请求返回的数据

const Aws::Vector<LifecycleRule>& GetRules()

Aws::S3::Model::LifecycleRule定义的方法如下：

//获取对象的到期删除时间

const LifecycleExpiration& GetExpiration()

//获取生命周期规则的特征ID

const Aws::String& GetID()

//获取生命周期规则的过滤条件

const LifecycleRuleFilter& GetFilter()

//获取当前是否应用生命周期规则

const ExpirationStatus& GetStatus()

//获取对象到期转存规则

const Aws::Vector<Transition>& GetTransitions()

//获取历史版本对象到期转存规则

const Aws::Vector<NoncurrentVersionTransition>& GetNoncurrentVersionTransitions()

//获取历史版本对象到期删除时间

const NoncurrentVersionExpiration& GetNoncurrentVersionExpiration()

//获取一次分段上传最多持续时间

const AbortIncompleteMultipartUpload& GetAbortIncompleteMultipartUpload()

Aws::S3::Model::LifecycleExpiration定义的方法如下：

//获取对象的到期删除时间

const Aws::Utils::DateTime& GetDate()

//获取对象受规则约束的天数

int GetDays()

//获取是否删除 删除标记标识

bool GetExpiredObjectDeleteMarker()

Aws::S3::Model::LifecycleRuleFilter定义的方法如下：

//获取过滤条件的前缀

const Aws::String& GetPrefix()

//获取过滤标签

const Tag& GetTag()

//获取LifecycleRuleAndOperator

const LifecycleRuleAndOperator& GetAnd()

Aws::S3::Model::Tag定义的方法如下：

//获取标签key

const Aws::String& GetKey()

//获取标签的value

const Aws::String& GetValue()

Aws::S3::Model::LifecycleRuleAndOperator定义的方法如下：

//获取过滤条件的前缀

const Aws::String& GetPrefix()

//获取And条件内定义的多个标签

const Aws::Vector<Tag>& GetTags()

Aws::S3::Model::Transition定义的方法如下：

//设置转存的存储级别,TransitionStorageClass是个enum class,有3个级别，//STANDARD(标准型)，INFREQUENT-ACCESS(低频型)，ARCHIVE(归档型，该级别暂未启用)

const TransitionStorageClass& GetStorageClass()

//获取对象的转存时间

const Aws::Utils::DateTime& GetDate()

//获取对象受规则约束的天数

int GetDays()

Aws::S3::Model::NoncurrentVersionTransition定义的方法如下：

//获取历史版本受规则约束的天数

int GetNoncurrentDays()

//设置转存的存储级别,TransitionStorageClass有3个级别，STANDARD(标准型)，//INFREQUENT-ACCESS(低频型)，ARCHIVE(归档型，该级别暂未启用)

const TransitionStorageClass& GetStorageClass()

Aws::S3::Model::NoncurrentVersionExpiration定义的方法如下：

//获取历史版本受规则约束的天数

int GetNoncurrentDays()

Aws::S3::Model:: AbortIncompleteMultipartUpload定义的方法如下：

//获取分段上传最大持续天数

int GetDaysAfterInitiation()

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/GetBucketLifecycleConfigurationRequest.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void get\_bucket\_lifecycle(S3Client &client, const Aws::String &bucket\_name) {

// 设置请求参数

GetBucketLifecycleConfigurationRequest request;

request.SetBucket(bucket\_name);

// 发出请求

auto outcome = client.GetBucketLifecycleConfiguration(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: GetBucketLifecycleConfigurationOutcome: " <<

"Http code: "<< (int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " <<

err.GetMessage() << std::endl;

} else {

auto ruleResult = outcome.GetResult();

auto vecRules = ruleResult.GetRules();

for (auto rule : vecRules) {

auto days = rule.GetExpiration().GetDays();

ExpirationStatus status = rule.GetStatus();

string strStatus;

switch(status) {

case ExpirationStatus::NOT\_SET:break;

case ExpirationStatus::Enabled:strStatus = "Enabled";break;

case ExpirationStatus::Disabled:strStatus = "Disabled";

}

auto id = rule.GetID();

std::cout << "success GetBucketlifecycle!" << std::endl

<< "ID: " << id << std::endl

<< "Status: " << strStatus << std::endl

<< "days: " << days << std::endl;

}

}

return;

}

int main(int argc, char\* argv[])

{

if(argc != 2)

{

std::cout << "pls input bucket\_name" << std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:80"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

get\_bucket\_lifecycle(client, bucket\_name);

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model::DeleteBucketLifecycleOutcome DeleteBucket (const Model::DeleteBucketLifecycleRequest& request) const

#### 参数说明

* request:类型Aws::S3::Model::DeleteBucketLifecycleRequest，删除Bucket Lifecycle请求接口参数，定义的方法如下:

//设置要删除lifecycle的Bucket名称，Aws::String可通过标准std::string赋值，//必须设置

void SetBucket(const Aws::String& value)

#### 返回结果及说明

* DeleteBucketLifecycleOutcome:类型Aws::S3::Model::DeleteBucketLifecycleOutcome,删除lifecycle请求接口返回参数，定义的方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/DeleteBucketLifecycleRequest.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void del\_bucket\_lifecycle(S3Client &client, const Aws::String &bucket\_name) {

// 设置请求参数

DeleteBucketLifecycleRequest request;

request.SetBucket(bucket\_name);

// 发出请求

auto outcome = client.DeleteBucketLifecycle(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: DeleteLifecycleOutcome: " << "Http code: "<< (int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " << err.GetMessage() << std::endl;

} else {

std::cout << "successful delete bucket lifecycle: " << bucket\_name << std::endl;;

}

return;

}

int main(int argc, char\* argv[])

{

if(argc != 2)

{

std::cout << "pls input bucket\_name" << std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:80"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

del\_bucket\_lifecycle(client, bucket\_name);

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model::PutBucketWebsiteOutcome PutBucketWebsite(const Aws::S3::Model::PutBucketWebsiteRequest& request) const;

#### 参数说明

* request:方法PutBucketWebsite请求接口的参数，具体定义方法如下:

//设置静态网站关联的存储桶名称，必须设置

void SetBucket(const Aws::String& value);

//设置静态网站配置参数的容器，WebsiteConfiguration类型，必须设置。

void SetWebsiteConfiguration(const WebsiteConfiguration& value);

WebsiteConfiguration类，具体方法如下:

//设置静态网站的错误文档信息

void SetErrorDocument(const ErrorDocument& value);

//设置静态网站的索引文档

void SetIndexDocument(const IndexDocument& value);

//设置重定向所有请求配置信息，使用了重定向规则就不能配置其他规则。

void SetRedirectAllRequestsTo(const RedirectAllRequestsTo& value);

//设置重定向规则配置

void SetRoutingRules(const Aws::Vector<RoutingRule>& value);

ErrorDocument类表示静态网站的错误文档配置，其具体方法如下：

//设置静态网站错误文档的key

void SetKey(const Aws::String& value);

IndexDocument类表示静态网站的索引文档配置，其具体方法如下：

//设置静态网站索引文档名称

void SetSuffix(const Aws::String& value);

RedirectAllRequestsTo类描述静态网站所有请求的重定向行为，其具体方法如下：

//设置重定向主机名

void SetHostName(const Aws::String& value)；

//设置请求重定向协议

void SetProtocol(const Protocol& value)

从定向协议如下：

enum class Protocol

  {

    NOT\_SET,

    http,

    https

  };

RoutingRule类表示重定向规则，其具体方法如下：

//重定向规则的条件配置接口

void SetCondition(const Condition& value);

//重定向规则，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。RoutingRules中的必要配置。

void SetRedirect(const Redirect& value);

Condition类描述静态网站重定向规则条件，其具体方法如下：

//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。当、、//condition配置后，HttpErrorCodeReturnedEquals 和KeyPrefixEquals 两者只能配置//一个。

void SetHttpErrorCodeReturnedEquals(const Aws::String& value);

//指定重定向规则的对象键前缀匹配条件,当condition配置后，//HttpErrorCodeReturnedEquals 和KeyPrefixEquals 两者只能配置一个。

void SetKeyPrefixEquals(const Aws::String& value);

Redirect类具体方法如下：

//重定向主机名设置接口

void SetHostName(const Aws::String& value);

//重定向请求http返回码规则的设置接口

void SetHttpRedirectCode(const Aws::String& value);

//重定向请求使用的协议的获取和设置接口

void SetProtocol(const Protocol& value);

#### 返回结果说明

* PutBucketWebsiteOutcome :

类型Aws::S3::Model::PutBucketWebsiteOutcome ，PutBucketWebsite 请 求接口返回参数，定义的方法如下:

//本次请求是否成功

void IsSuccess()

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

#### 示例

#include <cstdio>

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/CreateBucketRequest.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

#include <aws/s3/model/PutBucketWebsiteRequest.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

bool PutWebsiteConfig(S3Client \* client,const Aws::String& bucketName,

const Aws::String& indexPage, const Aws::String& errorPage,

const Aws::String& region)

{

ClientConfiguration config;

if (!region.empty())

{

config.region = region;

}

IndexDocument index\_doc;

index\_doc.SetSuffix(indexPage);

ErrorDocument error\_doc;

error\_doc.SetKey(errorPage);

WebsiteConfiguration website\_config;

website\_config.SetIndexDocument(index\_doc);

website\_config.SetErrorDocument(error\_doc);

PutBucketWebsiteRequest request;

request.SetBucket(bucketName);

request.SetWebsiteConfiguration(website\_config);

PutBucketWebsiteOutcome outcome =

client->PutBucketWebsite(request);

if (outcome.IsSuccess())

{

std::cout << "Success: Set website configuration for bucket '"

<< bucketName << "'." << std::endl;

return true;

}

else

{

std::cout << "Error: PutBucketWebsite: "

<< outcome.GetError().GetMessage() << std::endl;

return false;

}

return 1;

}

int main()

{

printf("hello from zos\_sdk\_test!\n");

Aws::SDKOptions options;

Aws::InitAPI(options);

{

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

const Aws::String bucket\_name = "rgwuser01-testbucket03";

//TODO: Set to the region in which the bucket was created.

const Aws::String region = "us-east-1";

//TODO: Create these two files to serve as your website

const Aws::String index\_page = "index.html";

const Aws::String error\_page = "404.html";

if (!PutWebsiteConfig(&client, bucket\_name, index\_page, error\_page, region))

{

return 1;

}

}

Aws::ShutdownAPI(options);

return 0;

}

## 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

Aws::S3::Model::GetBucketWebsiteOutcome GetBucketWebsite(const Aws::S3::Model::GetBucketWebsiteRequest& request) const;

#### 参数说明

* request:方法GetBucketWebsite请求接口的参数，具体定义方法如下:

//设置静态网站关联的存储桶名称，必须设置

void SetBucket(const Aws::String& value);

//获取静态网站关联的存储桶名称

const Aws::String& GetBucket() const;

#### 返回结果说明

* 返回结果为GetBucketWebsiteOutcome类型,该类为模板类，具体的返回值为GetBucketWebsiteResult,具体方法如下：

//获取静态网站的错误文档配置

const ErrorDocument& GetErrorDocument()const；

//获取静态网站的索引文档

const IndexDocument& GetIndexDocument() const;

//获取重定向所有请求配置，

const RedirectAllRequestsTo& GetRedirectAllRequestsTo() const;

//获取重定向规则配置列表

const Aws::Vector<RoutingRule>& GetRoutingRules();

ErrorDocument类表示静态网站的错误文档配置，其具体方法如下：

//获取静态网站错误文档的KEY

const Aws::String& GetKey() const;

IndexDocument类表示静态网站的索引文档配置，其具体方法如下：

//获取静态网站索引文档名称

const Aws::String& GetSuffix()const;

RedirectAllRequestsTo类描述静态网站所有请求的重定向行为，其具体方法如下：

//获取重定向主机名称

const Aws::String& GetHostName() const;

//获取请求重定向协议

const Protocol& GetProtocol() const;

从定向协议如下：

enum class Protocol

  {

    NOT\_SET,

    http,

    https

  };

RoutingRule类表示重定向规则，其具体方法如下：

//重定向规则的条件获取接口

const Condition& GetCondition() const;

//获取重定向规则

const Redirect& GetRedirect() const;

Condition类描述静态网站重定向规则条件，其具体方法如下：

//获取重定向规则的错误码匹配条件

const Aws::String& GetHttpErrorCodeReturnedEquals() const;

//获取重定向规则的对象键前缀匹配条件

void SetKeyPrefixEquals(const Aws::String& value);

Redirect类具体方法如下：

//重定向主机名获取接口

const Aws::String& GetHostName() const；

//重定向请求http返回码规则的设置和获取接口

const Aws::String& GetHttpRedirectCode()const;

//重定向请求使用的协议的获取和设置接口

const Protocol& GetProtocol() const;

#### 示例

#include <cstdio>

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/CreateBucketRequest.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

#include <aws/s3/model/GetBucketWebsiteRequest.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void GetWebsiteConfigure(S3Client\* client, const Aws::String& bucketName)

{

Aws::S3::Model::GetBucketWebsiteRequest request;

request.SetBucket(bucketName);

Aws::S3::Model::GetBucketWebsiteOutcome outcome =

client->GetBucketWebsite(request);

if (outcome.IsSuccess())

{

Aws::S3::Model::GetBucketWebsiteResult result = outcome.GetResult();

std::cout << "Success: GetBucketWebsite: "

<< std::endl << std::endl

<< "For bucket '" << bucketName << "':"

<< std::endl

<< "Index page : "

<< result.GetIndexDocument().GetSuffix()

<< std::endl

<< "Error page: "

<< result.GetErrorDocument().GetKey()

<< std::endl;

}

else

{

auto err = outcome.GetError();

std::cout << "Error: GetBucketWebsite: "

<< err.GetMessage() << std::endl;

}

}

int main()

{

printf("hello from zos\_sdk\_test!\n");

Aws::SDKOptions options;

Aws::InitAPI(options);

{

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

GetWebsiteConfigure(&client,"rgwuser01-testbucket03");

}

Aws::ShutdownAPI(options);

return 0;

}

## 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

Aws::S3::Model::DeleteBucketWebsiteOutcome DeleteBucketWebsite(const Aws::S3::Model::DeleteBucketWebsiteRequest& request) const;

#### 参数说明

* request:方法DeleteBucketWebsite请求接口的参数，具体定义方法如下:

//设置静态网站关联的存储桶名称，必须设置

void SetBucket(const Aws::String& value);

//获取静态网站关联的存储桶名称

const Aws::String& GetBucket() const;

#### 返回结果说明

* DeleteBucketWebsiteOutcome:

类型Aws::S3::Model::DeleteBucketWebsiteOutcome， DeleteBucketWebsite 请 求接口 返回参数，定义的方法如下:

//本次请求是否成功

void IsSuccess()

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

#### 示例

#include <cstdio>

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/CreateBucketRequest.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

#include <aws/s3/model/GetBucketWebsiteRequest.h>

#include <aws/s3/model/DeleteBucketWebsiteRequest.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void DeleteBucketWebsiteConfig(S3Client\* client, const Aws::String& bucketName)

{

Aws::S3::Model::DeleteBucketWebsiteRequest request;

request.SetBucket(bucketName);

Aws::S3::Model::DeleteBucketWebsiteOutcome outcome =

client->DeleteBucketWebsite(request);

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "Error: DeleteBucketWebsite: " <<

err.GetExceptionName() << ": " << err.GetMessage() << std::endl;

}

}

int main()

{

printf("hello from zos\_sdk\_test!\n");

Aws::SDKOptions options;

Aws::InitAPI(options);

{

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

DeleteBucketWebsiteConfig(&client,"rgwuser01-testbucket03");

}

Aws::ShutdownAPI(options);

return 0;

}

## 1.17、Put Bucket Request Payment

#### 功能说明

Put Bucket Request Payment请求用于设置bucket的请求支付配置。默认情况下，bucket所有者为bucket的下载付费。该接口可以使bucket所有者能够指定请求下载的人将为下载付费。

#### 方法原型

Aws::S3::Model::PutBucketRequestPaymentOutcome S3Client::PutBucketRequestPayment(const Aws::S3::Model::PutBucketRequestPaymentRequest& request) const

#### 参数说明

* request: Aws::S3::Model::PutBucketRequestPaymentRequest类型，该类定义的方法有:

// 指定为哪一个bucket设置请求者付费，必须要设置

void SetBucket(const Aws::String& value)

// 设置请求付费配置，这里是设置RequestPaymentConfiguration，

// RequestPaymentConfiguration是Payer的容器

void SetRequestPaymentConfiguration(const RequestPaymentConfiguration

& value)

PutBucketRequestPaymentRequest是存储设置请求付费规则的容器类，容器 中包含Payer，从而指定是桶的拥有者付费还是具体的请求者付费。 PutBucketRequestPaymentRequest类中定义中定义的方法有：

// 设置请求者付费，value的值可以为Requester或者BucketOwner

void SetPayer(const Payer& value)

Payer是枚举类，用于指定具体的请求付费者

enum class Payer

{

NOT\_SET,

Requester,

BucketOwner

};

#### 返回结果说明

* 返回结果类型为Aws::S3::Model::PutBucketRequestPaymentOutcome，该类型定义的方法有:

// 本次请求是否成功

void IsSuccess(const Aws::String& value)

// 获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

#### 示例

#include <aws/s3/S3Client.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

#include <aws/s3/model/PutBucketRequestPaymentRequest.h>

#include <aws/s3/model/RequestPaymentConfiguration.h>

#include <aws/s3/model/Payer.h>

#include <aws/core/http/Scheme.h>

int main(int argc, char\* argv[])

{

// 设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

Aws::Client::ClientConfiguration cfg;

cfg.endpointOverride = "http://192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

Aws::Auth::AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

// 创建S3Client

Aws::S3::S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

const Aws::String bucket\_name = "bucket1";

Aws::S3::Model::RequestPaymentConfiguration config;

config.SetPayer(Aws::S3::Model::Payer::Requester);

Aws::S3::Model::PutBucketRequestPaymentRequest request;

request.SetRequestPaymentConfiguration(config);

request.SetBucket(bucket\_name);

auto outcome = client.PutBucketRequestPayment(request);

if (outcome.IsSuccess())

{

std::cout << "Put Bucket Request Payment successfully." << std::endl;

} else

{

auto err = outcome.GetError();

std::cout << "PutBucketPolicy ERROR" << ", Http code: "<< (int)err.GetResponseCode() <<

", Error Type:" << (int)err.GetErrorType() << ", Error Msg: "

<< err.GetMessage() << std::endl;

}

Aws::ShutdownAPI(options);

return 0;

}

## 1.18、Get Bucket Request Payment

#### 功能说明

Get Bucket Request Payment请求用于获取bucket的请求支付配置。

#### 方法原型

Aws::S3::Model::GetBucketRequestPaymentOutcome S3Client::GetBucketRequestPayment(const Aws::S3::Model::GetBucketRequestPaymentRequest& request) const

#### 参数说明

* request: Aws::S3::Model::GetBucketRequestPaymentRequest类型，该类定义的方法有:

//指定获取哪一个bucket的请求者付费设置，必须要设置

void SetBucket(const Aws::String& value)

#### 返回结果说明

* 返回结果类型为Aws::S3::Model::GetBucketRequestPaymentOutcome，该类型定义的方法有:

// 本次请求是否成功

void IsSuccess(const Aws::String& value)

