ZOS对象存储C++_SDK使用手册

环境配置

Windows

前置环境要求

- Visual Studio 2015或之后版本；

下载SDK

- SDK压缩包快速下载地址: 见帮助中心

编译及安装

1. 添加环境变量Path，相应路径修改为本地的VS安装路径：

```
C:\Program Files\Microsoft Visual
Studio\2022\Community\Common7\IDE\CommonExtensions\Microsoft\CMake\CMake\bin
```

2. 解压安装包进入目录，以管理员身份打开cmd，执行以下命令编译安装：

```
mkdir build
cd build
cmake ../ -DCMAKE_BUILD_TYPE=Debug
cmake --build . --config=Debug
cmake --install . --config=Debug
```

Linux

前置环境要求

- CMake 3.13或之后版本；
- 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)

  对于CentOS，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
```

下载SDK

- SDK压缩包快速下载地址: 见帮助中心

编译及按照

1. 使用以下命令进行安装:

```
tar -xvf sdk-cpp.tar.gz
mkdir build
cd build
cmake3 ../sdk-cpp
make -j`nproc`
make install
```

构建初始化项目

1. 创建main.cpp

```
#include <aws/core/Aws.h>
#include <aws/core/utils/logging/LogLevel.h>
#include <aws/s3/S3Client.h>
#include <iostream>
#include <aws/core/auth/AWSCredentialsProvider.h>
using namespace Aws;
using namespace Aws::Client;
using namespace Aws::S3;
using namespace Aws::Auth;
using namespace std;

int main()
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    SDKOptions options;
    options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;

    InitAPI(options);
    {
        // 设置client配置
        Aws::Client::ClientConfiguration cfg;
        cfg.endpointOverride = ep;  // S3服务器地址和端口
        cfg.scheme = Http::Scheme::HTTP;
        cfg.verifySSL = false;

        Auth::AWSCredentials cred(ak, sk);  // ak,sk
        // 创建client
        S3::S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

        // 示例代码，列出所有桶
        auto outcome = client.ListBuckets();
        if (outcome.IsSuccess()) {
            cout << "Found " << outcome.GetResult().GetBuckets().size() << "
buckets\n";
            for (auto&& b : outcome.GetResult().GetBuckets()) {
                cout << b.GetName() << endl;
            }
        }
        else {
            cout << "Failed with error: " << outcome.GetError() << endl;
        }
    }

    ShutdownAPI(options);
    return 0;
}
```

客户端配置参数说明：

- ClientConfiguration

参数	类型	是否必选	描述
endpointOverride	String	是	设置终端节点
scheme	Http::Scheme	可选	选择协议：Http::Scheme::HTTP，HTTP协议；Http::Scheme::HTTPS，HTTPS协议。
verifySSL	bool	可选	是否验证SSL证书
maxConnections	unsigned	可选	设置最大连接数
requestTimeoutMs	long	可选	设置请求超时时间

2. 创建CMakeLists.txt，**注意要使用的服务模块设置完整**

```
cmake_minimum_required(VERSION 3.13)

# 设置要使用的服务模块，如果要使用客户端加密，则需要set(SERVICE_COMPONENTS s3 s3-encryption)，以此类推
set(SERVICE_COMPONENTS s3 sts iam sns kms s3-encryption)

# 设置项目名
project("main")

# 至少需要C++ 11
set(CMAKE_CXX_STANDARD 11)

set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD)
```

```
    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        main.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

3. 编译

```
mkdir build
cd build
cmake ../
```

4. 运行

```
#Windows下打开VS工程进行运行
#Linux下直接make
```

全局错误码及类定义

S3Error类方法定义如下:

获取结果方法	描述
HttpResponseCode GetResponseCode()	获取HTTP返回码
const Aws::String& GetMessage()	获取错误信息
ERROR_TYPE GetErrorType()	获取错误类型

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

错误码	错误码描述
422	UnprocessableEntity
500	InternalError

桶操作

创建桶

功能说明

创建桶请求可以在指定账号下创建一个新的桶。**通过SDK创建的桶，如果要在控制台使用，则需手动添加跨域访问配置，允许`https://console.ctyun.cn`域名访问，具体参加设置跨域访问配置。**

方法原型

`CreateBucketOutcome S3Client.CreateBucket(const CreateBucketRequest& request) const`