// 获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

// 获取bucket request payment result接口

const GetBucketRequestPaymentResult& GetResult() const

GetBucketRequestPaymentResult类中封装了GetPayer()方法，用于获取Payer

const Payer& GetPayer() const

Payer是枚举类，为具体的请求付费者

enum class Payer

{

NOT\_SET,

Requester,

BucketOwner

};

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

#include <aws/s3/model/GetBucketRequestPaymentRequest.h>

#include <aws/s3/model/Payer.h>

#include <aws/core/http/Scheme.h>

int main(int argc, char\* argv[])

{

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

//设置连接参数

Aws::Client::ClientConfiguration cfg;

cfg.endpointOverride = "http://192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

Aws::Auth::AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

//创建S3Client

Aws::S3::S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

const Aws::String bucket\_name = "bucket1";

Aws::S3::Model::GetBucketRequestPaymentRequest request;

request.SetBucket(bucket\_name);

auto outcome = client.GetBucketRequestPayment(request);

if (outcome.IsSuccess())

{

std::cout << "Get Bucket Request Payment successfully." << std::endl;

Aws::S3::Model::Payer payer = outcome.GetResult().GetPayer();

Aws::String payer\_name = Aws::S3::Model::PayerMapper::GetNameForPayer(payer);

std::cout << "Payer: " << payer\_name << std::endl;

} else

{

auto err = outcome.GetError();

std::cout << "PutBucketPolicy ERROR" << ", Http code: "

<< (int)err.GetResponseCode()

<< ", Error Type:" << (int)err.GetErrorType()

<< ", Error Msg: " << err.GetMessage() << std::endl;

}

Aws::ShutdownAPI(options);

return 0;

}

## 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

Aws::S3::Model::PutBucketTaggingOutcome S3Client::PutBucketTagging(const PutBucketTaggingRequest& request) const

#### 参数说明

* request: Aws::S3::Model::PutBucketTaggingRequest类型，该类定义的方法有:

// 指定桶名称

void SetBucket(const Aws::String& value)

// 指定标签集

void SetTagging(const Tagging& value)

Tagging类型，该类定义的方法有:

// 指定指定标签列表

void SetTagSet(const Aws::Vector<Tag>& value)

Tag类型，该类定义的方法有:

// 指定标签key，最大128字节

void SetKey(const Aws::String& value)

// 指定标签value，最大256字节

void SetValue(const Aws::String& value)

#### 返回结果说明

* 返回结果类型为Aws::S3::Model::PutBucketTaggingOutcome，该类型定义的方法有:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/PutBucketTaggingRequest.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void put\_bucket\_tagging(S3Client &client, const Aws::String &bucket\_name)

{

PutBucketTaggingRequest request;

request.SetBucket(bucket\_name);

Tag tag;

tag.SetKey("key1");

tag.SetValue("val1");

Aws::Vector<Tag> tags;

tags.push\_back(tag);

Tagging tagging;

tagging.SetTagSet(tags);

request.SetTagging(tagging);

auto outcome = client.PutBucketTagging(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: " << "Http code: "<< (int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " << err.GetMessage() << std::endl;

} else {

std::cout << "successful : " << endl;

}

return;

}

int main(int argc, char\* argv[])

{

if(argc != 2)

{

std::cout << "pls input bucket\_name" << std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

put\_bucket\_tagging(client, bucket\_name);

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model::GetBucketTaggingOutcome S3Client::GetBucketTagging(const GetBucketTaggingRequest& request) const

#### 参数说明

* request: Aws::S3::Model::GetBucketTaggingRequest类型，该类定义的方法有:

// 指定桶名称

void SetBucket(const Aws::String& value)

#### 返回结果说明

* 返回结果类型为Aws::S3::Model::GetBucketTaggingOutcome，该类型定义的方法有:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

//获取应答结果

const GetBucketTaggingResult& GetResult() const

GetBucketTaggingResult，该类型定义的方法有:

// 获取标签列表

const Aws::Vector<Tag>& GetTagSet() const

Tag，该类型定义的方法有:

// 获取标签key

const Aws::String& GetKey() const

// 获取标签value

const Aws::String& GetValue() const

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/GetBucketTaggingRequest.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void get\_bucket\_tagging(S3Client &client, const Aws::String &bucket\_name)

{

GetBucketTaggingRequest request;

request.SetBucket(bucket\_name);

auto outcome = client.GetBucketTagging(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: " << "Http code: "<< (int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " << err.GetMessage() << std::endl;

} else {

std::cout << "successful : " << endl;

auto result = outcome.GetResult();

auto tags = result.GetTagSet();

for (auto it = tags.cbegin(); it != tags.cend(); ++it) {

std::cout << it->GetKey() << " = " << it->GetValue() << endl;

}

}

return;

}

int main(int argc, char\* argv[])

{

if(argc != 2)

{

std::cout << "pls input bucket\_name" << std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

get\_bucket\_tagging(client, bucket\_name);

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model::DeleteBucketTaggingOutcome S3Client::DeleteBucketTagging(const DeleteBucketTaggingRequest& request

) const

#### 参数说明

* request: Aws::S3::Model::DeleteBucketTaggingRequest类型，该类定义的方法有:

// 指定桶名称

void SetBucket(const Aws::String& value)

#### 返回结果说明

* 返回结果类型为Aws::S3::Model::DeleteBucketTaggingOutcome，该类型定义的方法有:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/DeleteBucketTaggingRequest.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void delete\_bucket\_tagging(S3Client &client, const Aws::String &bucket\_name)

{

DeleteBucketTaggingRequest request;

request.SetBucket(bucket\_name);

auto outcome = client.DeleteBucketTagging(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: " << "Http code: "<<(int)err.GetResponseCode()

<< " Error Type:" << (int)err.GetErrorType() << " Error Msg: " << err.GetMessage() << std::endl;

} else {

std::cout << "successful : " << endl;

}

return;

}

int main(int argc, char\* argv[])

{

if(argc != 2)

{

std::cout << "pls input bucket\_name" << std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

delete\_bucket\_tagging(client, bucket\_name);

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model:: PutBucketEncryptionOutcome Aws::S3::S3Client:: PutBucketEncryption(const Aws::S3::Model:: PutBucketEncryptionRequest&

request) const

#### 参数说明

* request:类型Aws::S3::Model::PutBucketEncryptionRequest，启用Bucket加密请求接口参数，定义的方法如下:

//设置Bucket名称,必须设置，Aws::String可通过标准std::string赋值

void SetBucket(const Aws::String& value)

//设置服务端加密配置

void SetServerSideEncryptionConfiguration(const ServerSideEncryptionConfiguration& value)

类型Aws::S3::Model::ServerSideEncryptionByDefault，设置Bucket加密算法和密钥接口参数，定义的方法如下:

//设置Bucket的加密算法, ServerSideEncryption是个enum class类型，从小到大分别是NOT\_SET，AES256，aws\_kms，分别对应数字0到2

void SetSSEAlgorithm(Aws::S3::Model:: ServerSideEncryption&& value)

//设置加密密钥或CMKID，当算法是AES256时，可以调用此函数设置加密密钥，但字符长度需为32，也可以不设置加密密钥，系统会自动生成；当算法是aws\_kms时，此项必须设置，且按照cmkuuid:keyspec:userid模式配置，其中cmkuuid是KMSID，keyspec是指定生成的数据密钥长度，userid是用户id。

void SetKMSMasterKeyID(const char\* value)

类型Aws::S3::Model:: ServerSideEncryptionRule，设置Bucket加密规则接口参数，定义的方法如下:

//设置Bucket的加密规则

void SetApplyServerSideEncryptionByDefault(const ServerSideEncryptionByDefault& value)

类型Aws::S3::Model:: ServerSideEncryptionConfiguration，设置Bucket加密配置接口参数，定义的方法如下:

//设置Bucket的加密配置

void SetRules(const Aws::Vector<ServerSideEncryptionRule>& value)

#### 返回结果说明

* PutBucketEncryptionOutCome:类型Aws::S3::Model:: PutBucketEncryptionOutCome，启用Bucket默认加密功能接口返回参数，定义的方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/PutBucketEncryptionRequest.h>

#include <aws/core/Aws.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void put\_bucket\_encryption(S3Client& client, const Aws::String& bucket\_name) {

// 设置请求参数

PutBucketEncryptionRequest request;

request.SetBucket(bucket\_name);

ServerSideEncryptionByDefault SSE\_Default;

SSE\_Default.SetSSEAlgorithm(ServerSideEncryption::aws\_kms);

SSE\_Default.SetKMSMasterKeyID("6b1f657c-816b-4534-a41a-903e7a60e703:AES\_256:e3d16fba6ae84e33a1d386dd880696c0");

ServerSideEncryptionRule SSE\_Rule;

SSE\_Rule.SetApplyServerSideEncryptionByDefault(SSE\_Default);

Vector<ServerSideEncryptionRule> V\_SSE;

V\_SSE.push\_back(SSE\_Rule);

ServerSideEncryptionConfiguration SSE\_Config;

SSE\_Config.SetRules(V\_SSE);

request.SetServerSideEncryptionConfiguration(SSE\_Config);

// 发出请求

PutBucketEncryptionOutcome outcome = client.PutBucketEncryption(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: PutBucketEncryption: " << "Http code: " <<

(int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " <<

err.GetMessage() << std::endl;

}

else {

std::cout << "successful put bucket encryption: " << bucket\_name

<< "\n";

}

return;

}

int main(int argc, char\* argv[])

{

if (argc != 2)

{

std::cout << "pls input bucket\_name" << std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

put\_bucket\_encryption(client, bucket\_name);

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model:: GetBucketEncryptionOutcome Aws::S3::S3Client:: PutBucketEncryption(const Aws::S3::Model:: GetBucketEncryptionRequest&

request) const

#### 参数说明

* request:类型Aws::S3::Model::GetBucketEncryptionRequest，获取Bucket加密请求接口参数，定义的方法如下:

//设置Bucket名称,必须设置，Aws::String可通过标准std::string赋值

void SetBucket(const Aws::String& value)

#### 返回结果说明

* GetBucketEncryptionOutCome:类型Aws::S3::Model:: GetBucketEncryptionOutCome，获取Bucket默认加密配置接口返回参数，定义的方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

//获取存储桶加密配置结果

GetBucketEncryptionResult& GetResult()

类型Aws::S3::Model::GetBucketEncryptionResult，获取Bucket加密配置，定义的方法如下:

//获取Bucket的加密配置

const ServerSideEncryptionConfiguration&

GetServerSideEncryptionConfiguration() const

类型Aws::S3::Model::ServerSideEncryptionConfiguration，获取Bucket加密规则，定义的方法如下:

//获取Bucket的加密配置

const Aws::Vector<ServerSideEncryptionRule>& GetRules() const

类型Aws::S3::Model::ServerSideEncryptionRule，获取Bucket默认加密配置，定义的方法如下:

//获取Bucket的默认加密配置

const ServerSideEncryptionByDefault& GetApplyServerSideEncryptionByDefault() const

类型Aws::S3::Model::ServerSideEncryptionByDefault，获取Bucket默认加密配置，定义的方法如下:

//获取Bucket的默认加密算法

const ServerSideEncryption& GetSSEAlgorithm() const

//获取Bucket的默认加密密钥

const Aws::String& GetKMSMasterKeyID() const

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/GetBucketEncryptionRequest.h>

#include <aws/core/Aws.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

const char\* kms\_algorithm[] = {"NOT\_SET","AES256","aws\_kms"};

void get\_bucket\_encryption(S3Client& client, const Aws::String& bucket\_name) {

// 设置请求参数

GetBucketEncryptionRequest request;

request.SetBucket(bucket\_name);

// 发出请求

GetBucketEncryptionOutcome outcome = client.GetBucketEncryption(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: GetBucketEncryption: " << "Http code: " <<

(int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " <<

err.GetMessage() << std::endl;

}

else {

std::cout << "successful get bucket encryption: " << bucket\_name

<< "\n";

GetBucketEncryptionResult outresult = outcome.GetResult();

ServerSideEncryptionConfiguration SSE\_Config = outresult.GetServerSideEncryptionConfiguration();

Vector<ServerSideEncryptionRule> V\_SSE\_Rule = SSE\_Config.GetRules();

for (auto& SSE\_Rule : V\_SSE\_Rule) {

ServerSideEncryptionByDefault SSE\_Default = SSE\_Rule.GetApplyServerSideEncryptionByDefault();

ServerSideEncryption SSE\_Algorithm = SSE\_Default.GetSSEAlgorithm();

Aws::String SSE\_KeyId = SSE\_Default.GetKMSMasterKeyID();

cout << "SSEAlgorithm: " << kms\_algorithm[int(SSE\_Algorithm)]

<< endl;

cout << "KMSMasterKeyID: " << SSE\_KeyId << endl;

}

}

return;

}

int main(int argc, char\* argv[])

{

if (argc != 2)

{

std::cout << "pls input bucket\_name" << std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

get\_bucket\_encryption(client, bucket\_name);

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model:: DeleteBucketEncryptionOutcome Aws::S3::S3Client:: DeleteBucketEncryption(const Aws::S3::Model:: DeleteBucketEncryptionRequest& request) const

#### 参数说明

* request:类型Aws::S3::Model::DeleteBucketEncryptionRequest，删除Bucket加密请求接口参数，定义的方法如下:

//设置Bucket名称,必须设置，Aws::String可通过标准std::string赋值

void SetBucket(const Aws::String& value)

#### 返回结果说明

* DeleteBucketEncryptionOutCome:类型Aws::S3::Model:: DeleteBucketEncryptionOutCome，删除Bucket默认加密功能接口返回参数，定义的方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/DeleteBucketEncryptionRequest.h>

#include <aws/core/Aws.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void delete\_bucket\_encryption(S3Client& client, const Aws::String& bucket\_name) {

// 设置请求参数

DeleteBucketEncryptionRequest request;

request.SetBucket(bucket\_name);

// 发出请求

DeleteBucketEncryptionOutcome outcome = client.DeleteBucketEncryption(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: DeleteBucketEncryption: " << "Http code: "

<< (int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " <<

err.GetMessage() << std::endl;

}

else {

std::cout << "successful delete bucket encryption: " << bucket\_name

<< "\n";

}

return;

}

int main(int argc, char\* argv[])

{

if (argc != 2)

{

std::cout << "pls input bucket\_name" << std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

delete\_bucket\_encryption(client, bucket\_name);

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model::PutObjectLockConfigurationOutcome Aws::S3::S3Client::PutObjectLockConfiguration(const Aws::S3::Model::PutObjectLockConfigurationRequest& request) const

#### 参数说明

* request:类型Aws::S3::Model::PutObjectLockConfigurationRequest，启用Bucket对象锁定请求接口参数，定义的方法如下:

//设置Bucket名称，必须设置，Aws::String可通过标准std::string赋值

void SetBucket(const Aws::String& value)

//设置对象锁定配置

void SetObjectLockConfiguration(const ObjectLockConfiguration& value)

类型Aws::S3::Model::ObjectLockConfiguration，设置Bucket对象锁定接口参数，定义的方法如下:

//设置Bucket的对象锁定，ObjectLockEnabled是个enum class类型，从小到大分别是NOT\_SET，Enabled分别对应数字0到1

void SetObjectLockEnabled(ObjectLockEnabled&& value)

//设置Bucket的对象锁定规则

void SetRule(const ObjectLockRule& value)

类型Aws::S3::Model::ObjectLockRule，设置Bucket对象锁定规则接口参数，定义的方法如下:

//设置Bucket对象锁定的默认保留期限

void SetDefaultRetention(const DefaultRetention& value)

类型Aws::S3::Model::DefaultRetention，设置Bucket对象锁定默认保留期限接口参数，定义的方法如下:

//设置Bucket对象锁定模式

void SetMode(ObjectLockRetentionMode&& value)

//设置Bucket对象锁定保留天数

void SetDays(int value)

//设置Bucket对象锁定保留年数

void SetYears(int value)

#### 返回结果说明

* PutObjectLockConfigurationOutcome:类型Aws::S3::Model:: PutObjectLockConfigurationOutcome，启用Bucket对象锁定功能接口返回参数，定义的方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/PutObjectLockConfigurationRequest.h>

#include <aws/core/Aws.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void put\_bucket\_object\_lock(S3Client& client, const Aws::String& bucket\_name) {

// 设置请求参数

PutObjectLockConfigurationRequest request;

request.SetBucket(bucket\_name);

DefaultRetention defretention;

defretention.SetMode(ObjectLockRetentionMode::COMPLIANCE);

//defretention.SetDays(1);

defretention.SetYears(1);

ObjectLockRule objectrule;

objectrule.SetDefaultRetention(defretention);

ObjectLockConfiguration objectlock;

objectlock.SetObjectLockEnabled(ObjectLockEnabled::Enabled);

objectlock.SetRule(objectrule);

request.SetObjectLockConfiguration(objectlock);

// 发出请求

PutObjectLockConfigurationOutcome outcome = client.PutObjectLockConfiguration(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: PutObjectLockConfiguration: " << "Http code:

" << (int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " <<

err.GetMessage() << std::endl;

}

else {

std::cout << "successful put bucket object lock: " << bucket\_name

<< "\n";

}

return;

}

int main(int argc, char\* argv[])

{

if (argc != 2)

{

std::cout << "pls input bucket\_name" << std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

put\_bucket\_object\_lock(client, bucket\_name);

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model::GetObjectLockConfigurationOutcome Aws::S3::S3Client::GetObjectLockConfiguration(const Aws::S3::Model::GetObjectLockConfigurationRequest& request) const

#### 参数说明

* request:类型Aws::S3::Model::GetObjectLockConfigurationRequest，获取Bucket对象锁定配置请求接口参数，定义的方法如下:

//设置Bucket名称，必须设置，Aws::String可通过标准std::string赋值

void SetBucket(const Aws::String& value)

#### 返回结果说明

* GetObjectLockConfigurationOutCome:类型Aws::S3::Model:: GetObjectLockConfigurationOutCome，获取Bucket对象锁定配置接口返回参数，定义的方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

//获取存储桶对象锁定配置结果

GetObjectLockConfigurationResult& GetResult()

类型Aws::S3::Model::GetObjectLockConfigurationResult，获取Bucket对象锁定配置结果，定义的方法如下:

//获取Bucket的加密配置

const ObjectLockConfiguration& GetObjectLockConfiguration() const

类型Aws::S3::Model::ObjectLockConfiguration，获取Bucket对象锁定配置，定义的方法如下:

//获取Bucket对象锁定开启标识

const ObjectLockEnabled& GetObjectLockEnabled() const

//获取Bucket对象锁定规则

const ObjectLockRule& GetRule() const

类型Aws::S3::Model::ObjectLockRule，获取Bucket对象锁定规则，定义的方法如下:

//获取Bucket对象锁定默认保留期限

const DefaultRetention& GetDefaultRetention() const

类型Aws::S3::Model::DefaultRetention，获取Bucket对象锁定默认保留期限，定义的方法如下:

//获取Bucket对象锁定默认保留期限

const ObjectLockRetentionMode& GetMode() const

//获取Bucket对象锁定保留期限天数

int GetDays() const

//获取Bucket对象锁定保留期限年数

int GetYears() const

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/GetObjectLockConfigurationRequest.h>

#include <aws/core/Aws.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

const char\* object\_lock\_enable[] = { "NOT\_SET", "Enabled" };

const char\* object\_lock\_retention\_mode[] = { "NOT\_SET", "GOVERNANCE", "COMPLIANCE" };

void get\_bucket\_object\_lock(S3Client& client, const Aws::String& bucket\_name) {

// 设置请求参数

GetObjectLockConfigurationRequest request;

request.SetBucket(bucket\_name);

// 发出请求

GetObjectLockConfigurationOutcome outcome = client.GetObjectLockConfiguration(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: GetObjectLockConfiguration: " << "Http code:

" << (int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " <<

err.GetMessage() << std::endl;

}

else {

std::cout << "successful get bucket object lock: " << bucket\_name

<< "\n";

GetObjectLockConfigurationResult result = outcome.GetResult();

ObjectLockConfiguration objectlockconfig = result.GetObjectLockConfiguration();

ObjectLockEnabled object\_enable = objectlockconfig.GetObjectLockEnabled();

cout << "ObjectLockEnabled: " << object\_lock\_enable[int(object\_enable)] << endl;

ObjectLockRule object\_rule = objectlockconfig.GetRule();

DefaultRetention default\_retention = object\_rule.GetDefaultRetention();

ObjectLockRetentionMode mode = default\_retention.GetMode();

cout << "ObjectLockRetentionMode: " << object\_lock\_retention\_mode[int(mode)] << endl;

int days = default\_retention.GetDays();

int years = default\_retention.GetYears();

cout << "Days: " << days << endl;

cout << "Years: " << years << endl;

}

return;

}

int main(int argc, char\* argv[])

{

if (argc != 2)

{

std::cout << "pls input bucket\_name" << std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

get\_bucket\_object\_lock(client, bucket\_name);

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model::PutBucketLoggingOutcome Aws::S3::S3Client::PutBucketLogging (const Aws::S3::Model::PutBucketLoggingRequest& request) const

#### 参数说明

* request:类型Aws::S3::Model::PutBucketLoggingRequest，启用Bucket日志转存请求接口参数，定义的方法如下:

//设置Bucket名称，必须设置，Aws::String可通过标准std::string赋值

void SetBucket(const Aws::String& value)

//设置日志转存配置，配置空的BucketLoggingStatus参数即为关闭日志转存功能

void SetBucketLoggingStatus(const BucketLoggingStatus& value)

类型Aws::S3::Model::BucketLoggingStatus，设置Bucket日志转存状态接口参数，定义的方法如下:

//设置Bucket的日志转存功能

void SetLoggingEnabled(const LoggingEnabled& value)

类型Aws::S3::Model::LoggingEnabled，设置Bucket日志转存接口参数，定义的方法如下:

//设置日志转存功能的目标存储桶

void SetTargetBucket(const Aws::String& value)

//设置日志转存功能的目标前缀

void SetTargetPrefix(const char\* value)

//设置日志转存功能的目标授权

void SetTargetGrants(const Aws::Vector<TargetGrant>& value)

类型Aws::S3::Model::TargetGrant，设置Bucket日志转存目标授权接口参数，定义的方法如下:

//设置日志转存功能授权许可，BucketLogsPermission是个enum class类型，从小到大分别是NOT\_SET，FULL\_CONTROL，READ，WRITE分别对应数字0到3

void SetPermission(BucketLogsPermission&& value)

//设置日志转存功能授权信息

void SetGrantee(const Grantee& value)

类型Aws::S3::Model::Grantee，设置Bucket日志转存授权信息，定义的方法如下:

//设置日志转存授权类型，Type是个enum class类型，从小到大分别是NOT\_SET，CanonicalUser，AmazonCustomerByEmail，WRITE分别对应数字0到2

void SetType(Type&& value)

//设置日志转存授权显示名称

void SetDisplayName(const char\* value)

//设置日志转存授权用户ID

void SetID(const char\* value)

//设置日志转存授权用户邮箱地址

void SetEmailAddress(const char\* value)

#### 返回结果说明

* PutBucketLoggingOutcome:类型Aws::S3::Model:: PutBucketLoggingOutcome，启用Bucket日志转存功能接口返回参数，定义的方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/PutBucketLoggingRequest.h>

#include <aws/core/Aws.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void put\_bucket\_logging(S3Client& client,

const Aws::String& bucket\_name) {

// 设置请求参数

PutBucketLoggingRequest request;

request.SetBucket(bucket\_name);

Grantee grantee;

grantee.SetType(Type::CanonicalUser);

grantee.SetDisplayName("Second User");

grantee.SetID("s3");

TargetGrant target\_grant;

target\_grant.SetPermission(BucketLogsPermission::FULL\_CONTROL);

target\_grant.SetGrantee(grantee);

Aws::Vector<TargetGrant> v\_target\_bucket;

v\_target\_bucket.push\_back(target\_grant);

LoggingEnabled logging\_enable;

logging\_enable.SetTargetBucket("target\_bucket\_sdk");

logging\_enable.SetTargetPrefix("log/");

logging\_enable.SetTargetGrants(v\_target\_bucket);

BucketLoggingStatus bucket\_logging\_status;

bucket\_logging\_status.SetLoggingEnabled(logging\_enable);

request.SetBucketLoggingStatus(bucket\_logging\_status);

// 发出请求

PutBucketLoggingOutcome outcome = client.PutBucketLogging(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: PutBucketLogging: " << "Http code: " << (int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " <<

err.GetMessage() << std::endl;

}

else {

std::cout << "successful put bucket logging: " << bucket\_name <<

"\n";

}

return;

}

int main(int argc, char\* argv[])

{

if (argc != 2)

{

std::cout << "pls input bucket\_name" << std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

put\_bucket\_logging(client, bucket\_name);

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model::GetBucketLoggingOutcome Aws::S3::S3Client::GetBucketLogging(const Aws::S3::Model::GetBucketLoggingRequest& request) const

#### 参数说明

* request:类型Aws::S3::Model::GetBucketLoggingRequest，获取Bucket日志转存配置请求接口参数，定义的方法如下:

//设置Bucket名称，必须设置，Aws::String可通过标准std::string赋值

void SetBucket(const Aws::String& value)

#### 返回结果说明

* GetBucketLoggingOutCome:类型Aws::S3::Model:: GetBucketLoggingOutCome，获取Bucket日志转存配置接口返回参数，定义的方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

//获取存储桶日志转存配置结果

GetBucketLoggingResult& GetResult()

类型Aws::S3::Model::GetBucketLoggingResult，获取Bucket日志转存配置结果，定义的方法如下:

//获取Bucket的加密配置

const LoggingEnabled& GetLoggingEnabled() const

类型Aws::S3::Model::LoggingEnabled，获取Bucket日志转存配置，定义的方法如下:

//获取日志转存的目标存储桶

const Aws::String& GetTargetBucket() const

//获取日志转存的目标前缀

const Aws::String& GetTargetPrefix() const

//获取日志转存的目标授权信息

const Aws::Vector<TargetGrant>& GetTargetGrants() const

类型Aws::S3::Model::TargetGrant，获取Bucket日志转存目标授权，定义的方法如下:

//获取日志转存授权许可，BucketLogsPermission是个enum class类型，从小到大分别是NOT\_SET，FULL\_CONTROL，READ，WRITE分别对应数字0到3

const BucketLogsPermission& GetPermission() const

//获取日志转存授权组合

const Grantee& GetGrantee() const

类型Aws::S3::Model::Grantee，获取Bucket日志转存授权组合，定义的方法如下:

//获取授权显示名称

const Aws::String& GetDisplayName() const

//获取授权邮箱地址

const Aws::String& GetEmailAddress() const

//获取授权用户ID

const Aws::String& GetID() const

//获取授权类型，Type是个enum class类型，从小到大分别是NOT\_SET，CanonicalUser，AmazonCustomerByEmail，WRITE分别对应数字0到2

const Type& GetType() const

获取授权URI

const Aws::String& GetURI() const

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/GetBucketLoggingRequest.h>

#include <aws/core/Aws.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

const char\* bucket\_logs\_permission[] = { "NOT\_SET", "FULL\_CONTROL",

"READ", "WRITE" };

const char\* grantee\_type[] = { "NOT\_SET", "CanonicalUser", "AmazonCustomerByEmail", "Group" };

void get\_bucket\_logging(S3Client& client,

const Aws::String& bucket\_name) {

// 设置请求参数

GetBucketLoggingRequest request;

request.SetBucket(bucket\_name);

// 发出请求

GetBucketLoggingOutcome outcome = client.GetBucketLogging(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: GetBucketLogging: " << "Http code: " << (int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " <<

err.GetMessage() << std::endl;

}

else {

std::cout << "successful get bucket logging: " << bucket\_name <<

"\n";

GetBucketLoggingResult result = outcome.GetResult();

LoggingEnabled logging\_enable = result.GetLoggingEnabled();

Aws::String target\_bucket = logging\_enable.GetTargetBucket();

cout << "GetTargetBucket: " << target\_bucket << endl;

Aws::String target\_prefix = logging\_enable.GetTargetPrefix();

cout << "GetTargetPrefix: " << target\_prefix << endl;

Aws::Vector<TargetGrant> v\_target\_grant = logging\_enable.GetTargetGrants();

for (auto& target\_grant : v\_target\_grant) {

BucketLogsPermission permission = target\_grant.GetPermission();

cout << "GetPermission: " << bucket\_logs\_permission[int(permission)] << endl;

Grantee grantee = target\_grant.GetGrantee();

Aws::String display\_name = grantee.GetDisplayName();

cout << "GetDisplayName: " << display\_name << endl;

Aws::String email\_address = grantee.GetEmailAddress();

cout << "GetEmailAddress: " << email\_address << endl;

Aws::String id = grantee.GetID();

cout << "GetID: " << id << endl;

Type type = grantee.GetType();

cout << "GetType: " << grantee\_type[int(type)] << endl;

Aws::String uri = grantee.GetURI();

cout << "GetURI: " << uri << endl;

}

}

return;

}

int main(int argc, char\* argv[])

{

if (argc != 2)

{

std::cout << "pls input bucket\_name" << std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

get\_bucket\_logging(client, bucket\_name);

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model::PutBucketCorsOutcome PutBucketCors(const Aws::S3::Model::PutBucketCorsRequest& request) const;

#### 参数说明

* request:方法PutBucketCors请求接口的参数，PutBucketCorsRequest类型，具体定义方法如下:

//设置跨域规则的存储桶名称，必须设置

void SetBucket(const Aws::String& value);

//设置存储桶中对象的跨域访问配置参数的容器，CORSConfiguration类型，必须设置

void SetCORSConfiguration(const CORSConfiguration& value);

CORSConfiguration类描述跨域格则参数的容器，其具体方法如下：

//设置跨域规则的接口,最多允许设置100条跨域规则

void SetCORSRules(const Aws::Vector<CORSRule>& value);

CORSRule类描述跨域访问规则，其具体方法如下：

//设置允许浏览器发送 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。

void SetAllowedHeaders(const Aws::Vector<Aws::String>& value);

//设置允许该源执行的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。必须设置。

void SetAllowedMethods(const Aws::Vector<Aws::String>& value);

//设置允许能够访问该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。必须设置。

void SetAllowedOrigins(const Aws::Vector<Aws::String>& value);

//设置允许浏览器获取的 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。

void SetExposeHeaders(const Aws::Vector<Aws::String>& value);

//获取和设置跨域规则的ID,最大长度255

void SetID(const Aws::String& value);

//获取和设置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

void SetMaxAgeSeconds(int value)

#### 返回结果说明

* PutBucketCorsOutcome:类型Aws::S3::Model::PutBucketCorsOutcome，PutBucketCors 请求接口返回参数，定义的方法如下:

//本次请求是否成功

void IsSuccess()

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

#### 示例

#include <cstdio>

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/CreateBucketRequest.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

#include <aws/s3/model/PutBucketCorsRequest.h>

#include <aws/s3/model/GetBucketCorsRequest.h>

#include <aws/s3/model/DeleteBucketCorsRequest.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void PutBucketCors(S3Client\* client, const Aws::String& bucketName)

{

CORSRule rule\_1;

rule\_1.SetID("rule\_1");

Aws::Vector<Aws::String> origions;

origions.push\_back("http://100.100.8.1:80");

origions.push\_back("http://100.100.8.2:80");

rule\_1.SetAllowedOrigins(origions);

Aws::Vector<Aws::String> methods={"GET","HEAD","PUT"};

rule\_1.SetAllowedMethods(methods);

CORSRule rule\_2;

rule\_2.SetID("rule\_2");

rule\_2.SetAllowedOrigins(origions);

rule\_2.SetAllowedMethods(methods);

Aws::Vector<CORSRule> rules = { rule\_1,rule\_2 };

CORSConfiguration cors\_config;

cors\_config.SetCORSRules(rules);

PutBucketCorsRequest req;

req.SetBucket(bucketName);

req.SetCORSConfiguration(cors\_config);

client->PutBucketCors(req);

}

int main()

{

printf("hello from zos\_sdk\_test!\n");

Aws::SDKOptions options;

Aws::InitAPI(options);

{

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

PutBucketCors(&client,"rgwuser01-testbucket03");

}

Aws::ShutdownAPI(options);

return 0;

}

## 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

Aws::S3::Model::GetBucketCorsOutcome GetBucketCors(const Aws::S3::Model::GetBucketCorsRequest& request) const

#### 参数说明

* request:方法GetBucketCors请求接口的参数，GetBucketCorsRequest类型，具体方法如下:

//设置存储桶名称，必须设置

void SetBucket(const Aws::String& value);

//获取存储桶名称

const Aws::String& GetBucket() const;

#### 返回结果说明

* 返回结果为GetBucketCorsOutcome类型,该类为模板类，具体的返回值为GetBucketCorsResult,具体方法如下：

//获取跨域规则的接口

const Aws::Vector<CORSRule>& GetCORSRules() const;

CORSRule类描述跨域访问规则，其具体方法如下：

//获取允许浏览器发送 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。

const Aws::Vector<Aws::String>& GetAllowedHeaders() const;

//获取允许该源执行的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。必须设置。

const Aws::Vector<Aws::String>& GetAllowedMethods() const;

//获取允许能够访问该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。必须设置。

const Aws::Vector<Aws::String>& GetAllowedOrigins() const;

//获取允许浏览器获取的 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。

const Aws::Vector<Aws::String>& GetExposeHeaders() const;

//获取跨域规则的ID,最大长度255

const Aws::String& GetID() const;

//获取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

int GetMaxAgeSeconds() const;

#### 示例

#include <cstdio>

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/CreateBucketRequest.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

#include <aws/s3/model/PutBucketCorsRequest.h>

#include <aws/s3/model/GetBucketCorsRequest.h>

#include <aws/s3/model/DeleteBucketCorsRequest.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void GetBucketCors(S3Client\* client, const Aws::String& bucketName)

{

GetBucketCorsRequest req;

req.SetBucket(bucketName);

GetBucketCorsOutcome outcome = client->GetBucketCors(req);

if (outcome.IsSuccess())

{

GetBucketCorsResult result = outcome.GetResult();

Aws::Vector<CORSRule> rules = result.GetCORSRules();

for (auto item : rules)

{

cout << item.GetID() << endl;

Aws::Vector<String> methods = item.GetAllowedMethods();

for (auto meth : methods)

{

cout << meth << endl;

}

Aws::Vector<String> origions = item.GetAllowedOrigins();

for (auto ori : origions)

{

cout << ori << endl;

}

}

}

else

{

auto err = outcome.GetError();

std::cout << "Error: GetBucketWebsite: "

<< err.GetMessage() << std::endl;

}

}

int main()

{

printf("hello from zos\_sdk\_test!\n");

Aws::SDKOptions options;

Aws::InitAPI(options);

{

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

GetBucketCors(&client,"rgwuser01-testbucket03");

}

Aws::ShutdownAPI(options);

return 0;

}

## 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

Aws::S3::Model::DeleteBucketCorsOutcome DeleteBucketCors(

const Aws::S3::Model::DeleteBucketCorsRequest& request) const;

#### 参数说明

* request:方法DeleteBucketCors请求接口的参数，DeleteBucketCorsRequest类型具体定义方法如下:

//设置存储桶名称，必须设置

void SetBucket(const Aws::String& value);

//获取存储桶名称

const Aws::String& GetBucket() const;

#### 返回结果说明

* DeleteBucketCorsOutcome:类型

Aws::S3::Model::DeleteBucketCorsOutcome，DeleteBucketCors请求接口 返回参数，定义的方法如下:

//本次请求是否成功

void IsSuccess()

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

#### 示例

#include <cstdio>

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/CreateBucketRequest.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

#include <aws/s3/model/PutBucketCorsRequest.h>

#include <aws/s3/model/GetBucketCorsRequest.h>

#include <aws/s3/model/DeleteBucketCorsRequest.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void DeleteBucketCors(S3Client\* client, const Aws::String& bucketName)

{

DeleteBucketCorsRequest req;

req.SetBucket(bucketName);

client->DeleteBucketCors(req);

}

int main()

{

printf("hello from zos\_sdk\_test!\n");

Aws::SDKOptions options;

Aws::InitAPI(options);

{

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

DeleteBucketCors(&client,"rgwuser01-testbucket03");

}

Aws::ShutdownAPI(options);

return 0;

}

## 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

Aws::S3::Model::PutBucketVersioningOutcome PutBucketVersioning(const Aws::S3::Model::PutBucketVersioningRequest& request) const

#### 参数说明

* request:类型Aws::S3::Model::PutBucketVersioningRequest，PutBucket请求接口参数，具体方法定义如下:

//设置Bucket的名称

void SetBucket(const Aws::String& value)

//设置Bucket的多版本控制功能

void SetVersioningConfiguration(const Aws::S3::Model::VersioningConfiguration& value)

Aws::S3::Model::VersioningConfiguration定义的方法如下:

//设置Bucket的多版本控制功能，BucketVersioningStatus是个enum class，从小到大分别是NOT\_SET, Enabled和Suspended，对应0到2

void SetStatus(const Aws::S3::Model BucketVersioningStatus& value)

#### 返回结果说明

* Aws::S3::Model::PutBucketVersioningOutCome:PutBucketVersioning请求接口的返回参数，具体定义方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/PutBucketVersioningRequest.h>

#include <aws/s3/model/VersioningConfiguration.h>

#include <aws/core/Aws.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void put\_bucket\_version(S3Client &client,

const Aws::String &bucket\_name, int status) {

// 设置请求参数

PutBucketVersioningRequest request;

request.SetBucket(bucket\_name);

VersioningConfiguration cfg;

cfg.SetStatus((BucketVersioningStatus)status);

request.SetVersioningConfiguration(cfg);

// 发出请求

auto outcome = client.PutBucketVersioning(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: put\_bucket\_version: " << "Http code: "<< (int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " <<

err.GetMessage() << "Exception name:" << err.GetExceptionName() << std::endl;

} else {

std::cout << "request success " << std::endl;

}

return;

}

int main(int argc, char\* argv[])

{

if(argc < 3)

{

std::cout << "at least bucket name and object name are needed" <<

std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

int status = atoi(argv[2]);

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

put\_bucket\_version(client, bucket\_name, status);

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model::GetBucketVersioningOutcome GetBucketVersioning(const Aws::S3::Model::GetBucketVersioningRequest& request) const

#### 参数说明

* request:类型Aws::S3::Model::GetBucketVersioningOutcome，GetBucketVersioning请求接口的参数，具体定义方法如下:

//设置Bucket的名称

void SetBucket(const Aws::String& value)

#### 返回结果说明

* Aws::S3::Model::GetBucketVersioningOutcome:GetBucketVersioning请求接口的返回参数，具体定义方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

//获取本次请求返回的结果数据

const Aws::S3::Model::GetBucketVersioningResult& GetResult()

Aws::S3::Model::GetBucketVersioningResult定义的方法如下:

//获取bucket的多版本控制配置，BucketVersioningStatus是个enum class，从小到大分别是NOT\_SET,Enabled和Suspended，分别对应数字0到2

void GetStatus(const Aws::Model::S3::BucketVersioningStatus& value)

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/GetBucketVersioningRequest.h>

#include <aws/core/Aws.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void get\_bucket\_version(S3Client &client,

const Aws::String &bucket\_name) {

// 设置请求参数

GetBucketVersioningRequest request;

request.SetBucket(bucket\_name);

// 发出请求

auto outcome = client.GetBucketVersioning(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: get\_bucket\_version: " << "Http code: "<< (int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " <<

err.GetMessage() << "Exception name:" << err.GetExceptionName() << std::endl;

} else {

auto result = outcome.GetResult();

std::cout<<(int)result.GetStatus()<<std::endl;

std::cout << "request success " << std::endl;

}

return;

}

int main(int argc, char\* argv[])

{

if(argc < 2)

{

std::cout << "at least bucket name and object name are needed" <<

std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

get\_bucket\_version(client, bucket\_name);

Aws::ShutdownAPI(options);

}

## 1.34、Put Bucket Notification Configuration

#### 功能说明

设置桶的指定事件通知。使用此API，您可以替换现有的通知配置。配置定义了希望ZOS发布的事件类型，以及当ZOS检测到指定类型的事件时，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。有关更多信息，请参见为Amazon S3事件配置通知。

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

Aws::S3::Model::PutBucketNotificationConfigurationOutcome S3Client::PutBucketNotificationConfiguration(const PutBucketNotificationConfigurationRequest& request) const

#### 参数说明

* request:类型Aws::S3::Model::PutBucketNotificationConfigurationRequest，创建Bucket Notification请求接口参数，定义的方法如下:

//设置桶名称

void SetBucket(const Aws::String& value)

//设置桶通知信息

void SetNotificationConfiguration(const NotificationConfiguration& value)

Aws::S3::Model::NotificationConfiguration的定义方法如下：

//设置组成NotificationConfiguration的信息，TopicConfiguration是一个class类型，必须设置

void SetTopicConfigurations(const Aws::Vector<TopicConfiguration>& value)

Aws::S3::Model::TopicConfiguration的定义方法如下：

//设置Notification的Id值，必须设置

void SetId(const Aws::String& value)

//设置通知的Topic，必须先创建

void SetTopicArn(Aws::String&& value)

//设置通知的事件，Event是一个枚举class类型，必须设置

void SetEvents(const Aws::Vector<Event>& value)

Aws::S3::Model::Event中，通知支持的事件如下：

Event::s3\_ObjectCreated,

Event::s3\_ObjectCreated\_Put,

Event::s3\_ObjectCreated\_Post,

Event::s3\_ObjectCreated\_Copy,

Event::s3\_ObjectCreated\_CompleteMultipartUpload,

Event::s3\_ObjectRemoved,

Event::s3\_ObjectRemoved\_Delete,

Event::s3\_ObjectRemoved\_DeleteMarkerCreated

#### 返回结果及说明

* PutBucketNotificationConfigurationOutcome:类型Aws::S3::Model::PutBucketNotificationConfigurationOutcome，创建Notification请求接口返回参数，定义的方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/PutBucketNotificationConfigurationRequest.h>

#include <aws/s3/model/NotificationConfiguration.h>

#include <aws/s3/model/TopicConfiguration.h>

#include <aws/s3/model/Event.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void put\_bucket\_notification(S3Client &client, const Aws::String &bucket\_name, const Aws::String &notify\_id, const Aws::String &topic\_arn)

{

Aws::Vector<Event> event\_notify;

event\_notify.push\_back(Event::s3\_ObjectCreated);

event\_notify.push\_back(Event::s3\_ObjectRemoved);

TopicConfiguration topic\_config;

topic\_config.SetId(notify\_id);

topic\_config.SetTopicArn(topic\_arn);

topic\_config.SetEvents(event\_notify);

Vector<TopicConfiguration> topic\_configs;

topic\_configs.push\_back(topic\_config);

NotificationConfiguration notify\_config;

notify\_config.SetTopicConfigurations(topic\_configs);

PutBucketNotificationConfigurationRequest request;

request.SetBucket(bucket\_name);

request.SetNotificationConfiguration(notify\_config);

auto outcome = client.PutBucketNotificationConfiguration(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: " << "Http code: "<<

(int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: "

<< err.GetMessage() << std::endl;

} else {

std::cout << "Successful" << std::endl;

}

return;

}

int main(int argc, char\* argv[])

{

const std::string bucket\_name = "java-zos";

const std::string topic\_arn = "arn:aws:sns:default::Kafka\_topic";

const std::string notify\_id = "notify\_kafka";

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

put\_bucket\_notification(client, bucket\_name, notify\_id, topic\_arn);

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model::GetBucketNotificationConfigurationOutcome S3Client::GetBucketNotificationConfiguration(const GetBucketNotificationConfigurationRequest& request) const

#### 参数说明

* request:类型Aws::S3::Model::GetBucketNotificationConfigurationRequest，获取Bucket Notification请求接口参数，定义的方法如下:

//设置桶名称，必须设置

void SetBucket(const Aws::String& value)

#### 返回结果及说明

* GetBucketNotificationConfigurationOutcome:类型Aws::S3::Model::GetBucketNotificationConfigurationOutcome，获取Notification请求接口返回参数，定义的方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

//获取请求结果

const GetBucketNotificationConfigurationResult& GetResult

//获取本次请求返回的数据

const Aws::S3::Model::NotificationConfiguration& GetRules()

Aws::S3::Model::NotificationConfiguration定义的方法如下：

//获取通知的TopicConfiguration信息

const Aws::Vector<TopicConfiguration>& GetTopicConfigurations() const

Aws::S3::Model::TopicConfiguration定义的方法如下：

//获取通知的Id

const Aws::String& GetId()

//获取通知的TopicArn

const Aws::String& GetTopicArn()

//获取通知的事件

const Aws::Vector<Event>& GetEvents() const

Aws::S3::Model::Event定义的方法如下：

//通过Event类型获取对应的String输出

Aws::String EventMapper::GetNameForEvent(Event value)

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/GetBucketNotificationConfigurationRequest.h>

#include <aws/s3/model/TopicConfiguration.h>

#include <aws/s3/model/Event.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void get\_bucket\_notification(S3Client &client, const Aws::String &bucket\_name)

{

GetBucketNotificationConfigurationRequest request;

request.SetBucket(bucket\_name);

auto outcome = client.GetBucketNotificationConfiguration(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: " << "Http code: "<<

(int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " <<

err.GetMessage() << std::endl;

} else {

auto topicConfig = outcome.GetResult().GetTopicConfigurations();

std::cout << "Successful : " << std::endl;

for (auto it = topicConfig.cbegin();

it != topicConfig.cend(); ++it) {

std::cout << "Id: " << it->GetId() << ", TopicArn: " << it->GetTopicArn() << std::endl;

auto events = it->GetEvents();

for (int i=0; i<events.size(); i++)

std::cout << EventMapper::GetNameForEvent(events.at(i)) << std::endl;

}

}

return;

}

int main(int argc, char\* argv[])

{

const std::string bucket\_name = "java-zos";

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

get\_bucket\_notification(client, bucket\_name);

Aws::ShutdownAPI(options);

}

# 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

Aws::S3::Model::GetObjectOutcome Aws::S3::S3Client::GetObject(const Aws::S3::Model::GetObjectRequest &request) const

#### 参数说明

* request:类型Aws::S3::Model::GetObjectRequest，GetObject请求接口参数，定义方法如下:

//设置Object所在Bucket名称，必须设置

void SetBucket(const Aws::String& value)

//设置Object的名字，必须设置

void SetKey(const Aws::String& value)

// 当对象的Etag和IfMatch中的值一致时返回对象，否则返回412错误

void SetIfMatch(const Aws::String& value)

// 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

void SetIfModifiedSince(const Aws::Utils::DateTime& value)

// 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

void SetIfNoneMatch(const Aws::String& value)

// 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

void SetIfUnmodifiedSince(const Aws::Utils::DateTime& value)

// 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

void SetRange(const Aws::String& value)

// 确认由请求者付费， RequestPayer 为枚举类型，合法值为NOT\_SET 和 requester

void SetRequestPayer(const RequestPayer& value)

// 对象的某个特定版本的版本号，没有该版本则返回404错误void SetVersionId(const Aws::String& value)

#### 返回结果说明

* Aws::S3::Model::GetObjectOutcome: GetObject请求接口返回参数，定义方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

//获取本次请求返回的结果数据

Aws::S3::Model::GetObjectResult && GetResultWithOwnership()

Aws::S3::Model::GetObjectResult 的定义如下:

// 获取返回对象的内容

Aws::IOStream& GetBody()

//获取Object的Metadata，格式为key-value形式

const Aws::Map<Aws::String, Aws::String>& GetMetadata() const

// 获取Object 的创建时间

const Aws::Utils::DateTime& GetLastModified() const

// 获取Object 的ETag

const Aws::String& GetETag() const

// 获取Object 的VersionId

const Aws::String& GetVersionId()

// 获取本次请求者是否付费，RequestCharged 为枚举类型，合法值为request或NOT SET

const RequestCharged& GetRequestCharged()

// 若请求时指定了Range，则返回结果为 “bytes”

const Aws::String& GetAcceptRanges() const

// 指明获取的Object是否是一个DeleteMarker

bool GetDeleteMarker() const

// 获取Object LegalHold 的设置，返回结果为枚举，有效值为NOT\_SET, ON, OFF

const ObjectLockLegalHoldStatus& GetObjectLockLegalHoldStatus() const

// 获取Object Retention 的模式， 有效值为NOT\_SET, GOVERNANCE, COMPLIANCE

const ObjectLockMode& GetObjectLockMode() const

// 获取Retention 过期日期

const Aws::Utils::DateTime& GetObjectLockRetainUntilDate() const

// 获取对象的存储级别

const StorageClass& GetStorageClass() const

#### 示例

#include <iostream>

#include <fstream>

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/GetObjectRequest.h>

#include <aws/core/Aws.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Client;

using namespace std;

void GetObject(S3Client &client, const Aws::String &bucket,

Aws::String objectName, Aws::String fname)

{

// 设置请求

GetObjectRequest request;

request.WithBucket(bucket)

.WithKey(objectName)

.WithRange("bytes=0-100");

auto outcome = client.GetObject(request);

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

cout << "ERROR: GetObject: " << endl

<< "Http code: " << (int)err.GetResponseCode() << endl

<< "Error Type:" << (int)err.GetErrorType() << endl

<< "Error Msg: " << err.GetMessage() << endl

<< "Exception Name: " << err.GetExceptionName() << endl;

}

else

{

auto result = outcome.GetResultWithOwnership();

auto &file = result.GetBody();

auto meta = result.GetMetadata();

// cout << "Etag:" << etag << endl;

char file\_data[255] = {0};

file.getline(file\_data, 254);

cout << "Beginning of file contents: " << endl

<< file\_data << endl;

if (meta.empty())

{

cout << "No Metadata" << endl;

}

else

{

cout << "metadata: " << endl;

for (auto m = meta.begin(); m != meta.end(); ++m)

{

cout << "\tKey: " << m->first << "\tValue: " << m->second

<< endl;

}

}

}

}

int main(int argc, char \*argv[])

{

if (argc != 3)

{

cout << "pls input bucket\_name and object\_name" << endl;

return -1;

}

const Aws::String bucket = argv[1];

const Aws::String fname = argv[2];

Aws::SDKOptions options;

Aws::InitAPI(options);

{

// 设置连接参数

Aws::Client::ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

Aws::Auth::AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

GetObject(client, bucket, fname, fname);

}

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model::HeadObjectOutcome HeadObject(const Aws::S3::Model::HeadObejctRequest&

request) const

#### 参数说明

* request:类型AWS::S3::Model::HeadObejctRequest，HeadObject请求接口参数，定义方法如下:

//设置Object所在Bucket名称，必须设置

void SetBucket(const Aws::String& value)

//设置Object的名字，必须设置

void SetKey(const Aws::String& value)

//设置IfMatch配置，只有当文件的Etag和IfMatch中的值一致时返回文件元数据，否则返回412错误

void SetIfMatch(const Aws::String& value)

//指定文件的修改时间，这个时间之后文件有修改记录才返回文件元数据，不然返回304错误(指定时间之后文件无修改记录)，通过直接传递Unix时间戳就可以使用，单位毫秒，例如SetIfModifiedSince(1619851847000)

void SetIfModifiedSince(const Aws::Utils::DateTime& value)

//设置IfNoneMatch配置，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

void SetIfNoneMatch(const Aws::String& value)

//指定文件的修改时间，这个时间之后文件无修改记录才返回文件元数据，不然返回412错误(指定时间之后文件有修改记录),通过直接传递Unix时间戳就可以使用，单位毫秒，例如SetIfUnmodifiedSince(1619851847000)

void SetIfUnmodifiedSince(const Aws::Utils::DateTime& value)

//指定文件的版本号(多版本)，文件无该版本则返回404错误

void SetVersionId(const Aws::String& value)

#### 返回结果说明

* Aws::S3::Model::HeadObjectOutCome:HeadObject请求接口返回参数，定义方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

//获取本次请求返回的结果数据

const Aws::S3::Model::HeadObjectResult& GetResult()

Aws::S3::Model::HeadObjectResult的定义如下:

//获取文件长度，单位字节

long long GetContentLength() const

//获取文件的Etag

const Aws::String& GetETag() const

//获取文件的存储级别，StorageClass是enum class，从低到高分别是NOT\_SET、STANDARD、REDUCED\_REDUNDANCY、STANDARD\_IA、ONEZONE\_IA、INTELLIGENT\_TIERING、GLACIER、DEEP\_ARCHIVE和OUTPOSTS，分别对应数字0到8

const Aws::S3::Model::StorageClass& GetStorageClass() const

//获取本次请求返回的文件的版本号

const Aws::String& GetVersionId() const

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/HeadObjectRequest.h>

#include <aws/core/Aws.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void head\_object(S3Client &client, const Aws::String &bucket\_name,const

Aws::String &object\_name) {

// 设置请求参数

HeadObjectRequest request;

request.SetBucket(bucket\_name);

request.SetKey(object\_name);

// 发出请求

auto outcome = client.HeadObject(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: Headobject: " << "Http code: "<< (int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " <<

err.GetMessage() << std::endl;

} else {

std::cout << "successful get object metadata: " << std::endl;

}

return;

}

int main(int argc, char\* argv[])

{

if(argc < 3)

{

std::cout << "at least bucket name and object name are needed" <<

std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

const std::string object\_name = argv[2];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

head\_object(client, bucket\_name, object\_name);

Aws::ShutdownAPI(options);

}

## 2.3、Put Object

#### 功能说明

Put Object 请求可以将一个文件（Object）上传至指定 Bucket。

#### 方法原型

Aws::S3::Model::PutObjectOutcome Aws::S3::S3Client::PutObject(const Aws::S3::Model::PutObjectRequest &request) const

#### 参数说明

* request:类型Aws::S3::Model::PutObjectRequest，PutObject请求接口参数，定义方法如下:

//设置Object所在Bucket名称，必须设置

void SetBucket(const Aws::String& value)

//设置Object的名字，必须设置

void SetKey(const Aws::String& value)

// 设置Object的ACL, 详见 [2.6](#_2.6、Put Object ACL)

void SetACL(const ObjectCannedACL& value)

// 其他设置Object ACL相关的函数族，详见[2.6](#_2.6、Put Object ACL)

void SetGrantFullControl(const Aws::String& value)

void SetGrantRead(const Aws::String& value)

void SetGrantReadACP(const Aws::String& value)

void SetGrantWriteACP(const Aws::String& value)

// 设置request body 的 MD5

void SetContentMD5(const Aws::String& value)

// 设置key value形式的Object Metadata

void SetMetadata(const Aws::Map<Aws::String, Aws::String>& value)

// 设置Object LegalHold 详见 [2.11](#_2.11、Put Object Legal Hold)

void SetObjectLockLegalHoldStatus(const ObjectLockLegalHoldStatus& value)

// Object Retention 相关设置，详见 [2.13](#_2.13、Put Object Retention)

void SetObjectLockMode(const ObjectLockMode& value)

void SetObjectLockRetainUntilDate(const Aws::Utils::DateTime& value)

// 确认由请求者付费， RequestPayer 为枚举类型，合法值为NOT\_SET 和 requester

void SetRequestPayer(const RequestPayer& value)

// 设置Object的标签，详见[2.8](#_2.8、Put Object Tagging)

void SetTagging(const Aws::String& value)

// 设置追加模式上传Object，该项置为True则需要同时指定追加位置

void SetAppend(bool value)

// 设置追加位置

void SetAppendPosition(int value)

#### 返回结果说明

* Aws::S3::Model::PutObjectOutcome: PutObject请求接口返回参数，定义方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

//获取本次请求返回的结果数据

Aws::S3::Model::PutObjectResult &GetResult()

Aws::S3::Model::PutObjectResult 的定义如下:

// 追加模式下，获取下一次追加的起始位置

inline int GetAppendPosition() const

//获取文件的Etag

inline const Aws::String& GetETag() const

//获取本次请求返回的文件的版本号

const Aws::String& GetVersionId() const

// 获取本次请求者是否付费，RequestCharged 为枚举类型，合法值为request或NOT SET

inline const RequestCharged& GetRequestCharged()

#### 示例

#include<iostream>

#include<fstream>

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/PutObjectRequest.h>

#include <aws/core/Aws.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Client;

using namespace std;

void PutObject(S3Client &client, const Aws::String &bucket, Aws::String objectName, Aws::String fname)

{

// 设置请求

PutObjectRequest request;

shared\_ptr<Aws::IOStream> input\_data =

Aws::MakeShared<Aws::FStream>(fname.c\_str(),

objectName.c\_str(),

std::ios\_base::in | std::ios\_base::binary);

request.SetBody(input\_data);

Aws::Map<Aws::String, Aws::String> meta;

meta["m1"] = "k1";

request.WithBucket(bucket)

.WithKey(objectName)

.WithACL(ObjectCannedACL::public\_read\_write)

.WithTagging("key1=test\_tag")

.WithMetadata(meta)

.WithObjectLockMode(ObjectLockMode::COMPLIANCE)

.WithStorageClass(StorageClass::STANDARD)

.WithObjectLockRetainUntilDate(Aws::Utils::DateTime(Aws::Utils::DateTime().Now().Millis() + 81920000));

auto outcome = client.PutObject(request);

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

cout << "ERROR: PutObject: " << endl

<< "Http code: " << (int)err.GetResponseCode() << endl

<< "Error Type:" << (int)err.GetErrorType() << endl

<< "Error Msg: " << err.GetMessage() << endl

<< "Exception Name: " << err.GetExceptionName() << endl;

}

else

{

auto result = outcome.GetResult();

auto ID = result.GetVersionId();

Aws::String etag = result.GetETag();

cout << "Etag:" << etag << endl

<< "VersionId: " << ID << endl;

}

}

int main(int argc, char \*argv[])

{

if (argc != 3)

{

cout << "pls input bucket\_name and object\_name" << endl;

return -1;

}

const Aws::String bucket = argv[1];

const Aws::String fname = argv[2];

const Aws::String objName = fname;

Aws::SDKOptions options;

Aws::InitAPI(options);

{

// 设置连接参数

Aws::Client::ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

Aws::Auth::AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

PutObject(client, bucket, fname, objName);