参数说明

- CreateBucketRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetACL(const BucketCannedACL& value)	否	设置桶ACL，可选项： BucketCannedACL::private_，私有（默认）； BucketCannedACL::public_read，公共读，但只有桶所有者可以写入和更改ACL； BucketCannedACL::public_read_write，公共读写，但只有桶所有者可以写入和更改ACL。
void SetObjectLockEnabledForBucket(bool value)	否	设置是否开启对象锁，开启对象锁会自动开启版本控制且不可关闭。只有在桶创建时可以开启对象锁，后续不可。
void SetAZPolicy(const Aws::String& value)	否	设置桶冗余策略，可选项： "SINGLE-AZ"，单可用区； "MULTI-AZ"，多可用区。
void SetStorageClass(const Aws::String& value)	否	设置桶存储类型，可选项： "STANDARD"，标准存储； "STANDARD_IA"，低频存储； "GLACIER"，归档存储。

返回结果说明

- CreateBucketOutCome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类

代码示例

```
#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();
        cout << "ERROR: " << 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 << "successful created bucket: " << bucket_name << "\n";
    }
    return;
}

int main()
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    create_bucket(client, bucket);
    Aws::ShutdownAPI(options);
}
```

删除桶

功能说明

删除桶请求可以在指定账号下删除指定桶，删除之前要求桶为空。

方法原型

`DeleteBucketOutcome DeleteBucket(const DeleteBucketRequest &request) const`

参数说明

- DeleteBucketRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名

返回结果说明

- DeleteBucketOutCome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类

代码示例

```
#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;
using namespace Aws::Client;
using namespace Aws::S3;
using namespace Aws::S3::Model;
using namespace Aws::Auth;
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: " << 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()
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    DeleteBucket(client, bucket);
    Aws::ShutdownAPI(options);
}
```

判断桶是否存在

功能说明

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

`HeadBucketOutcome HeadBucket(const HeadBucketRequest&request) const`

参数说明

- HeadBucketRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名

返回结果说明

- HeadBucketOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类

代码示例

```
#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();
        if((int)err.GetResponseCode() == 404){
            cout << "bucket: " << bucket_name << " is not exist" << endl;
        }
        else {
            cout << "ERROR: " << 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 << "bucket: " << bucket_name << " is exist" << endl;
    }
    return;
}

int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    head_bucket(client, bucket);
    Aws::ShutdownAPI(options);
}
```

列出所有桶

功能说明

可以列出该用户名下所有的桶列表。

方法原型

`ListBucketsOutcome ListBuckets() const`

参数说明

无

返回结果说明

- HeadBucketOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
ListBucketsResult& GetResult()	获取请求结果

- ListBucketsResult

获取结果方法	描述
const Vector<Bucket>& ListBucketsResult::GetBuckets() const	获取桶列表
const Owner& ListBucketsResult::GetOwner() const	获取Owner

- Bucket

获取结果方法	描述
const Aws::String& GetName() const	获取桶名
const Aws::Utils::DateTime& GetCreationDate() const	获取桶创建时间

- Owner

获取结果方法	描述
const Aws::String& GetDisplayName() const	获取名字
const Aws::String& GetID() const	获取Id

代码示例

```
#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;
using namespace Aws::Client;
using namespace Aws::S3;
using namespace Aws::S3::Model;
using namespace Aws::Auth;
using namespace std;

void ListBuckets(S3Client &client)
{
    // 设置请求
    auto outcome = client.ListBuckets();

    if (!outcome.IsSuccess())
    {
        auto err = outcome.GetError();
        cout << "ERROR: " << 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;
    options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;
    Aws::InitAPI(options);
    {
        String ep = "<your-ep>";
        String ak = "<your-ak>";
        String sk = "<your-sk>";
        String bucket = "<your-bucket>";

        // 设置连接参数
        ClientConfiguration cfg;
        cfg.endpointOverride = ep;
        cfg.scheme = Aws::Http::Scheme::HTTP;
        cfg.verifySSL = false;
        AWSCredentials cred(ak, sk);  // ak,sk
        S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

        ListBuckets(client);
    }
    Aws::ShutdownAPI(options);
}
```

设置桶策略

功能说明

请求用于为某个Bucket设置桶策略。

方法原型

`PutBucketPolicyOutcome S3Client::PutBucketPolicy(const PutBucketPolicyRequest& request) const`

参数说明

- PutBucketPolicyRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetBody(const std::shared_ptr<Aws::IOStream>& body)	是	设置桶策略，通过String表示json格式的桶策略，再加入body中

- json格式的桶策略字段描述

字段	描述	是否必须
Version	保持与AmazonS3一致，当前支持"2012-10-17"	否
Id	桶策略ID，桶策略的唯一标识	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", "NotPrincipal", "Action", "NotAction","Resource", "NotResource", "Condition"。	是
Sid	本条桶策略描述的ID	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"	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"	否，与NotPrincipal只能选一个
NotPrincipal	不被授权人，即指定本条桶策略描述所**排除**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"	否，与Principal只能选一个
Action	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	否，与NotAction只能选一个
NotAction	指定本条桶策略描述所**排除**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	否，与Action只能选一个
Resource	桶策略作用的资源，"arn:aws:s3:::bucket_name"用于指定桶级的策略，适用于对整个桶的操作；"arn:aws:s3:::bucket_name/*"用于指定桶中所有对象的策略，适用于影响桶中所有对象的操作。	是，与NotResource只能选一个
NotResource	桶策略**排除**的资源，"arn:aws:s3:::bucket_name"用于指定桶级的策略，适用于对整个桶的操作；"arn:aws:s3:::bucket_name/*"用于指定桶中所有对象的策略，适用于影响桶中所有对象的操作。	是，与Resource只能选一个
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.ctyun.cn"]}，此时如果Effect为“Allow”，则允许来自"www.ctyun.cn"的请求；如果为“Deny”，则拒绝。	否

返回结果说明

- PutBucketPolicyOutCome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类

代码示例

```
#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>
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[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    Aws::S3::Model::PutBucketPolicyRequest request;
    request.SetBucket(bucket);
    // 示例策略：允许所有人对桶的getobject权限
    Aws::String policyBody =
        "{"
        "   \"Version\": \"2012-10-17\","
        "   \"Statement\": ["
        "       {"
        "           \"Effect\": \"Allow\","
        "           \"Principal\": \"*\","
        "           \"Action\": ["
        "               \"s3:GetObject\""
        "           ],"
        "            \"Resource\": ["
        "                \"arn:aws:s3:::" + bucket + "/*\""
        "            ]"
        "        }"
        "    ]"
        "}";
    shared_ptr<Aws::StringStream> request_body =
Aws::MakeShared<Aws::StringStream>("");
    *request_body << policyBody;
    request.SetBody(request_body);

    auto outcome = client.PutBucketPolicy(request);
    if (outcome.IsSuccess())
    {
        std::cout << "Bucket policy set successfully." << std::endl;
    } else
    {
        auto err = outcome.GetError();
        cout << "ERROR: " << endl
            << "Http code: " << (int)err.GetResponseCode() << endl
            << "Error Type:" << (int)err.GetErrorType() << endl
            << "Error Msg: " << err.GetMessage() << endl
            << "Exception Name: " << err.GetExceptionName() << endl;
    }

    Aws::ShutdownAPI(options);
    return 0;
}
```

获取桶策略

功能说明

请求用于获取某个Bucket的桶策略。

方法原型

`GetBucketPolicyOutcome S3Client::GetBucketPolicy(const GetBucketPolicyRequest& request) const`

参数说明

- GetBucketPolicyRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名

返回结果说明

- GetBucketPolicyOutCome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
const GetBucketPolicyResult& GetResult() const	获取请求返回的结果

- GetBucketPolicyResult

获取结果方法	描述
Aws::IOStream& GetPolicy()	获取桶策略

代码示例

```
#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>
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[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    Aws::S3::Model::GetBucketPolicyRequest request;
    request.SetBucket(bucket);
    auto outcome = client.GetBucketPolicy(request);
    if (outcome.IsSuccess())
    {
        Aws::StringStream ss;
        ss << outcome.GetResult().GetPolicy().rdbuf();
```

```
            cout << "Bucket Policy: " << ss.str() << endl;
        } else
        {
            auto err = outcome.GetError();
            cout << "ERROR: " << endl
                << "Http code: " << (int)err.GetResponseCode() << endl
                << "Error Type:" << (int)err.GetErrorType() << endl
                << "Error Msg: " << err.GetMessage() << endl
                << "Exception Name: " << err.GetExceptionName() << endl;
        }

        Aws::ShutdownAPI(options);
        return 0;
    }
```

删除桶策略

功能说明

请求用于删除某个Bucket的桶策略。

方法原型

`DeleteBucketPolicyOutcome S3Client::DeleteBucketPolicy(const DeleteBucketPolicyRequest& request) const`

参数说明

- DeleteBucketPolicyRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名

返回结果说明

- DeleteBucketPolicyOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类

代码示例

```
#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>
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[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    Aws::S3::Model::DeleteBucketPolicyRequest request;
    request.SetBucket(bucket);

    auto outcome = client.DeleteBucketPolicy(request);
    if (outcome.IsSuccess())
    {
        cout << "Bucket policy delete successfully." << endl;
    } else
    {
        auto err = outcome.GetError();
        cout << "ERROR: " << endl
            << "Http code: " << (int)err.GetResponseCode() << endl
            << "Error Type:" << (int)err.GetErrorType() << endl
            << "Error Msg: " << err.GetMessage() << endl
            << "Exception Name: " << err.GetExceptionName() << endl;
    }
    Aws::ShutdownAPI(options);
    return 0;
}
```