}

Aws::ShutdownAPI(options);

}

追加模式

#include<iostream>

#include<fstream>

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/PutObjectRequest.h>

#include <aws/core/Aws.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Client;

using namespace std;

void AppendObject(S3Client &client, const Aws::String &bucket, Aws::String objectName, Aws::String fname)

{

// 设置请求

PutObjectRequest request;

shared\_ptr<Aws::IOStream> input\_data =

Aws::MakeShared<Aws::FStream>(fname.c\_str(),

objectName.c\_str(),

std::ios\_base::in | std::ios\_base::binary);

shared\_ptr<Aws::IOStream> input\_data2 =

Aws::MakeShared<Aws::FStream>(fname.c\_str(),

objectName.c\_str(),

std::ios\_base::in | std::ios\_base::binary);

Aws::Map<Aws::String, Aws::String> meta;

meta["m1"] = "k1";

request.WithBucket(bucket)

.WithKey(objectName)

.WithACL(ObjectCannedACL::public\_read\_write)

.WithTagging("key1=test\_tag")

.WithMetadata(meta)

.WithAppend(true)

.WithAppendPosition(0);

request.SetBody(input\_data);

auto outcome = client.PutObject(request);

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

cout << "ERROR: PutObject: " << endl

<< "Http code: " << (int)err.GetResponseCode() << endl

<< "Error Type:" << (int)err.GetErrorType() << endl

<< "Error Msg: " << err.GetMessage() << endl

<< "Exception Name: " << err.GetExceptionName() << endl;

}

else

{

auto result = outcome.GetResult();

auto appPos = result.GetAppendPosition();

cout<<"Append Pos: " << appPos << endl;

request.WithAppend(true)

.WithAppendPosition(appPos);

request.SetBody(input\_data2);

outcome = client.PutObject(request);

result = outcome.GetResult();

cout << "Etag:" << result.GetETag() << endl

<< "VersionId: " << result.GetVersionId() << endl;

}

}

int main(int argc, char \*argv[])

{

if (argc != 3)

{

cout << "pls input bucket\_name and object\_name" << endl;

return -1;

}

const Aws::String bucket = argv[1];

const Aws::String fname = argv[2];

Aws::SDKOptions options;

Aws::InitAPI(options);

{

// 设置连接参数

Aws::Client::ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

Aws::Auth::AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

AppendObject(client, bucket, fname, fname);

}

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model::DeleteObjectOutcome Aws::S3::S3Client::DeleteObject(const Aws::S3::Model::DeleteObjectRequest &request) const

#### 参数说明

* request:类型Aws::S3::Model::DeleteObjectRequest ，DeleteObject请求接口参数，定义方法如下:

//设置Object所在Bucket名称，必须设置

void SetBucket(const Aws::String& value)

//设置Object的名字，必须设置

void SetKey(const Aws::String& value)

// 执行操作时是否绕过Governance模式锁的限制

void SetBypassGovernanceRetention(bool value)

// 设置要删除的Object 的 VersionId

void SetVersionId(const Aws::String& value)

#### 返回结果说明

* Aws::S3::Model::DeleteObjectOutcome: DeleteObject请求接口返回参数，定义方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

//获取本次请求返回的结果数据

Aws::S3::Model::DeleteObjectResult &GetResult()

Aws::S3::Model::DeleteObjectResult 的定义如下:

// 若创建了DeleteMarker ，则可获取到DeleteMarker 的VersionId

const Aws::String& GetVersionId() const

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/DeleteObjectRequest.h>

#include <aws/core/Aws.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Client;

using namespace std;

void DeleteObject(S3Client &client, const Aws::String &bucket, const Aws::String &object)

{

// 设置请求

DeleteObjectRequest request;

request.WithBucket(bucket).WithKey(object);

auto outcome = client.DeleteObject(request);

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

cout << "ERROR: DeleteObject: " << endl

<< "Http code: " << (int)err.GetResponseCode() << endl

<< "Error Type:" << (int)err.GetErrorType() << endl

<< "Error Msg: " << err.GetMessage() << endl

<< "Exception Name: " << err.GetExceptionName() << endl;

}

else

{

cout << "object: " << object << " deleted" << endl;

}

}

int main(int argc, char \*argv[])

{

if (argc != 3)

{

cout << "pls input bucket\_name" << endl;

return -1;

}

const Aws::String bucket = argv[1];

const Aws::String object = argv[2];

Aws::SDKOptions options;

Aws::InitAPI(options);

{

// 设置连接参数

Aws::Client::ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

Aws::Auth::AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

DeleteObject(client, bucket, object);

}

Aws::ShutdownAPI(options);

}

## 2.5、Delete Objects

#### 功能说明

Delete Multiple Object 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，ZOS 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

Aws::S3::Model::DeleteObjectsOutcome Aws::S3::S3Client::DeleteObjects(const Aws::S3::Model::DeleteObjectsRequest &request) const

#### 参数说明

* request:类型Aws::S3::Model::DeleteObjectsRequest，DeleteObjects请求接口参数，定义方法如下:

//设置Object所在Bucket名称，必须设置

void SetBucket(const Aws::String& value)

//设置要删除的Object集合，必须设置

void SetDelete(const Aws::S3::Model::Delete& value)

// 执行操作时是否绕过Governance模式锁的限制

void SetBypassGovernanceRetention(bool value)

Aws::S3::Model::Delete 定义方法如下：

//设置要删除的Object标识集合

void SetObjects(const Aws::Vector<Aws::S3::Model::ObjectIdentifier>& value)

//向Object标识集合中添加Object标识

Delete& AddObjects(const Aws::S3::Model::ObjectIdentifier& value)

// 设置是否开启quiet模式，开启quiet模式后，返回结果中只返回报错的Object信息

void SetQuiet(bool value)

Aws::S3::Model::ObjectIdentifier定义方法如下：

//设置Object Key 必须设置

void SetKey(const Aws::String& value)

//设置要删除的Object VersionId

void SetVersionId(const Aws::String& value)

#### 返回结果说明

* Aws::S3::Model::DeleteObjectsOutcome: DeleteObjects请求接口返回参数，定义方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

//获取本次请求返回的结果数据

Aws::S3::Model::DeleteObjectsResult &GetResult()

Aws::S3::Model::DeleteObjectsResult 定义方法如下:

// 若请求时未指定Quiet模式，则可以获取删除成功的Object

const Aws::Vector<Aws::S3::Model::DeletedObject>& GetDeleted()

// 获取删除失败的Object

const Aws::Vector<Aws::S3::Model::Error>& GetErrors() const

Aws::S3::Model::DeletedObject 定义方法如下

// 获取Object Key

const Aws::String& GetKey()

// 获取删除的Object 的 VersionId

const Aws::String& GetVersionId()

Aws::S3::Model::Error 定义方法如下：

// 获取删除失败的错误码

const Aws::String& GetCode() const

// 获取删除失败的Object 的 Key

const Aws::String& GetKey()

// 获取删除失败的Object 的VersionId

const Aws::String& GetVersionId() const

// 获取删除失败的Message

const Aws::String& GetMessage() const

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/DeleteObjectsRequest.h>

#include <aws/core/Aws.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Client;

using namespace std;

void DeleteObject(S3Client &client, const Aws::String &bucket, const Aws::String &object\_prefix)

{

// 设置请求

DeleteObjectsRequest request;

Delete delete\_objects;

Aws::Vector<ObjectIdentifier> idnetfiers;

for (int i = 0; i <= 10; i++)

{

ObjectIdentifier ident;

ident.SetKey(object\_prefix + to\_string(i));

idnetfiers.push\_back(ident);

}

delete\_objects.SetObjects(idnetfiers);

delete\_objects.SetQuiet(false);

request.SetBucket(bucket);

request.SetDelete(delete\_objects);

auto outcome = client.DeleteObjects(request);

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

cout << "ERROR: DeleteObjects: " << endl

<< "Http code: " << (int)err.GetResponseCode() << endl

<< "Error Type:" << (int)err.GetErrorType() << endl

<< "Error Msg: " << err.GetMessage() << endl

<< "Exception Name: " << err.GetExceptionName() << endl;

}

else

{

auto result = outcome.GetResult();

auto deleted = result.GetDeleted();

auto errors = result.GetErrors();

cout << "deleted Objects" << endl;

for(auto iter = deleted.begin(); iter!= deleted.end(); iter++)

{

cout << iter->GetKey() << " deleted" << endl;

}

if (!errors.empty())

{

cout << "Objects failed to delete:" << endl;

for (auto iter = errors.begin(); iter != errors.end(); iter++)

{

cout << iter->GetKey() << " " << iter->GetMessage() << " Code: " << iter->GetCode() << endl;

}

}

}

}

int main(int argc, char \*argv[])

{

if (argc != 3)

{

cout << "pls input bucket\_name" << endl;

return -1;

}

const Aws::String bucket = argv[1];

const Aws::String object = argv[2];

Aws::SDKOptions options;

Aws::InitAPI(options);

{

// 设置连接参数

Aws::Client::ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

Aws::Auth::AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

DeleteObject(client, bucket, object);

}

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model::PutObjectAclOutcome S3Client::PutObjectAcl(const Aws::S3::Model::PutObjectAclRequest& request) const

#### 参数说明

* request: Aws::S3::Model::PutObjectAclRequest类型，该类定义的方法有:

//指定桶名称

void SetBucket(const Aws::String& value)

// 指定对象名称

void SetKey(const Aws::String& value)

// 指定对象版本号，历史版本需要指定

void SetVersionId(const Aws::String& value)

// 以下三组参数，对应三种方式，不能同时使用

// 设置ACL，ObjectCannedACL取值范围为

// private\_,

// public\_read,

// public\_read\_write,

// authenticated\_read

void SetACL(const ObjectCannedACL& value)

// 设置AccessControlPolicy

void SetAccessControlPolicy(const AccessControlPolicy& value)

// 设置Grant\*,字符串格式为key=value，"id=xxxx, emailAdddress=xxxx, uri=xxxx"

// 可以组合多个key=value

// uri取值为http://acs.amazonaws.com/groups/global/AllUsers

// 或者http://acs.amazonaws.com/groups/global/AuthenticatedUsers

void SetGrantFullControl(const Aws::String& value)

void SetGrantRead(const Aws::String& value)

void SetGrantReadACP(const Aws::String& value)

void SetGrantWrite(const Aws::String& value)

void SetGrantWriteACP(const Aws::String& value)

AccessControlPolicy类型，该类定义的方法有:

//指定对象所有者

void SetOwner(const Owner& value)

// 设置授权列表

void SetGrants(const Aws::Vector<Grant>& value)

Owner类型，该类定义的方法有:

//指定对象所有者用户ID

void SetID(const Aws::String& value)

Grant类型，该类定义的方法有:

//指定被授权用户

void SetGrantee(const Grantee& value)

// 指定被授权权限，取值范围为

// FULL\_CONTROL,

// WRITE,

// WRITE\_ACP,

// READ,

// READ\_ACP

void SetPermission(const Permission& value)

Grantee类型，该类定义的方法有:

//指定被授权用户类型，取值范围

// CanonicalUser,

// AmazonCustomerByEmail,

// Group

void SetType(const Type& value)

// 指定被授权用户ID，如果用户类型为CanonicalUser，需要指定该字段

void SetID(const Aws::String& value)

// 指定被授权用户邮箱，如果用户类型为AmazonCustomerByEmail，需要指定该字段

void SetEmailAddress(Aws::String&& value)

// 指定被授权组，如果用户类型为Group，需要指定该字段

// 取值为http://acs.amazonaws.com/groups/global/AllUsers

// 或者http://acs.amazonaws.com/groups/global/AuthenticatedUsers

void SetURI(const Aws::String& value)

#### 返回结果说明

* 返回结果类型为Aws::S3::Model::PutObjectAclOutcome，该类型定义的方法有:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/PutObjectAclRequest.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void put\_object\_acl(S3Client &client, const Aws::String &bucket\_name, const Aws::String &key, const Aws::String &version\_id)

{

PutObjectAclRequest request;

request.SetBucket(bucket\_name);

request.SetKey(key);

if (!version\_id.empty()) {

request.SetVersionId(version\_id);

}

// request.SetACL(ObjectCannedACL::public\_read);

Owner owner;

owner.SetID("test-1");

Grantee grantee;

// grantee.SetType(Type::CanonicalUser);

grantee.SetType(Type::AmazonCustomerByEmail);

// grantee.SetID("test-3");

// grantee.SetURI();

grantee.SetEmailAddress("abc@abc.com");

Grant grant;

grant.SetGrantee(grantee);

grant.SetPermission(Permission::FULL\_CONTROL);

Aws::Vector<Grant> grants;

grants.push\_back(grant);

AccessControlPolicy policy;

policy.SetOwner(owner);

policy.SetGrants(grants);

request.SetAccessControlPolicy(policy);

// request.SetGrantRead(

// "id=test-2,uri=http://acs.amazonaws.com/groups/global/AllUsers"

// );

auto outcome = client.PutObjectAcl(request);

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: " << "Http code: "<<

(int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " <<

err.GetMessage() << std::endl;

} else {

std::cout << "successful" << endl;

}

return;

}

int main(int argc, char\* argv[])

{

if(argc != 4)

{

std::cout << "pls input bucket\_name" << std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

const std::string key = argv[2];

const std::string version\_id = argv[3];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

put\_object\_acl(client, bucket\_name, key, version\_id);

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model::GetObjectAclOutcome S3Client::GetObjectAcl(const Aws::S3::Model::GetObjectAclRequest& request) const

#### 参数说明

* request: Aws::S3::Model::GetObjectAclRequest类型，该类定义的方法有:

//指定bucket

void SetBucket(const Aws::String& value)

// 指定对象名

void SetKey(const Aws::String& value)

// 指定对象版本号，历史版本需要指定

void SetVersionId(const Aws::String& value)

#### 返回结果说明

* 返回结果类型为Aws::S3::Model::GetObjectAclOutcome，该类型定义的方法有:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

// 获取bucket acl result接口

const GetObjectAclResult& GetResult() const

GetObjectAclResult，该类型定义的方法有:

// 获取对象所有者

const Owner& GetOwner() const

// 获取授权列表

const Aws::Vector<Grant>& GetGrants() const

Owner，该类型定义的方法有:

// 获取对象所有者display name

const Aws::String& GetDisplayName() const

// 获取对象所有者用户ID

const Aws::String& GetID() const

Grant，该类型定义的方法有:

// 获取被授权用户

const Grantee& GetGrantee() const

// 获取被授权权限

const Permission& GetPermission()

Grantee，该类型定义的方法有:

// 获取被授权用户类型

const Type& GetType() const

// 获取被授权用户ID

const Aws::String& GetID() const

// 获取被授权用户display name

const Aws::String& GetDisplayName() const

// 获取被授权用户邮箱

const Aws::String& GetEmailAddress() const

// 获取被授权组uri

const Aws::String& GetURI() const

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/GetObjectAclRequest.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::S3::Model::TypeMapper;

using namespace Aws::S3::Model::PermissionMapper;

using namespace Aws::Auth;

using namespace std;

void get\_object\_acl(S3Client &client, const Aws::String &bucket\_name, const Aws::String &key, const Aws::String &version\_id)

{

GetObjectAclRequest request;

request.SetBucket(bucket\_name);

request.SetKey(key);

if (!version\_id.empty()) {

request.SetVersionId(version\_id);

}

auto outcome = client.GetObjectAcl(request);

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: " << "Http code: "<<

(int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " <<

err.GetMessage() << std::endl;

} else {

std::cout << "successful : " << endl;

auto result = outcome.GetResult();

auto owner = result.GetOwner();

std::cout << "Owner ID = " << owner.GetID() << endl;

std::cout << "Owner DisplayName = " << owner.GetDisplayName() <<

endl;

auto grants = result.GetGrants();

for (auto it = grants.cbegin(); it != grants.cend(); ++it) {

auto grantee = it->GetGrantee();

auto type = grantee.GetType();

std::cout << "Type = " << GetNameForType(type) << endl;

if (type == Type::Group) {

auto uri = grantee.GetURI();

std::cout << "URI = " << uri << endl;

} else {

auto id = grantee.GetID();

auto display\_name = grantee.GetDisplayName();

auto email = grantee.GetEmailAddress();

std::cout << "ID = " << id << endl;

std::cout << "DisplayName = " << display\_name << endl;

std::cout << "Email = " << email << endl;

}

auto permission = it->GetPermission();

std::cout << "Permission = " << GetNameForPermission(permission) << endl;

}

}

return;

}

int main(int argc, char\* argv[])

{

if(argc != 4)

{

std::cout << "pls input bucket\_name" << std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

const std::string key = argv[2];

const std::string version\_id = argv[3];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

get\_object\_acl(client, bucket\_name, key, version\_id);

Aws::ShutdownAPI(options);

}

## 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，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

Aws::S3::Model::PutObjectTaggingOutcome S3Client::PutObjectTagging(const PutObjectTaggingRequest& request) const

#### 参数说明

* request: Aws::S3::Model::PutObjectTaggingRequest类型，该类定义的方法有:

// 指定桶名称

void SetBucket(const Aws::String& value)

// 指定对象名称

void SetKey(const Aws::String& value)

// 指定对象版本号，历史版本需要指定

void SetVersionId(const Aws::String& value)

// 指定标签集

void SetTagging(const Tagging& value)

Tagging类型，该类定义的方法有:

// 指定指定标签列表

void SetTagSet(const Aws::Vector<Tag>& value)

Tag类型，该类定义的方法有:

// 指定标签key

void SetKey(const Aws::String& value)

// 指定标签value

void SetValue(const Aws::String& value)

#### 返回结果说明

* 返回结果类型为Aws::S3::Model::PutObjectTaggingOutcome，该类型定义的方法有:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/PutObjectTaggingRequest.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void put\_object\_tagging(S3Client &client,

const Aws::String &bucket\_name,

const Aws::String &object\_key,

const Aws::String &version\_id)

{

PutObjectTaggingRequest request;

request.SetBucket(bucket\_name);

request.SetKey(object\_key);

request.SetVersionId(version\_id);

Tag tag;

tag.SetKey("key1");

tag.SetValue("val1");

Aws::Vector<Tag> tags;

tags.push\_back(tag);

Tagging tagging;

tagging.SetTagSet(tags);

request.SetTagging(tagging);

auto outcome = client.PutObjectTagging(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: " << "Http code: "<<

(int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error

Msg: " << err.GetMessage() << std::endl;

} else {

std::cout << "successful : " << endl;

}

return;

}

int main(int argc, char\* argv[])

{

const std::string bucket\_name = "java-zos";

const std::string object\_name = "object-002";

const std::string version\_id = "yGuRzcUi-2RxqngoTLzdt1jLyKR-p1w";

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTPS;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

put\_object\_tagging(client, bucket\_name, object\_name, version\_id);

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model::GetObjectTaggingOutcome S3Client::GetObjectTagging(const GetObjectTaggingRequest& request) const

#### 参数说明

* request: Aws::S3::Model::GetObjectTaggingRequest类型，该类定义的方法有:

// 指定桶名称

void SetBucket(const Aws::String& value)

// 指定对象名称

void SetKey(const Aws::String& value)

// 指定对象版本号，历史版本需要指定

void SetVersionId(const Aws::String& value)

#### 返回结果说明

* 返回结果类型为Aws::S3::Model::GetObjectTaggingOutcome，该类型定义的方法有:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

//获取应答结果

const GetObjectTaggingResult& GetResult() const

GetBucketTaggingResult，该类型定义的方法有:

// 获取标签列表

const Aws::Vector<Tag>& GetTagSet() const

// 获取对象版本号

const Aws::String& GetVersionId() const

Tag，该类型定义的方法有:

// 获取标签key

const Aws::String& GetKey() const

// 获取标签value

const Aws::String& GetValue() const

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/GetObjectTaggingRequest.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void get\_object\_tagging(S3Client &client,

const Aws::String &bucket\_name,

const Aws::String &object\_key,

const Aws::String &version\_id)

{

GetObjectTaggingRequest request;

request.SetBucket(bucket\_name);

request.SetKey(object\_key);

request.SetVersionId(version\_id);

auto outcome = client.GetObjectTagging(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: " << "Http code: "<<

(int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType()

<< " Error Msg: " << err.GetMessage() << std::endl;

} else {

std::cout << "successful : " << endl;

auto result = outcome.GetResult();

auto tags = result.GetTagSet();

for (auto it = tags.cbegin(); it != tags.cend(); ++it) {

std::cout << it->GetKey() << " = " << it->GetValue() << endl;

}

}

return;

}

int main(int argc, char\* argv[])

{

const std::string bucket\_name = "java-zos";

const std::string object\_name = "object-002";

const std::string version\_id = "yGuRzcUi-2RxqngoTLzdt1jLyKR-p1w";

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTPS;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

get\_object\_tagging(client, bucket\_name, object\_name, version\_id);

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model::DeleteObjectTaggingOutcome S3Client::DeleteObjectTagging(

const DeleteObjectTaggingRequest& request) const

#### 参数说明

* request: Aws::S3::Model::DeleteObjectTaggingRequest类型，该类定义的方法有:

// 指定桶名称

void SetBucket(const Aws::String& value)

// 指定对象名称

void SetKey(const Aws::String& value)

// 指定对象版本号，历史版本需要指定

void SetVersionId(const Aws::String& value)

#### 返回结果说明

* 返回结果类型为Aws::S3::Model::DeleteObjectTaggingOutcome，该类型定义的方法有:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/DeleteObjectTaggingRequest.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void delete\_object\_tagging(S3Client &client, const Aws::String &bucket\_name, const Aws::String &object\_key, const Aws::String &version\_id)

{

DeleteObjectTaggingRequest request;

request.SetBucket(bucket\_name);

request.SetKey(object\_key);

request.SetVersionId(version\_id);

auto outcome = client.DeleteObjectTagging(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: " << "Http code: "<<

(int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() <<

" Error Msg: " << err.GetMessage() << std::endl;

} else {

std::cout << "successful : " << endl;

}

return;

}

int main(int argc, char\* argv[])

{

const std::string bucket\_name = "java-zos";

const std::string object\_name = "object-002";

const std::string version\_id = "yGuRzcUi-2RxqngoTLzdt1jLyKR-p1w";

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTPS;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

delete\_object\_tagging(client, bucket\_name, object\_name, version\_id);

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model::PutObjectLegalHold Aws::S3::S3Client::PutObjectLegalHold(const Aws::S3::Model::PutObjectLegalHoldRequest& request) const

#### 参数说明

* request:类型Aws::S3::Model::PutObjectLegalHoldRequest，设置对象依法保留请求接口参数，定义的方法如下:

//设置Bucket名称，必须设置，Aws::String可通过标准std::string赋值

void SetBucket(const Aws::String& value)

//设置对象名称，必须设置

void SetKey(const Aws::String& value)

//设置对象版本

void SetVersionId(const Aws::String& value)

//设置依法保留配置

void SetLegalHold(const ObjectLockLegalHold& value)

类型Aws::S3::Model::ObjectLockLegalHold，设置对象依法保留接口参数，定义的方法如下:

//设置依法保留状态，ObjectLockLegalHoldStatus是个enum class类型，从小到大分别是NOT\_SET，ON，OFF分别对应数字0到2

void SetStatus(ObjectLockLegalHoldStatus&& value)

#### 返回结果说明

* PutObjectLegalHoldOutcome:类型Aws::S3::Model:: PutObjectLegalHoldOutcome，启用对象依法保留功能接口返回参数，定义的方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/PutBucketLoggingRequest.h>

#include <aws/core/Aws.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void put\_bucket\_logging(S3Client& client,

const Aws::String& bucket\_name) {

// 设置请求参数

PutBucketLoggingRequest request;

request.SetBucket(bucket\_name);

Grantee grantee;

grantee.SetType(Type::CanonicalUser);

grantee.SetDisplayName("Second User");

grantee.SetID("s3");

TargetGrant target\_grant;

target\_grant.SetPermission(BucketLogsPermission::FULL\_CONTROL);

target\_grant.SetGrantee(grantee);

Aws::Vector<TargetGrant> v\_target\_bucket;

v\_target\_bucket.push\_back(target\_grant);

LoggingEnabled logging\_enable;

logging\_enable.SetTargetBucket("target\_bucket\_sdk");

logging\_enable.SetTargetPrefix("log/");

logging\_enable.SetTargetGrants(v\_target\_bucket);

BucketLoggingStatus bucket\_logging\_status;

bucket\_logging\_status.SetLoggingEnabled(logging\_enable);

request.SetBucketLoggingStatus(bucket\_logging\_status);

// 发出请求

PutBucketLoggingOutcome outcome = client.PutBucketLogging(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: PutBucketLogging: " << "Http code: " << (int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() <<

" Error Msg: " << err.GetMessage() << std::endl;

}

else {

std::cout << "successful put bucket logging: " << bucket\_name <<

"\n";

}

return;

}

int main(int argc, char\* argv[])

{

if (argc != 2)

{

std::cout << "pls input bucket\_name" << std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

put\_bucket\_logging(client, bucket\_name);

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model::GetObjectLegalHold Aws::S3::S3Client::GetObjectLegalHold(const Aws::S3::Model::GetObjectLegalHoldRequest& request) const

#### 参数说明

* request:类型Aws::S3::Model::GetObjectLegalHoldRequest，获取对象依法保留请求接口参数，定义的方法如下:

//设置Bucket名称，必须设置，Aws::String可通过标准std::string赋值

void SetBucket(const Aws::String& value)

//设置对象名称，必须设置

void SetKey(const Aws::String& value)

//设置对象版本

void SetVersionId(const Aws::String& value)

//设置依法保留配置

void SetLegalHold(const ObjectLockLegalHold& value)

#### 返回结果说明

* GetObjectLegalHoldOutCome:类型Aws::S3::Model:: GetObjectLegalHoldOutCome，获取对象依法保留配置接口返回参数，定义的方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

//获取对象依法保留配置结果

GetObjectLegalHoldResult& GetResult()

类型Aws::S3::Model::GetObjectLegalHoldResult，获取对象依法保留结果接口参数，定义的方法如下:

//获取对象锁定依法保留

const ObjectLockLegalHold& GetLegalHold() const

类型Aws::S3::Model::ObjectLockLegalHold，获取对象依法保留配置接口参数，定义的方法如下:

//获取对象锁定依法保留，ObjectLockLegalHoldStatus是个enum class类型，从小到大分别是NOT\_SET，ON，OFF分别对应数字0到2

const ObjectLockLegalHoldStatus& GetStatus() const

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/GetObjectLegalHoldRequest.h>

#include <aws/core/Aws.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

const char\* legal\_hold\_status[] = { "NOT\_SET", "ON", "OFF" };

void get\_object\_legal\_hold(S3Client& client, const Aws::String& bucket\_name, const Aws::String& object\_name, const Aws::String& object\_version) {

// 设置请求参数

GetObjectLegalHoldRequest request;

request.SetBucket(bucket\_name);

request.SetKey(object\_name);

request.SetVersionId(object\_version);

// 发出请求

GetObjectLegalHoldOutcome outcome = client.GetObjectLegalHold(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: GetObjectLegalHold: " << "Http code: " << (int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " <<

err.GetMessage() << std::endl;

}

else {

std::cout << "successful get object legal hold: " << bucket\_name

<< "\n";

GetObjectLegalHoldResult result = outcome.GetResult();

ObjectLockLegalHold legalhold = result.GetLegalHold();

ObjectLockLegalHoldStatus status = legalhold.GetStatus();

cout << "GetStatus: " << legal\_hold\_status[int(status)] << endl;

}

return;

}

int main(int argc, char\* argv[])

{

if (argc != 4)

{

std::cout << "pls input bucket\_name" << std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

const std::string object\_name = argv[2];

const std::string object\_version = argv[3];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

get\_object\_legal\_hold(client, bucket\_name, object\_name, object\_version);

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model::PutObjectRetention Aws::S3::S3Client::PutObjectRetention (const Aws::S3::Model::PutObjectRetentionRequest& request) const

#### 参数说明

* request:类型Aws::S3::Model::PutObjectRetentionRequest，设置对象保留期限请求接口参数，定义的方法如下:

//设置Bucket名称，必须设置，Aws::String可通过标准std::string赋值

void SetBucket(const Aws::String& value)

//设置对象名称，必须设置

void SetKey(const Aws::String& value)

//设置对象版本

void SetVersionId(const Aws::String& value)

//设置保留期限配置

void SetRetention(const ObjectLockRetention& value)

//设置绕过保留期限限制

void SetBypassGovernanceRetention(bool value)

类型Aws::S3::Model::ObjectLockRetention，设置对象保留期限接口参数，定义的方法如下:

//设置保留期限模式，ObjectLockRetentionMode是个enum class类型，从小到大分别是NOT\_SET，GOVERNANCE，COMPLIANCE分别对应数字0到2

void SetMode(ObjectLockRetentionMode&& value)

//设置保留到期日期，DateTime参数赋值方式如"2021-06-13T16:45:19Z"

void SetRetainUntilDate(const Aws::Utils::DateTime& value)

#### 返回结果说明

* PutObjectRetentionOutcome:类型Aws::S3::Model:: PutObjectRetentionOutcome，启用对象保留期限功能接口返回参数，定义的方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/PutObjectRetentionRequest.h>

#include <aws/core/Aws.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void put\_object\_retention(S3Client& client, const Aws::String& bucket\_name, const Aws::String& object\_name, const Aws::String& object\_version) {

// 设置请求参数

PutObjectRetentionRequest request;

request.SetBucket(bucket\_name);

request.SetKey(object\_name);

request.SetVersionId(object\_version);

Aws::Utils::DateTime date\_time;

//Aws::String date = date\_time.ToLocalTimeString(Aws::Utils::DateFormat::ISO\_8601);

date\_time = "2021-06-13T16:45:19Z";

ObjectLockRetention retention;

retention.SetMode(ObjectLockRetentionMode::COMPLIANCE);

retention.SetRetainUntilDate(date\_time);

request.SetRetention(retention);

request.SetBypassGovernanceRetention(true);

// 发出请求

PutObjectRetentionOutcome outcome = client.PutObjectRetention(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: PutObjectRetention: " << "Http code: " << (int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " <<

err.GetMessage() << std::endl;

}

else {

std::cout << "successful put object retention: " << bucket\_name

<< "\n";

}

return;

}

int main(int argc, char\* argv[])

{

if (argc != 4)

{

std::cout << "pls input bucket\_name" << std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

const std::string object\_name = argv[2];

const std::string object\_version = argv[3];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

put\_object\_retention(client, bucket\_name, object\_name, object\_version);

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model::GetObjectRetention Aws::S3::S3Client::GetObjectRetention

(const Aws::S3::Model::GetObjectRetentionRequest& request) const

#### 参数说明

* request:类型Aws::S3::Model::GetObjectRetentionRequest，获取对象保留期限请求接口参数，定义的方法如下:

//设置Bucket名称，必须设置，Aws::String可通过标准std::string赋值

void SetBucket(const Aws::String& value)

//设置对象名称，必须设置

void SetKey(const Aws::String& value)

//设置对象版本

void SetVersionId(const Aws::String& value)

#### 返回结果说明

* GetObjectRetentionOutCome:类型Aws::S3::Model:: GetObjectRetentionOutCome，获取对象保留期限配置接口返回参数，定义的方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

//获取对象保留期限配置结果

GetObjectRetentionResult& GetResult()

类型Aws::S3::Model::GetObjectRetentionResult，获取对象保留期限结果接口参数，定义的方法如下:

//获取对象锁定依法保留

const ObjectLockRetention& GetRetention() const

类型Aws::S3::Model::ObjectLockRetention，获取对象保留期限配置接口参数，定义的方法如下:

//获取对象保留期限模式，ObjectLockRetentionMode是个enum class类型，从小到大分别是NOT\_SET，GOVERNANCE，COMPLIANCE分别对应数字0到2

const ObjectLockRetentionMode& GetMode() const

//获取对象保留到期日期

const Aws::Utils::DateTime& GetRetainUntilDate() const

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/GetObjectRetentionRequest.h>

#include <aws/core/Aws.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

const char\* object\_lock\_retention[] = { "NOT\_SET", "GOVERNANCE", "COMPLIANCE" };

void get\_object\_retention(S3Client& client, const Aws::String& bucket\_name, const Aws::String& object\_name, const Aws::String& object\_version) {

// 设置请求参数

GetObjectRetentionRequest request;

request.SetBucket(bucket\_name);

request.SetKey(object\_name);

request.SetVersionId(object\_version);

// 发出请求

GetObjectRetentionOutcome outcome = client.GetObjectRetention(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: GetObjectRetention: " << "Http code: " << (int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " << err.GetMessage() << std::endl;

}

else {

std::cout << "successful get object retention: " << bucket\_name << "\n";

GetObjectRetentionResult result = outcome.GetResult();

ObjectLockRetention retention = result.GetRetention();

ObjectLockRetentionMode mode = retention.GetMode();

cout << "GetMode: " << object\_lock\_retention[int(mode)] << endl;

Aws::Utils::DateTime date\_time = retention.GetRetainUntilDate();

cout << "GetRetainUntilDate: " << date\_time.GetYear(true) << "-"

<< int(date\_time.GetMonth(true)) + 1 << "-" << date\_time.GetDay(true)

<< "T" << date\_time.GetHour(true) << ":" << date\_time.GetMinute(true)

<< ":" << date\_time.GetSecond(true) << "Z" << endl;

}

return;

}

int main(int argc, char\* argv[])

{

if (argc != 4)

{

std::cout << "pls input bucket\_name" << std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

const std::string object\_name = argv[2];

const std::string object\_version = argv[3];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("C2WDY6K468DVOCRJVCX1", "134u4nkxkQdkOS4V1otUDd5gZI3V4lyHI5a4w85S"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

get\_object\_retention(client, bucket\_name, object\_name, object\_version);

Aws::ShutdownAPI(options);

}

## 2.15、Genertae Presigned Url

#### 功能说明

Generate Presigned Url请求生成一个临时的预签名的Url，没有权限访问集群的用户可以通过该Url访问集群，包括上传、下载文件等等。

#### 方法原型

Aws::String GeneratePresignedUrl(const Aws::String& bucket, const Aws::String& key, Aws::Http::HttpMethod method, long long expirationInSeconds = MAX\_EXPIRATION\_SECONDS);

#### 参数说明

* bucket:Url要访问的Bucket
* key:Url要访问的位于bucket中的文件名
* method:类型Aws::Http::HttpMethod是个enum class，从0到5分别对应HTTP\_GET、HTTP\_POST、HTTP\_DELETE、HTTP\_PUT、HTTP\_HEAD和HTTP\_PATCH
* expirationInSeconds:Url的超时时间，超出该时间Url无效，单位秒，默认时间一周

#### 返回结果说明

* Aws::String:返回Url的字符串

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/core/Aws.h>

#include <aws/core/http/HttpTypes.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

#include <sys/stat.h>

#include <iostream>

#include <fstream>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

int main(int argc, char\* argv[])

{

if(argc <= 2)

{

std::cout << "at least bucket name and key name are needed" << std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

const std::string key\_name = argv[2];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

auto outcome = client.GeneratePresignedUrl(bucket\_name, key\_name, Aws::Http::HttpMethod::HTTP\_PUT);

std::cout<< outcome << std::endl;

Aws::ShutdownAPI(options);

}

## 2.16、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

Aws::S3::Model::CopyObjectOutcome Aws::S3::S3Client::CopyObject(const Aws::S3::Model::CopyObjectRequest &request) const

#### 参数说明

* request:类型Aws::S3::Model::CopyObjectRequest ，CopyObject请求接口参数，定义方法如下:

//设置Object所在Bucket名称，必须设置

void SetBucket(const Aws::String& value)

//设置Object的名字，必须设置

void SetKey(const Aws::String& value)

// 设置copy的源，格式如 Bucket/objectkey 必须设置

void SetCopySource(const char\* value)

// 仅当指定的Etag和Copy Source指定的object的Etag匹配时才复制

void SetCopySourceIfMatch(const Aws::String& value)

// 仅当指定的Etag和Copy Source指定的object的Etag不匹配时才复制

void SetCopySourceIfNoneMatch(const Aws::String& value)

// 仅当CopySource在指定时间后更新过才复制

void SetCopySourceIfModifiedSince(const Aws::Utils::DateTime& value)

// 仅当CopySource在指定时间后未更新过才复制

void SetCopySourceIfUnmodifiedSince(const Aws::Utils::DateTime& value)

// ACL相关设置，详见[2.6](#_2.6、Put Object ACL)

void SetGrantFullControl(const Aws::String& value)

void SetGrantRead(const Aws::String& value)

void SetGrantReadACP(const Aws::String& value)

void SetGrantWriteACP(const Aws::String& value)

// 设置copy对象的Metadata

void SetMetadata(const Aws::Map<Aws::String, Aws::String>& value)

// Metadata复制选项，'COPY' 复制源的Metadata或 'REPLACE'使用新指定的Metadata覆盖

void SetMetadataDirective(const MetadataDirective& value)

// 设置复制对象的Tag

void SetTagging(const Aws::String& value)

// Tagging 复制选项 'COPY' 复制源的Tag或 'REPLACE'使用新指定的Tag覆盖

void SetTaggingDirective(const TaggingDirective& value)

// 设置复制的Object 的LegalHold 开关， 详见[2.11](#_2.11、Put Object Legal Hold)

void SetObjectLockLegalHoldStatus(const ObjectLockLegalHoldStatus& value)

// 设置复制的Object 的 Retention 模式 详见 [2.13](#_2.13、Put Object Retention)

void SetObjectLockMode(const ObjectLockMode& value)

// 设置复制的Object 的 Retention 保留过期时间 详见 [2.13](#_2.13、Put Object Retention)

void SetObjectLockRetainUntilDate(const Aws::Utils::DateTime& value)

#### 返回结果说明

* Aws::S3::Model::CopyObjectOutcome: CopyObject请求接口返回参数，定义方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

//获取本次请求返回的结果数据

Aws::S3::Model::CopyObjectResult &GetResult()

Aws::S3::Model::CopyObjectResult 定义方法如下:

// 获取复制的详细结果

const CopyObjectResultDetails& GetCopyObjectResultDetails() const

// 获取Source 的VersionId

const Aws::String& GetCopySourceVersionId() const

// 获取新创建的Object 的VersionId

const Aws::String& GetVersionId() const

Aws::S3::Model::CopyObjectResultDetails 定义方法如下：

// 获取的复制的Object 的Etag

const Aws::String& GetETag() const

// 复制Object 的创建时间

const Aws::Utils::DateTime& GetLastModified() const

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/CopyObjectRequest.h>

#include <aws/core/Aws.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Client;

using namespace std;

void CopyObject(S3Client &client, const Aws::String &bucket, const Aws::String &key, const Aws::String source)