设置桶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。

设置方法有三种：

1. BucketCannedACL，对整个桶的权限设置；
2. AccessControlPolicy，根据给出的配置，具体的给某些用户授予某些权限；
3. Grant方式，直接赋予某个用户某种权限。

方法原型

`PutBucketAclOutcome S3Client::PutBucketAcl(const PutBucketAclRequest& request) const`

参数说明

- PutBucketAclRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetACL(const BucketCannedACL& value)	三种方法选一种	设置ACL（方法1），可选项：BucketCannedACL::private_，私有（默认）；BucketCannedACL::public_read，公共读，但只有桶所有者可以写入和更改ACL；BucketCannedACL::public_read_write，公共读写，但只有桶所有者可以写入和更改ACL。
void SetAccessControlPolicy(const AccessControlPolicy& value)	三种方法选一种	设置ACL（方法2）
void SetGrantFullControl(const Aws::String& value)	三种方法选一种	设置ACL（方法3），给与用户所有权限，格式："id=xxx, uri=xxx, emailAddress=xxx"，只给出id即可。
void SetGrantRead(const Aws::String& value)	三种方法选一种	设置ACL（方法3），给与用户读权限，格式："id=xxx, uri=xxx, emailAddress=xxx"，只给出id即可。
void SetGrantReadACP(const Aws::String& value)	三种方法选一种	设置ACL（方法3），给与用户读ACL权限，格式："id=xxx, uri=xxx, emailAddress=xxx"，只给出id即可。
void SetGrantWrite(const Aws::String& value)	三种方法选一种	设置ACL（方法3），给与用户写权限，格式："id=xxx, uri=xxx, emailAddress=xxx"，只给出id即可。
void SetGrantWriteACP(const Aws::String& value)	三种方法选一种	设置ACL（方法3），给与用户写ACL权限，格式："id=xxx, uri=xxx, emailAddress=xxx"，只给出id即可。

- AccessControlPolicy

设定参数方法	是否必选	描述
void SetOwner(const Owner& value)	是	设置桶所有者
void SetGrants(const Aws::Vector<Grant>& value)	是	设置授权列表

- Owner

设定参数方法	是否必选	描述
void SetID(const Aws::String& value)	是	设置ID
void SetDisplayName(const Aws::String& value)	否	设置名字

- Grant

设定参数方法	是否必选	描述
void SetGrantee(const Grantee& value)	是	设置被授权用户
void SetPermission(const Permission& value)	是	设置权限： Permission::FULL_CONTROL，所有权限； Permission::READ，读权限； Permission::READ_ACP，读ACL权限； Permission::WRITE，写权限； Permission::WRITE_ACP，写ACL权限。

- Grantee

设定参数方法	是否必选	描述
void SetType(const Type& value)	是	设置被授权用户类型： Type::CanonicalUser，规范用户； Type::Group，IAM组。
void SetID(const Aws::String& value)	是（CanonicalUser）	设置ID
void SetURI(const Aws::String& value)	是（Group）	设置组URI
void SetDisplayName(const Aws::String& value)	可选	设置名字

返回结果说明

- PutBucketAclOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类

代码示例

```
#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);
    // 第一种方式
    request.SetACL(Aws::S3::Model::BucketCannedACL::public_read);

    // 第二种方式
    //Aws::S3::Model::Owner owner;
    //owner.SetID("testid");
    //Aws::S3::Model::Grantee grantee;
    //grantee.SetType(Aws::S3::Model::Type::CanonicalUser);
    //grantee.SetID("test");
    //Aws::S3::Model::Grant grant;
    //grant.SetGrantee(grantee);
    //grant.SetPermission(Aws::S3::Model::Permission::FULL_CONTROL);
    //Aws::Vector<Aws::S3::Model::Grant> grants;
    //grants.push_back(grant);
    //Aws::S3::Model::AccessControlPolicy policy;
    //policy.SetOwner(owner);
    //policy.SetGrants(grants);
    //request.SetAccessControlPolicy(policy);

    //第三种方式
    //request.SetGrantFullControl("id=xxx");

    auto outcome = client.PutBucketAcl(request);

    if (!outcome.IsSuccess())
    {
        auto err = outcome.GetError();
        cout << "ERROR: " << 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 << "Set ACL successful" << endl;
    }

    return;
}
int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    put_bucket_acl(client, bucket);
    Aws::ShutdownAPI(options);
}
```

获取桶ACL

功能说明

获取桶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。

方法原型

`GetBucketAclOutcome S3Client::GetBucketAcl(const GetBucketAclRequest& request) const`

参数说明

- GetBucketAclRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名

返回结果说明

- GetBucketAclOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
const GetBucketAclResult& GetResult() const	获取桶ACL

- GetBucketAclResult

获取结果方法	描述
const Owner& GetOwner() const	获取桶Owner
const Aws::Vector<Grant>& GetGrants() const	获取授权列表

- Owner

设定参数方法	描述
const Aws::String& GetID() const	获取ID
const Aws::String& GetDisplayName() const	获取名字

- Grant

获取结果方法	描述
const Grantee& GetGrantee() const	获取授权用户信息
const Permission& GetPermission()	获取权限类型： Permission::FULL_CONTROL，所有权限； Permission::READ，读权限； Permission::READ_ACP，读ACL权限； Permission::WRITE，写权限； Permission::WRITE_ACP，写ACL权限。

- Grantee

获取结果方法	描述
const Type& GetType() const	获取用户类型： Type::CanonicalUser，规范用户； Type::Group，IAM组。
const Aws::String& GetID() const	获取用户ID
const Aws::String& GetURI() const	获取用户组URI
const Aws::String& GetDisplayName() const	获取名字

代码示例

```
#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();
        cout << "ERROR: " << 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 << "Get ACL 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[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    get_bucket_acl(client, bucket);
    Aws::ShutdownAPI(options);
}
```

设置桶生命周期配置

功能说明

设置桶的生命周期规则。

方法原型

`PutBucketLifecycleConfigurationOutcome PutBucketLifecycleConfiguration(const PutBucketLifecycleConfigurationRequest& request) const`

参数说明

- PutBucketLifecycleConfigurationRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetLifecycleConfiguration(const BucketLifecycleConfiguration& value)	是	设置生命周期配置

- BucketLifecycleConfiguration

设定参数方法	是否必选	描述
void SetRules(const Aws::Vector<LifecycleRule>& value)	可选	设置规则列表
BucketLifecycleConfiguration& AddRules(const LifecycleRule& value)	可选	添加一条规则

- LifecycleRule

设定参数方法	是否必选	描述
void SetStatus(const ExpirationStatus& value)	是	设置规则是否启用：ExpirationStatus::Enabled，启用；ExpirationStatus::Disabled，不启用。
void SetID(const Aws::String& value)	可选	设置规则ID
void SetFilter(const LifecycleRuleFilter& value)	是	设置对象过滤条件
void SetTransitions(const Aws::Vector<Transition>& value)	可选	设置对象到期转存规则
LifecycleRule& AddTransitions(const Transition& value)	可选	添加一条对象到期转存规则
void SetExpiration(const LifecycleExpiration& value)	可选	设置对象到期删除时间
void SetNoncurrentVersionTransitions(const Aws::Vector<NoncurrentVersionTransition>& value)	可选	设置对象历史版本到期转存规则
LifecycleRule& AddNoncurrentVersionTransitions(const NoncurrentVersionTransition& value)	可选	添加一条对象历史版本到期转存规则
void SetNoncurrentVersionExpiration(const NoncurrentVersionExpiration& value)	可选	设置对象历史版本到期删除时间

- LifecycleRuleFilter

设定参数方法	是否必选	描述
void SetPrefix(const Aws::String& value)	可选	设置对象前缀过滤条件
void SetTag(const Tag& value)	可选	设置对象标签过滤条件

- Tag

设定参数方法	是否必选	描述
void SetKey(const Aws::String& value)	是	设置标签key
void SetValue(const Aws::String& value)	是	设置标签value

- Transition

设定参数方法	是否必选	描述
void SetStorageClass(const TransitionStorageClass& value)	是	设置转存的存储类型，可选项： "STANDARD"，标准存储； "STANDARD_IA"，低频存储； "GLACIER"，归档存储。
void SetDate(const Aws::Utils::DateTime& value)	可选	设置固定时间点转存
void SetDays(int value)	可选	设置固定天数后转存

- LifecycleExpiration

设定参数方法	是否必选	描述
void SetDate(const Aws::Utils::DateTime& value)	可选	设置固定时间点删除
void SetDays(int value)	可选	设置固定天数后删除
void SetExpiredObjectDeleteMarker(bool value)	可选	设置是否删除删除标记

- NoncurrentVersionTransition

设定参数方法	是否必选	描述
void SetStorageClass(const TransitionStorageClass& value)	是	设置转存的存储类型，可选项： "STANDARD"，标准存储； "STANDARD_IA"，低频存储； "GLACIER"，归档存储。
void SetNoncurrentDays(int value)	可选	设置固定天数后转存