{

// 设置请求

CopyObjectRequest request;

request.WithACL(ObjectCannedACL::public\_read\_write)

.WithBucket(bucket)

.WithKey(key)

.WithCopySource(source);

auto outcome = client.CopyObject(request);

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

cout << "ERROR: CopyObject: " << endl

<< "Http code: " << (int)err.GetResponseCode() << endl

<< "Error Type:" << (int)err.GetErrorType() << endl

<< "Error Msg: " << err.GetMessage() << endl

<< "Exception Name: " << err.GetExceptionName() << endl;

}

else

{

auto result = outcome.GetResult();

auto copyResultDetail = result.GetCopyObjectResultDetails();

auto eTag = copyResultDetail.GetETag();

cout << "Object Copyed" << eTag << endl;

}

}

int main(int argc, char \*argv[])

{

if (argc != 4)

{

cout << "pls input bucket\_name" << endl;

return -1;

}

const Aws::String bucket = argv[1];

const Aws::String object = argv[2];

const Aws::String source = argv[3];

Aws::SDKOptions options;

Aws::InitAPI(options);

{

// 设置连接参数

Aws::Client::ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

Aws::Auth::AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

CopyObject(client, bucket, object, source);

}

Aws::ShutdownAPI(options);

}

## 2.17、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

Aws::S3::Model::UploadPartCopyOutcome UploadPartCopy(const Aws::S3::Model::UploadPartCopyRequest& request)

#### 参数说明

* request:类型UploadPartCopyRequest ，UploadPartCopy请求接口参数，定义方法如下:

// 设置Copy目的Bucket名称

void SetBucket(Aws::String&& value)

// 设置Copy目的Object名称

void SetKey(const Aws::String& value)

// 设置copy的源，格式如 Bucket/objectkey 必须设置

void SetCopySource(const Aws::String& value)

// 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

void SetCopySourceRange(const Aws::String& value)

// 设置当前Part的编号

void SetPartNumber(int value)

// 设置MultiUpload ID，由3.1 initMultiPartUpload返回

void SetUploadId(const Aws::String& value)

// 当对象的Etag和指定的值一致时才复制对象

void SetCopySourceIfMatch(const Aws::String& value)

// 只有指定时间之后有修改记录的对象才复制对象

void SetCopySourceIfModifiedSince(const Aws::Utils::DateTime& value)

// 当对象的Etag和指定的值不一致时才复制对象

void SetCopySourceIfNoneMatch(const Aws::String& value)

// 只有指定时间之后没有修改记录的对象才复制对象

void SetCopySourceIfUnmodifiedSince(const Aws::Utils::DateTime& value)

#### 返回结果说明

* Aws::S3::Model::UploadPartCopyOutcome，UploadPartCopy返回结果，定义方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

//获取本次请求返回的结果数据

Aws::S3::Model::UploadPartCopyResult& GetResult()

Aws::S3::Model::UploadPartCopyResult 定义方法如下:

// 获取源Object的VersionId

const Aws::String& GetCopySourceVersionId() const

//获取Copy结果

const CopyPartResult& GetCopyPartResult() const

Aws::S3::Model::CopyPartResult定义方法如下:

// 获取本次Copy后Object的Etag

const Aws::String& GetETag() const

//获取本次Copy的上传时间

const Aws::Utils::DateTime& GetLastModified() const

#### 示例

#include <sstream>

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/UploadPartCopyRequest.h>

#include <aws/s3/model/CreateMultipartUploadRequest.h>

#include <aws/s3/model/CompleteMultipartUploadRequest.h>

#include <aws/s3/model/HeadObjectRequest.h>

#include <aws/core/Aws.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Client;

using namespace std;

void Copy(S3Client &client)

{

// 设置请求

UploadPartCopyRequest request;

CreateMultipartUploadRequest initRequest;

CompleteMultipartUploadRequest finishRequest;

HeadObjectRequest metaRequest;

Aws::String sourceBucekt = "java-sdk-bucket";

Aws::String sourceKey = "zos-sdk-cpp.tar.gz";

Aws::String destBucket = "java-sdk-bucket";

Aws::String destKey = "Cpp-copy-part";

metaRequest.WithBucket(sourceBucekt);

metaRequest.WithKey(sourceKey);

auto metaOutcome = client.HeadObject(metaRequest);

if (!metaOutcome.IsSuccess())

{

auto err = metaOutcome.GetError();

cout << "ERROR: get meta: " << endl

<< "Http code: " << (int)err.GetResponseCode() << endl

<< "Error Type:" << (int)err.GetErrorType() << endl

<< "Error Msg: " << err.GetMessage() << endl

<< "Exception Name: " << err.GetExceptionName() << endl;

return;

}

auto initResponse = client.CreateMultipartUpload(

initRequest.WithBucket(destBucket)

.WithKey(destKey));

if (!initResponse.IsSuccess())

{

auto err = initResponse.GetError();

cout << "ERROR: init: " << endl

<< "Http code: " << (int)err.GetResponseCode() << endl

<< "Error Type:" << (int)err.GetErrorType() << endl

<< "Error Msg: " << err.GetMessage() << endl

<< "Exception Name: " << err.GetExceptionName() << endl;

return;

}

long long size = metaOutcome.GetResult().GetContentLength();

Aws::Vector<CompletedPart> completedPart;

// 以5M为一段复制Object

long long partSize = 5 \* 1024 \* 1024;

long long bytePosition = 0;

int partNum = 1;

while (bytePosition < size)

{

stringstream ss;

long lastByte = std::min(bytePosition + partSize - 1, size - 1);

// 复制一段

ss << "bytes=" << bytePosition <<"-" << lastByte;

request.WithBucket(destBucket)

.WithKey(destKey)

.WithCopySource(sourceBucekt + "/" + sourceKey)

.WithUploadId(initResponse.GetResult().GetUploadId())

.WithPartNumber(partNum)

.WithCopySourceRange(ss.str().c\_str());

auto uploadOutCome = client.UploadPartCopy(request);

if (!uploadOutCome.IsSuccess())

{

auto err = uploadOutCome.GetError();

cout << "ERROR: uploadPartCopy: " << endl

<< "Http code: " << (int)err.GetResponseCode() << endl

<< "Error Type:" << (int)err.GetErrorType() << endl

<< "Error Msg: " << err.GetMessage() << endl

<< "Exception Name: " << err.GetExceptionName() << endl;

}

auto result = uploadOutCome.GetResult();

CompletedPart part;

part.SetETag(result.GetCopyPartResult().GetETag());

part.SetPartNumber(partNum);

completedPart.push\_back(part);

bytePosition += partSize;

partNum++;

}

CompletedMultipartUpload uploadResult;

uploadResult.SetParts(completedPart);

// 完成 multiPartUpload

finishRequest.WithBucket(destBucket)

.WithKey(destKey)

.WithUploadId(initResponse.GetResult().GetUploadId())

.WithMultipartUpload(uploadResult);

auto finishOutcom = client.CompleteMultipartUpload(finishRequest);

if (!finishOutcom.IsSuccess())

{

auto err = finishOutcom.GetError();

cout << "ERROR: finishUpload: " << endl

<< "Http code: " << (int)err.GetResponseCode() << endl

<< "Error Type:" << (int)err.GetErrorType() << endl

<< "Error Msg: " << err.GetMessage() << endl

<< "Exception Name: " << err.GetExceptionName() << endl;

return;

}

cout << "Object Copyed" << endl;

}

int main(int argc, char \*argv[])

{

Aws::SDKOptions options;

Aws::InitAPI(options);

{

// 设置连接参数

Aws::Client::ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

Aws::Auth::AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

Copy(client);

}

Aws::ShutdownAPI(options);

}

## 2.18、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

Aws::S3::MdSearchOutcome S3Client::MdSearch(const Aws::S3::MdSearchRequest& request) const

#### 参数说明

* request:类型Aws::S3::Model::MdSearchRequest，对象搜索接口参数，定义的方法如下:

/\*\*

\*设置搜索请求的条件，必须设置，Aws::String可通过标准std::string赋值

\*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)’

\*/

void SetQuery(const Aws::String& value)

/\*\*设置搜索模式，可选值：”wild”(通配符)|”fuzzy”(模糊)|”regex”(正

\*则)|”term”(精确,默认)

\*/

void SetQmode(const Aws::String& value)

//设置搜索结果排序方式，可选值：“asc”(升序，默认)|“desc”(降序)

void SetQorder(const Aws::String& value)

//设置搜索结果排序字段，默认根据name排序

void SetQorderkey(const Aws::String& value)

//设置返回搜索结果的起始位置，默认为0

void SetMarker(int value)

//设置搜索结果一次返回的最大数目，默认100，不能超过10000

void SetMax\_keys(int value)

#### 返回结果及说明

* MdSearchOutcome:类型Aws::S3::Model::MdSearchOutcome,搜索接口返回参数，定义的方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

//获取本次请求返回的结果数据

const Aws::S3::Model::MdSearchResult& GetResult()

Aws::S3::Model::MdSearchResult定义的具体方法如下：

//获取搜索结果的容器

const Aws::Vector<Content>& GetContents()

//获取搜索结果是否完整

bool GetIsTruncated()

//获取本次搜索结果起始位置

const Aws::String& GetMarker()

//获取下一次搜索起始位置

const Aws::String& GetNextMarker()

Aws::S3::Model::Content定义的具体方法如下：

//获取对象所在bucket

const Aws::String& GetBucket()

//获取对象类型

const Aws::String& GetContentType()

//获取对象etag

const Aws::String& GetETag()

//获取对象instance

const Aws::String& GetInstance()

//获取对象名称

const Aws::String& GetKey()

//获取对象最后一次修改时间

const Aws::Utils::DateTime& GetLastModified()

//获取对象拥有者

const Owner& GetOwner()

//获取对象大小

long long GetSize()

//获取对象存储级别,StorageClass是一个enum class,包括NOT\_SET,STANDARD,REDUCED\_REDUNDANCY,STANDARD\_IA,ONEZONE\_IA,INTELLIGENT\_TIERING,GLACIER,DEEP\_ARCHIVE,OUTPOSTS

const StorageClass& GetStorageClass()

//获取对象VersionedEpoch

int GetVersionedEpoch()

Aws::S3::Model::Content定义的具体方法如下：

//获取对象拥有者的DisplayName

const Aws::String& GetDisplayName()

//获取对象拥有者的ID

const Aws::String& GetID()

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/MdSearchRequest.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void md\_search(S3Client &client) {

// 设置请求参数

MdSearchRequest request;

request.SetQmode("wild");

request.SetQorder("asc");

request.SetQuery("name==im\*");

// 发出请求

auto outcome = client.MdSearch(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: MdSearch: " << "Http code: "<< (int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " <<

err.GetMessage() << std::endl;

} else {

std::cout << "successful Metadata Search" << std::endl;

auto result = outcome.GetResult();

std::cout << "marker: " << result.GetMarker() << std::endl;

auto mdContents = result.GetContents();

for (auto mdContent: mdContents) {

std::cout << "Bucket: " << mdContent.GetBucket() << std::endl

<< "Key: " << mdContent.GetKey() << std::endl

<< "ContentType: " << mdContent.GetContentType() << std::endl

<< "Etag: " << mdContent.GetETag() << std::endl

<< "Instance: " << mdContent.GetInstance() << std::endl

<< "LastModified: " << mdContent.GetLastModified().ToGmtString(Utils::DateFormat::ISO\_8601) << std::endl

<< "Size: " << mdContent.GetSize() << std::endl

<< "VersionedEpoch: " << mdContent.GetVersionedEpoch() <<

std::endl;

}

}

return;

}

int main(int argc, char\* argv[])

{

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

md\_search(client);

Aws::ShutdownAPI(options);

}

# 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

Aws::S3::Model::CreateMultipartUploadOutcome CreateMultipartUpload(const Aws::S3::Model::CreateMultipartUploadRequest& request) const

#### 参数说明

* request:类型Aws::S3::Model::CreateMultipartUploadRequest，CreateMultipartUpload请求接口的参数，具体方法定义如下:

//设置Object所在Bucket名称，必须设置

void SetBucket(const Aws::String& value)

//设置文件在集群中的名字，一般和文件名一致，必须设置

void SetKey(const Aws::String& value)

//设置文件的存储级别，StorageClass是个enum class，有0到8种级别，分别对应NOT\_SET、STANDARD、REDUCED\_REDUNDANCY、STANDARD\_IA、ONEZONE\_IA、INTELLIGENT\_TIERING、GLACIER、DEEP\_ARCHIVE和OUTPOSTS

void SetStorageClass(const Aaws::S3::Model::StorageClass& value)

//设置文件的ACL规则，ObjectCannedACL是个enum class，有0到7种规则，分别对应NOT\_SET、private\_、public\_read、public\_read\_write、authenticated\_read、aws\_exec\_read、bucket\_owner\_read和bucket\_owner\_full\_control

void SetACL(const Aws::S3::Model::ObjectCannedACL& value)

//设置文件的标签，字符串必须满足这种Key1=Value1的形式，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

void SetTagging(const Aws::String& value)

#### 返回结果说明

* Aws::S3::Model::CreateMultipartUploadOutcome:CreateMultipartUpload请求接口的返回参数，定义的方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

//获取本次请求返回的结果数据

const Aws::S3::Model::CreateMultipartUploadResult& GetResult()

Aws::S3::Model::CreateMultipartUploadResult定义的具体方法如下:

//获取分段上传的uploadid

const Aws::String& GetUploadId()

//获取文件在集群中保存的名字

const Aws::String& GetKey()

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/CreateMultipartUploadRequest.h>

#include <aws/s3/model/ObjectCannedACL.h>

#include <aws/s3/model/StorageClass.h>

#include <aws/core/Aws.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void create\_mulripart\_upload(S3Client &client, const Aws::String &bucket\_name, const Aws::String &key\_name) {

// 设置请求参数

CreateMultipartUploadRequest request;

request.SetBucket(bucket\_name);

request.SetKey(key\_name);

request.SetTagging("key1=value1");

// 发出请求

auto outcome = client.CreateMultipartUpload(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: create\_mulripart\_upload: " << "Http code: "<<

(int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " <<

err.GetMessage() << std::endl;

} else {

auto outresult = outcome.GetResult();

std::cout<<outresult.GetUploadId()<<std::endl;

std::cout << "request success " << std::endl;

}

return;

}

int main(int argc, char\* argv[])

{

if(argc < 3)

{

std::cout << "at least bucket name and object name are needed" <<

std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

const std::string key\_name = argv[2];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

create\_mulripart\_upload(client, bucket\_name, key\_name);

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model::UploadPartOutcome UploadPart(const Aws::S3::Model::UploadPartRequest& request)

#### 参数说明

* request:类型Aws::S3::Model::UploadPartOutcome，UploadPart请求接口的参数，具体方法定义如下:

//设置Object所在Bucket名称，必须设置

void SetBucket(const Aws::String& value)

//设置文件在集群中的名字，一般和文件名一致，必须设置

void SetKey(const Aws::String& value)

//设置3.1中返回的UploadId，必须设置

void SetUploadId(const Aws::String& value)

//设置文件数据，必须设置,Aws::IOStream的使用参考示例

void SetBody(const std::shared\_ptr<Aws::IOStream>& body)

//设置分段号，从1开始计算，必须设置

void SetPartNumber(int value)

//设置本次文件上传的大小，非必须，接口可自行计算

void SetContentLength(long long value)

#### 返回结果说明

* Aws::S3::Model::UploadPartOutcome :UploadPart请求接口的返回参数，其具体定义方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

//获取本次请求返回的结果数据

const Aws::S3::Model::UploadPartResult& GetResult()

Aws::S3::Model::UploadPartResult定义的方法如下:

//获取当前分段的Etag，用于complete multipart upload的时候使用

const Aws::String& GetETag()

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/UploadPartRequest.h>

#include <aws/core/Aws.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

#include <sys/stat.h>

#include <iostream>

#include <fstream>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void upload\_part(S3Client &client, const Aws::String &bucket\_name,

const Aws::String &key\_name,

const Aws::String &object\_name,

int partnumber ) {

//确保文件存在

struct stat buffer;

if (stat(object\_name.c\_str(), &buffer) == -1)

{

std::cout << "Error: upload\_part: File '" <<

object\_name << "' does not exist." << std::endl;

return;

}

std::shared\_ptr<Aws::IOStream> input\_data =

Aws::MakeShared<Aws::FStream>("SampleAllocationTag",

object\_name.c\_str(),

std::ios\_base::in | std::ios\_base::binary);

// 设置请求参数

UploadPartRequest request;

request.SetBucket(bucket\_name);

request.SetKey(key\_name);

request.SetBody(input\_data);

request.SetUploadId("2~Czg8OK5viHeISmIAeXdE1kT1V5SIWkU");

request.SetPartNumber(partnumber);

// 发出请求

auto outcome = client.UploadPart(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: upload\_part: " << "Http code: "<< (int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " <<

err.GetMessage() << "Exception name:" << err.GetExceptionName() << std::endl;

} else {

auto outresult = outcome.GetResult();

std::cout<<outresult.GetETag()<<std::endl;

std::cout << "request success " << std::endl;

}

return;

}

int main(int argc, char\* argv[])

{

if(argc < 5)

{

std::cout << "at least bucket name、object name、key\_name and partnumber are needed" << std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

const std::string key\_name = argv[2];

const std::string object\_name = argv[3];

const int partnumber = atoi(argv[4]);

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

upload\_part(client, bucket\_name, key\_name, object\_name, partnumber);

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model::CompleteMultipartUploadOutcome CompleteMultipartUpload(const Aws::S3::Model::CompleteMultipartUploadRequest& request)

#### 参数说明

* request:类型Aws::S3Model::CompleteMultipartUploadRequest,CompleteMultipartUpload请求接口的参数，具体定义方法如下:

//设置Object所在Bucket名称，必须设置

void SetBucket(const Aws::String& value)

//设置文件在集群中的名字，一般和文件名一致，必须设置

void SetKey(const Aws::String& value)

//设置3.1中返回的UploadId，必须设置

void SetUploadId(const Aws::String& value)

//设置该分段上传不同的分段信息，必须设置

void SetMultipartUpload(const Aws::S3::Model::CompletedMultipartUpload& value)

Aws::S3::Model::CompletedMultipartUpload的方法定义如下:

//设置分段上传不同的分段信息，必须按照分段号从小到大进行设置，必须设置

void SetParts(const Aws::Vector<CompletedPart>& value)

Aws::S3::Model::CompletedPart的方法定义如下

//设置该分段的Etag，必须设置

void void SetETag(const Aws::String& value)

//设置该分段的分段号，必须设置

void SetPartNumber(int value)

#### 返回结果说明

* Aws::S3::Model::CompleteMultipartUploadOutcome:CompleteMultipartUpload请求接口的返回参数，具体定义如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

//获取本次请求返回的结果数据

const Aws::S3::Model::CompleteMultipartUploadResult& GetResult()

Aws::S3::Model::CompleteMultipartUploadResult定义方法如下:

//获取整体文件的Etag

const Aws::String& GetETag()

//获取文件位置

const Aws::String& GetLocation() const

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/CompleteMultipartUploadRequest.h>

#include <aws/core/Aws.h>

#include <aws/s3/model/CompletedMultipartUpload.h>

#include <aws/s3/model/CompletedPart.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

#include <sys/stat.h>

#include <iostream>

#include <fstream>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void complete\_multipart\_upload(S3Client &client, const Aws::String &bucket\_name, const Aws::String &key\_name ) {