- NoncurrentVersionExpiration

设定参数方法	是否必选	描述
void SetNoncurrentDays(int value)	可选	设置固定天数后删除

返回结果说明

- PutBucketLifecycleConfigurationOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类

代码示例

```
#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) {
    // 设置规则，前缀为“xxx”的对象在创建30天后删除
    PutBucketLifecycleConfigurationRequest request;
    BucketLifecycleConfiguration lifecycle;
    LifecycleRuleFilter filter;
    LifecycleRule rule;
    LifecycleExpiration expire;
    filter.SetPrefix("xxx");
    expire.SetDays(30);
    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();
        cout << "ERROR: " << 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 << "successful put bucket lifecycle: " << bucket_name <<
endl;
    }
    return;
}
```

```
int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    put_bucket_lifecycle(client, bucket);

    Aws::ShutdownAPI(options);
}
```

获取桶生命周期配置

功能说明

获取 Bucket 的生命周期规则。

方法原型

`GetBucketLifecycleConfigurationOutcome GetBucketLifecycleConfiguration(const GetBucketLifecycleConfigurationRequest& request) const`

参数说明

- GetBucketLifecycleConfigurationRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名

返回结果说明

- GetBucketLifecycleConfigurationOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
const GetBucketLifecycleConfigurationResult& GetResult()	获取桶生命周期配置

- GetBucketLifecycleConfigurationResult

获取结果方法	描述
const Aws::Vector<LifecycleRule>& GetRules()	获取生命周期规则列表

- LifecycleRule

获取结果方法	描述
const Aws::String& GetID()	获取ID
const LifecycleRuleFilter& GetFilter()	获取过滤器
const ExpirationStatus& GetStatus()	获取规则开启状态：ExpirationStatus::Enabled，启用；ExpirationStatus::Disabled，不启用
const Aws::Vector<Transition>& GetTransitions()	获取对象过期转存规则
const LifecycleExpiration& GetExpiration()	获取对象过期删除规则
const Aws::Vector<NoncurrentVersionTransition>& GetNoncurrentVersionTransitions()	获取对象历史版本过期转存规则
const NoncurrentVersionExpiration& GetNoncurrentVersionExpiration()	获取对象历史版本过期删除规则

- LifecycleRuleFilter

获取结果方法	描述
const Aws::String& GetPrefix()	获取对象前缀过滤条件
const Tag& GetTag()	获取对象标签过滤条件

- Tag

获取结果方法	描述
const Aws::String& GetKey()	获取标签key
const Aws::String& GetValue()	获取标签value

- Transition

获取结果方法	描述
const TransitionStorageClass& GetStorageClass()	获取转存的存储类型，可选项： "STANDARD"，标准存储； "STANDARD_IA"，低频存储； "GLACIER"，归档存储。
const Aws::Utils::DateTime& GetDate()	获取固定时间点转存
int GetDays()	获取固定天数后转存

- LifecycleExpiration

获取结果方法	描述
const Aws::Utils::DateTime& GetDate()	获取固定时间点删除
int GetDays()	获取固定天数后删除
bool GetExpiredObjectDeleteMarker()	获取是否删除删除标记

- NoncurrentVersionTransition

获取结果方法	描述
const TransitionStorageClass& GetStorageClass()	获取转存的存储类型，可选项： "STANDARD"，标准存储； "STANDARD_IA"，低频存储； "GLACIER"，归档存储。
int GetNoncurrentDays()	获取固定天数后转存

- NoncurrentVersionExpiration

获取结果方法	描述
int GetNoncurrentDays()	获取固定天数后删除

代码示例

```
#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();
        cout << "ERROR: " << 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 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[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    get_bucket_lifecycle(client, bucket);

    Aws::ShutdownAPI(options);
```

```
}
```

删除桶生命周期配置

功能说明

删除 Bucket 的生命周期规则。

方法原型

`DeleteBucketLifecycleOutcome DeleteBucket (const DeleteBucketLifecycleRequest& request) const`

参数说明

- DeleteBucketLifecycleRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名

返回结果说明

- DeleteBucketLifecycleOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类

代码示例

```
#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();
        cout << "ERROR: " << 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 << "successful delete bucket lifecycle: " << bucket_name << endl;
    }
    return;
}

int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    del_bucket_lifecycle(client, bucket);
    Aws::ShutdownAPI(options);
}
```

设置桶合规保留配置

功能说明

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。**必须在创建桶时设置开启对象锁，不然后续无法进行配置。**

方法原型

`PutObjectLockConfigurationOutcome PutObjectLockConfiguration(const PutObjectLockConfigurationRequest& request) const`

参数说明

- PutObjectLockConfigurationRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetObjectLockConfiguration(const ObjectLockConfiguration& value)	是	设置合规保留规则

- ObjectLockConfiguration

设定参数方法	是否必选	描述
void SetObjectLockEnabled(const ObjectLockEnabled& value)	是	设置是否启用，需设置为：ObjectLockEnabled::Enabled
void SetRule(const ObjectLockRule& value)	是	设置规则

- ObjectLockRule

设定参数方法	是否必选	描述
void SetDefaultRetention(const DefaultRetention& value)	是	设置默认保留期限

- DefaultRetention

设定参数方法	是否必选	描述
void SetMode(const ObjectLockRetentionMode& value)	是	设置模式：ObjectLockRetentionMode::COMPLIANCE，设置后不允许修改；ObjectLockRetentionMode::GOVERNANCE，设置后允许修改。
void SetDays(int value)	可选	设置保留天数
void SetYears(int value)	可选	设置保留年数

返回结果说明

- PutObjectLockConfigurationOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类

代码示例

```
#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();
        cout << "ERROR: " << 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 << "successful put bucket object lock: " << bucket_name <<
"\n";
    }
    return;
}

int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    put_bucket_object_lock(client, bucket);
    Aws::ShutdownAPI(options);
}
```

获取桶合规保留配置

功能说明

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

`GetObjectLockConfigurationOutcome GetObjectLockConfiguration(const GetObjectLockConfigurationRequest& request) const`

参数说明

- GetObjectLockConfigurationRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名

返回结果说明

- GetObjectLockConfigurationOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
GetObjectLockConfigurationResult& GetResult()	获取结果

- GetObjectLockConfigurationResult

获取结果方法	描述
const ObjectLockConfiguration& GetObjectLockConfiguration() const	获取合规保留配置

- ObjectLockConfiguration

获取结果方法	描述
const ObjectLockEnabled& GetObjectLockEnabled() const	查看是否启用
const ObjectLockRule& GetRule() const	获取规则

- ObjectLockRule

获取结果方法	描述
const DefaultRetention& GetDefaultRetention() const	获取默认保留期限

- DefaultRetention

获取结果方法	描述
const ObjectLockRetentionMode& GetMode() const	获取模式
int GetDays() const	获取保留天数
int GetYears() const	获取保留年数

代码示例

```
#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();
        cout << "ERROR: " << 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 << "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[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    get_bucket_object_lock(client, bucket);
    Aws::ShutdownAPI(options);
}
```

设置桶跨域访问配置

功能说明

请求设置 Bucket 的跨域资源访问配置。

方法原型

`PutBucketCorsOutcome PutBucketCors(const PutBucketCorsRequest& request) const`

参数说明

- PutBucketCorsRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetCORSConfiguration(const CORSConfiguration& value)	是	设置跨域访问配置

- CORSConfiguration

设定参数方法	是否必选	描述
void SetCORSRules(const Aws::Vector<CORSRule>& value)	是	设置规则列表

- CORSRule

设定参数方法	是否必选	描述
void SetAllowedHeaders(const Aws::Vector<Aws::String>& value)	可选	设置客户端允许携带的headers
void SetAllowedMethods(const Aws::Vector<Aws::String>& value)	可选	设置允许的http方法
void SetAllowedOrigins(const Aws::Vector<Aws::String>& value)	可选	设置允许的域名来源
void SetExposeHeaders(const Aws::Vector<Aws::String>& value)	可选	设置响应中允许暴露的headers
void SetID(const Aws::String& value)	可选	设置规则ID
void SetMaxAgeSeconds(int value)	可选	设置有效时间（秒）

返回结果说明

- PutBucketCorsOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类

代码示例

```
#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://ip:port");
    origions.push_back("http://ip:port");
    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);
    auto outcome = client.PutBucketCors(req);

    if (!outcome.IsSuccess())
    {
        auto err = outcome.GetError();
        cout << "ERROR: " << 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 << "successful put bucket Cors: " << bucketName << "\n";
    }
    return;
}

int main()
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    PutBucketCors(client, bucket);

    Aws::ShutdownAPI(options);
    return 0;
}
```

获取桶跨域访问配置

功能说明

请求获取 Bucket 的跨域资源访问配置。

方法原型

`GetBucketCorsOutcome GetBucketCors(const GetBucketCorsRequest& request) const`

参数说明

- GetBucketCorsRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名

返回结果说明

- GetBucketCorsOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
GetBucketCorsResult& GetResult()	获取结果