// 设置请求参数

CompleteMultipartUploadRequest request;

request.SetBucket(bucket\_name);

request.SetKey(key\_name);

request.SetUploadId("2~Czg8OK5viHeISmIAeXdE1kT1V5SIWkU");

CompletedPart part1;

part1.SetPartNumber(1);

part1.SetETag("da6a0d097e307ac52ed9b4ad551801fc");

CompletedPart part2;

part2.SetPartNumber(2);

part2.SetETag("5f363e0e58a95f06cbe9bbc662c5dfb6");

Aws::Vector<CompletedPart> vec\_value;

vec\_value.push\_back(part1);

vec\_value.push\_back(part2);

CompletedMultipartUpload value;

value.SetParts(vec\_value);

request.SetMultipartUpload(value);

// 发出请求

auto outcome = client.CompleteMultipartUpload(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: complete\_multipart\_upload: " << "Http code:

"<< (int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " <<

err.GetMessage() << "Exception name:" << err.GetExceptionName() << std::endl;

} else {

auto outresult = outcome.GetResult();

std::cout<< "GetETag:" << outresult.GetETag() <<std::endl;

std::cout<< "GetLocation:" << outresult.GetLocation() <<std::endl;

std::cout << "request success " << std::endl;

}

return;

}

int main(int argc, char\* argv[])

{

if(argc < 2)

{

std::cout << "at least bucket name and key name is needed" << std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

const std::string key\_name = argv[2];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

complete\_multipart\_upload(client, bucket\_name, key\_name);

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model::ListPartsOutcome ListParts(const Aws::Model::ListPartsRequest& request) const

#### 参数说明

* request:类型Aws::S3::Model::ListPartsRequest，ListParts请求接口的参数，具体定义方法如下:

//设置Bucket名称，必须设置

void SetBucket(const Aws::String& value)

//设置文件在集群中的名字，一般和文件名一致，必须设置

void SetKey(const Aws::String& value)

//设置分段上传的ID，在Create Multipart Upload中返回，必须设置

void SetUploadId(const Aws::String& value)

//设置返回的最多分段数目，最大1000

void SetMaxParts(int value)

//设置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

void SetPartNumberMarker(int value)

#### 返回结果说明

* Aws::S3::Model::ListPartsOutCome:ListParts请求接口的返回参数，具体定义方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

//获取本次请求返回的结果数据

const Aws::S3::Model::ListPartsResult& GetResult()

Aws::S3::Model::ListPartsResult定义的方法如下:

//获取文件保存的Bucket

const Aws::String& GetBucket()

//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 GetNextPartNumberMarker()

//是否是截断获取

bool GetIsTruncated() const

//获取当前list的分段起始编号

int GetPartNumberMarker() const

//获取当前list返回的最大分段数目

int GetMaxParts() const

//获取分段上传的UploadID

const Aws::String& GetUploadId() const

//获取文件的存储级别，StorageClass是个enum class，从0到8分别对应NOT\_SET、STANDARD、REDUCED\_REDUNDANCY、STANDARD\_IA、ONEZONE\_IA、INTELLIGENT\_TIERING、GLACIER、DEEP\_ARCHIVE和OUTPOSTS

const StorageClass& GetStorageClass() const

//获取文件在集群中保存的Key名字

const Aws::String& GetKey() const

//获取文件所属的用户，Owner是个class，主要包括两部分:display name和ID，可以通过方法const Aws::String& GetDisplayName()和const Aws::String& GetID()来获取

const Owner& GetOwner() const

//获取分段信息

const Aws::Vector<Aws::S3::Model::Part>& GetParts() const

Aws::S3::Model::Part定义的方法如下:

//获取当前分段的分段号

int GetPartNumber() const

//获取该分段上一次修改时间，DateTime是个class，可以通过其方法int64\_t Millis() const获取对应Unix时间戳，详细用法参考示例

const Aws::Utils::DateTime& GetLastModified()

//获取该分段的Etag

const Aws::String& GetETag() const

//获取当前分段的大小，单位字节

long long GetSize() const

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/ListPartsRequest.h>

#include <aws/core/Aws.h>

#include <aws/s3/model/Part.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

#include <sys/stat.h>

#include <iostream>

#include <fstream>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void list\_parts(S3Client &client, const Aws::String &bucket\_name, const

Aws::String &key\_name) {

// 设置请求参数

ListPartsRequest request;

request.SetBucket(bucket\_name);

request.SetKey(key\_name);

request.SetUploadId("2~Czg8OK5viHeISmIAeXdE1kT1V5SIWkU");

// 发出请求

auto outcome = client.ListParts(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: list\_part: " << "Http code: "<< (int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " <<

err.GetMessage() << "Exception name:" << err.GetExceptionName() << std::endl;

} else {

auto outresult = outcome.GetResult();

std::cout<< "GetBucket:" << outresult.GetBucket() <<std::endl;

std::cout<< "GetNextPartNumberMarker:" << outresult.GetNextPartNumberMarker() <<std::endl;

std::cout<< "GetIsTruncated:" << outresult.GetIsTruncated() <<std::endl;

std::cout<< "GetMaxParts:" << outresult.GetMaxParts() <<std::endl;

std::cout<< "GetUploadId:" << outresult.GetUploadId() <<std::endl;

std::cout<< "GetStorageClass:" << (int)outresult.GetStorageClass()

<<std::endl;

auto owner = outresult.GetOwner();

std::cout<< "Owner:GetDisplayName:" << owner.GetDisplayName() <<std::endl;

std::cout<< "Owner:GetID:" << owner.GetID() <<std::endl;

auto parts = outresult.GetParts();

for(auto part: parts){

std::cout<< "part:GetPartNumber:" << part.GetPartNumber() <<std::endl;

std::cout<< "part:GetLastModified:" << part.GetLastModified().Millis() <<std::endl;

std::cout<< "part:GetETag:" << part.GetETag() <<std::endl;

std::cout<< "part:GetSize:" << part.GetSize() <<std::endl;

}

std::cout << "request success " << std::endl;

}

return;

}

int main(int argc, char\* argv[])

{

if(argc < 3)

{

std::cout << "at least bucket name and key\_name are needed" << std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

const std::string key\_name = argv[2];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

list\_parts(client, bucket\_name, key\_name);

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model::AbortMultipartUploadOutcome AbortMultipartUpload(const Aws::Model::AbortMultipartUploadRequest& request) const

#### 参数说明

* request:类型Aws::Model::AbortMultipartUploadRequest请求接口的参数，具体定义方法如下:

//设置Bucket名称，必须设置

void SetBucket(const Aws::String& value)

//设置文件在集群中的名字，一般和文件名一致，必须设置

void SetKey(const Aws::String& value)

//设置分段上传的ID，在Create Multipart Upload中返回，必须设置

void SetUploadId(const Aws::String& value)

#### 返回结果说明

* Aws::S3::Model::AbortMultipartUploadOutcome:

/本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/AbortMultipartUploadRequest.h>

#include <aws/core/Aws.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

#include <sys/stat.h>

#include <iostream>

#include <fstream>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void abort\_multipart\_upload(S3Client &client, const Aws::String &bucket\_name, const Aws::String &key\_name ) {

// 设置请求参数

AbortMultipartUploadRequest request;

request.SetBucket(bucket\_name);

request.SetKey(key\_name);

request.SetUploadId("2~BMsfWyZeOqdZPiFL4aZYGzsx7V5qxZW");

// 发出请求

auto outcome = client.AbortMultipartUpload(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: AbortMultipartUpload: " << "Http code: "<<

(int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " <<

err.GetMessage() << "Exception name:" << err.GetExceptionName() << std::endl;

} else {

auto outresult = outcome.GetResult();

std::cout << "request success " << std::endl;

}

return;

}

int main(int argc, char\* argv[])

{

if(argc < 2)

{

std::cout << "at least bucket name and key name is needed" << std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

const std::string key\_name = argv[2];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

abort\_multipart\_upload(client, bucket\_name, key\_name);

Aws::ShutdownAPI(options);

}

## 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

Aws::S3::Model::ListMultipartUploadsOutcome ListMultipartUploads(const Aws::S3::Model::ListMultipartUploadsRequest& request) const

#### 参数说明

* request:类型Aws::S3::Model::ListMultipartUploadsRequest,ListMultipartUploads请求接口的参数，具体定义方法如下:

//设置Bucket名称，必须设置

void SetBucket(const Aws::String& value)

//设置KeyMarker，只有Key大于KeyMarker的分段上传数据才会返回

void SetKeyMarker(const Aws::String& value)

//设置Key的前缀Prefix，只有以Prefix为开头的Key的分段上传数据才会返回

void SetPrefix(const Aws::String& value)

//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

void SetMaxUploads(int value)

#### 返回结果说明

* Aws::S3::Model::ListMultipartUploadsOutcome:ListMultipartUploads请求接口的返回参数，具体定义方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

//获取本次请求返回的结果数据

const Aws::S3::Model::ListMultipartUploadsResult& GetResult()

Aws::S3::Model::ListMultipartUploadsResult的方法定义如下:

//获取本次请求最大可返回的分段上传数据

int GetMaxUploads()

//获取本次请求返回的分段上传数据

const Aws::Vector<Aws::S3::Model::MultipartUpload>& GetUploads()

Aws::S3::Model::MultipartUpload的方法定义如下:

//获取分段上传ID

const Aws::String& GetUploadId()

//获取分段上传的

const Aws::String& GetKey()

//获取文件存储级别，Storage是个enum class， 从0到8分别对应NOT\_SET、STANDARD、REDUCED\_REDUNDANCY、STANDARD\_IA、ONEZONE\_IA、INTELLIGENT\_TIERING、GLACIER、DEEP\_ARCHIVE和OUTPOSTS

const StorageClass& GetStorageClass()

//获取初始分段上传数据的用户，Initiator是个class，可以通过const Aws::String& GetID()和const Aws::String& GetDisplayName() const方法来获取用户ID和DisplayName

const Aws::S3::Model::Initiator& GetInitiator()

//获取分段所属用户，Owen是个class，可以通过const Aws::String& GetID()const 和const Aws::String& GetDisplayName() const方法来获取用户ID和DisplayName

const Aws::S3::Model::Owner& GetOwner()

//获取分段上传的初始化时间,DateTime是个class,可以通过其方法int64\_t Millis() const获取对应Unix时间戳，详细用法参考示例

const Aws::Utils::DateTime& GetInitiated() const

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/ListMultipartUploadsRequest.h>

#include <aws/core/Aws.h>

#include <aws/s3/model/MultipartUpload.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

#include <sys/stat.h>

#include <iostream>

#include <fstream>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void list\_multi\_uploads(S3Client &client,

const Aws::String &bucket\_name) {

// 设置请求参数

ListMultipartUploadsRequest request;

request.SetBucket(bucket\_name);

// 发出请求

auto outcome = client.ListMultipartUploads(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: list\_multi\_uploads: " << "Http code: "<< (int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " << err.GetMessage() << "Exception name:" << err.GetExceptionName() << std::endl;

} else {

auto outresult = outcome.GetResult();

std::cout<< "GetMaxUploads:" << outresult.GetMaxUploads() <<std::endl;

auto uploads = outresult.GetUploads();

for(auto upload: uploads){

std::cout<< "upload:GetUploadId:" << upload.GetUploadId() <<std::endl;

std::cout<< "upload:GetKey:" << upload.GetKey() <<std::endl;

std::cout<< "upload:GetStorageClass:" << (int)upload.GetStorageClass() <<std::endl;

std::cout<< "upload:GetInitiated:" << upload.GetInitiated().Millis() <<std::endl;

auto initiator = upload.GetInitiator();

std::cout<< "upload:initiator:GetID:" << initiator.GetID() <<std::endl;

std::cout<< "upload:initiator:GetDisplayName:" << initiator.GetDisplayName() <<std::endl;

auto owner = upload.GetOwner();

std::cout<< "upload:owner:GetID:" << owner.GetID() <<std::endl;

std::cout<< "upload:owner:GetDisplayName:" << owner.GetDisplayName() <<std::endl;

}

std::cout << "request success " << std::endl;

}

return;

}

int main(int argc, char\* argv[])

{

if(argc < 2)

{

std::cout << "at least bucket name is needed" << std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

list\_multi\_uploads(client, bucket\_name);

Aws::ShutdownAPI(options);

}

# 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，并持久化到指定的存储桶中。

#### 方法原型

Aws::S3::Model::ProcessObjectOutcome S3Client::ProcessObject(const Aws::S3::Model::ProcessObjectRequest& request)

#### 参数说明

* request:类型Aws::S3::Model::ProcessObject，图片处理持久化接口参数，定义的方法如下:

//设置持久化的目的桶,必须设置

void SetBucket(const Aws::String& value)

//设置待处理的图片所在桶和对象名,必须设置

void SetProcessSource(const Aws::String& value)

//设置持久化存储的目的名称,必须设置

void SetKey(const Aws::String& value)

//设置图片处理方式，包含缩放(resize)，裁剪(crop)，旋转(rotate)，水印(watermark)，//格式转换(format)，获取信息功能(info),必须设置

//e.g. image/resize,w\_200

void SetZosProcess(const Aws::String& value)

//设置用于请求的请求体，请求体为具体的处理方法,e.g. image/resize,w\_200,必须设置

void SetBody(const std::shared\_ptr<Aws::IOStream>& body)

SetZosProcess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)

#### 返回结果及说明

* ProcessObjectOutcome:类型Aws::S3::Model::ProcessObjectOutcome,图片持久化请求接口返回参数，定义的方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

//获取本次请求返回的结果数据

const Aws::S3::Model::ProcessObjectResult& GetResult()

Aws::S3::Model::ProcessObjectResult定义的具体方法如下：

//获取对象etag

const Aws::String& GetETag()

//获取对象versionid

const Aws::String& GetVersionId()

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/ProcessObjectRequest.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void process\_object(S3Client &client, const Aws::String &dest\_bucket, const Aws::String &dest\_name, const Aws::String &process\_source, const

Aws::String &process\_method) {

// 设置请求参数

ProcessObjectRequest request;

request.SetBucket(dest\_bucket);

request.SetKey(dest\_name);

request.SetProcessSource(process\_source);

request.SetBody(std::make\_shared<std::stringstream>(process\_method));

// 发出请求

auto outcome = client.ProcessObject(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: ProcessObj: " << "Http code: "<< (int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " <<

err.GetMessage() << std::endl;

} else {

auto result = outcome.GetResult();

std::cout << "successful Process image obj: " << process\_source << std::endl

<< "etag: " << result.GetETag() << std::endl

<< "VersionId: " << result.GetVersionId() << std::endl;

}

return;

}

{

if(argc != 5)

{

std::cout << "pls input: dest\_buckek dest\_name process\_source process\_method" << std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

const std::string obj\_name = argv[2];

const std::string process\_source = argv[3];

const std::string process\_method = argv[4];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

process\_object(client, bucket\_name, obj\_name, process\_source, process\_method);

Aws::ShutdownAPI(options);

}

## 4.2、Get请求获取图片

#### 功能说明

图片处理是在get\_object基础上进行的扩展，用来对存储桶中的图片对象在线进行处理。

#### 方法原型

Aws::S3::Model::GetObjectOutcome S3Client::GetObject(const Aws::S3::Model::GetObjectRequest& request)

#### 参数说明

* request:类型Aws::S3::Model::GetObjectRequest，图片处理接口参数，定义的方法如下:

//设置图片所在桶,必须设置

void SetBucket(const Aws::String& value)

//设置待处理图片名称,必须设置

void SetKey(const Aws::String& value)

//设置图片处理方式，包含缩放(resize)，裁剪(crop)，旋转(rotate)，水印(watermark)，//格式转换(format)，获取信息功能(info),定义方法同5.1参数说明必须设置

//e.g. image/resize,w\_200

void SetZosProcess(const Aws::String& value)

#### 返回结果及说明

* GetObjectOutcome:类型Aws::S3::Model::GetObjectOutcome,图片处理请求接口返回参数，定义的方法如下:

//本次请求是否成功

void IsSuccess(const Aws::String& value)

//获取本次请求错误类，S3Error方法定义参考[S3Error类方法定义](#_全局错误码及类定义)

const Aws::S3::S3Error& GetError()

//获取本次请求返回的结果数据

const Aws::S3::Model::GetObjectResult& GetResult()

Aws::S3::Model::GetObjectResult定义的具体方法如下：

//获取body

Aws::IOStream& GetBody()

//获取对象类型

const Aws::String& GetContentType()

//获取对象etag

const Aws::String& GetETag()

//获取对象的单位

const Aws::String& GetAcceptRanges()

//获取对象最后一次修改时间

const Aws::Utils::DateTime& GetLastModified()

//获取body的大小

long long GetContentLength()

#### 示例

#include <aws/core/Aws.h>

#include <aws/s3/S3Client.h>

#include <aws/s3/model/GetObjectRequest.h>

#include <aws/core/auth/AWSCredentialsProvider.h>

#include <fstream>

using namespace Aws;

using namespace Aws::Client;

using namespace Aws::S3;

using namespace Aws::S3::Model;

using namespace Aws::Auth;

using namespace std;

void get\_object(S3Client &client, const Aws::String &bucket\_name, const

Aws::String &obj\_name, const Aws::String &process\_method) {

// 设置请求参数

GetObjectRequest request;

request.SetBucket(bucket\_name);

request.SetKey(obj\_name);

request.SetZosProcess(process\_method);

// 发出请求

auto outcome = client.GetObject(request);

//处理请求结果

if (!outcome.IsSuccess())

{

auto err = outcome.GetError();

std::cout << "ERROR: GetObj: " << "Http code: "<< (int)err.GetResponseCode() <<

" Error Type:" << (int)err.GetErrorType() << " Error Msg: " <<

err.GetMessage() << std::endl;

} else {

auto result = std::move(outcome.GetResult());

ofstream hFile;

//将图片文件处理结果写入磁盘，验证是否符合要求

hFile.open("/root/target.jpg");

hFile << result.GetBody().rdbuf();

hFile.close();

std::cout << "successful get and process obj: " << obj\_name

<< std::endl

<< "AcceptRange: " << result.GetAcceptRanges() << std::endl

<< "ContentType: " << result.GetContentType() << std::endl

<< "Etag" << result.GetETag() <<std::endl

<< "LastModified: " << result.GetLastModified().ToGmtString(Utils::DateFormat::ISO\_8601) << std::endl

<< "ContentLength: " << result.GetContentLength() << std::endl;

for (auto meta : result.GetMetadata()) {

std::cout << meta.first << " : " << meta.second << std::endl;

}

}

return;

}

int main(int argc, char\* argv[])

{

if(argc != 4)

{

std::cout << "pls input args: bucket\_name obj\_name process\_method"

<< std::endl;

return -1;

}

const std::string bucket\_name = argv[1];

const std::string obj\_name = argv[2];

const std::string process\_method = argv[3];

//设置打印级别

Aws::SDKOptions options;

options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;

Aws::InitAPI(options);

// 设置连接参数

ClientConfiguration cfg;

cfg.endpointOverride = "192.168.218.130:7480"; // S3服务器地址和端口

cfg.scheme = Aws::Http::Scheme::HTTP;

cfg.verifySSL = false;

AWSCredentials cred("9L6CF1NST0D4ATG7HFS4", "JCTrIQtmALJ1inU6yQvXHv8Lust1AGIgHD5uN7BD"); // ak,sk

S3Client client(cred, cfg, Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

get\_object(client, bucket\_name, obj\_name, process\_method);

Aws::ShutdownAPI(options);

}