- GetBucketCorsResult

获取结果方法	描述
const Aws::Vector<CORSRule>& GetCORSRules() const	获取跨域访问规则

- CORSRule

获取结果方法	描述
const Aws::Vector<Aws::String>& GetAllowedHeaders() const	获取客户端允许携带的headers
const Aws::Vector<Aws::String>& GetAllowedMethods() const	获取允许的http方法
const Aws::Vector<Aws::String>& GetAllowedOrigins() const	获取允许的域名来源
const Aws::Vector<Aws::String>& GetExposeHeaders() const	获取响应中允许暴露的headers
const Aws::String& GetID() const	获取规则ID
int GetMaxAgeSeconds() const	获取有效时间（秒）

代码示例

```
#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();
        cout << "ERROR: " << endl
            << "Http code: " << (int)err.GetResponseCode() << endl
            << "Error Type:" << (int)err.GetErrorType() << endl
            << "Error Msg: " << err.GetMessage() << endl
            << "Exception Name: " << err.GetExceptionName() << endl;
    }
}

int main()
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    GetBucketCors(client, bucket);

    Aws::ShutdownAPI(options);
    return 0;
}
```

删除桶跨域访问配置

功能说明

删除 Bucket 的跨域资源访问配置。

方法原型

`DeleteBucketCorsOutcome DeleteBucketCors(const Model::DeleteBucketCorsRequest& request) const`

参数说明

- DeleteBucketCorsRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名

返回结果说明

- DeleteBucketCorsOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类

代码示例

```
#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);
    auto outcome = client.DeleteBucketCors(req);
    if (!outcome.IsSuccess())
    {
        auto err = outcome.GetError();
        cout << "ERROR: " << 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 << "successful delete bucket Cors: " << bucketName << "\n";
    }
    return;
}

int main()
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    DeleteBucketCors(client, bucket);
    Aws::ShutdownAPI(options);
    return 0;
}
```

设置桶版本控制状态

功能说明

启用或者暂停Bucket的版本控制功能。开启后只能暂停不能关闭。

方法原型

`PutBucketVersioningOutcome PutBucketVersioning(const PutBucketVersioningRequest& request) const`

参数说明

- PutBucketVersioningRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetVersioningConfiguration(const VersioningConfiguration& value)	是	设置配置

- VersioningConfiguration

设定参数方法	是否必选	描述
void SetStatus(const BucketVersioningStatus& value)	是	设置是否开启：BucketVersioningStatus::Enabled，开启；BucketVersioningStatus::Suspended，暂停。

返回结果说明

- PutBucketVersioningOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类

代码示例

```
#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) {
    // 设置请求参数
    PutBucketVersioningRequest request;
    request.SetBucket(bucket_name);
    VersioningConfiguration cfg;
    cfg.SetStatus(BucketVersioningStatus::Enabled);
    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();
        cout << "ERROR: " << 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 << "request success " << std::endl;
    }
    return;
}

int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    put_bucket_version(client, bucket);
    Aws::ShutdownAPI(options);
}
```

获取桶版本控制状态

功能说明

获得Bucket的版本控制配置。

方法原型

`GetBucketVersioningOutcome GetBucketVersioning(const GetBucketVersioningRequest& request) const`

参数说明

- GetBucketVersioningRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名

返回结果说明

- GetBucketVersioningOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
const GetBucketVersioningResult& GetResult()	获取结果

- GetBucketVersioningResult

获取结果方法	描述
const BucketVersioningStatus& GetStatus()	获取是否开启： BucketVersioningStatus::Enabled，开启； BucketVersioningStatus::Suspended，暂停。

代码示例

```
#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();
        cout << "ERROR: " << 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();
        std::cout <<(int)result.GetStatus() << std::endl;
        std::cout << "request success " << std::endl;
    }
    return;
}

int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    get_bucket_version(client, bucket);
    Aws::ShutdownAPI(options);
}
```

设置桶标签

功能说明

为指定的Bucket设置标签。一个Bucket最多设置50个标签。

方法原型

`PutBucketTaggingOutcome PutBucketTagging(const PutBucketTaggingRequest& request) const`

参数说明

- PutBucketTaggingRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetTagging(const Tagging& value)	是	设置标签集

- Tagging

设定参数方法	是否必选	描述
void SetTagSet(const Aws::Vector<Tag>& value)	是	设置标签集

- Tag

设定参数方法	是否必选	描述
void SetKey(const Aws::String& value)	是	设置标签key
void SetValue(const Aws::String& value)	是	设置标签value

返回结果说明

- PutBucketTaggingOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类

代码示例

```
#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();
        cout << "ERROR: " << 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 << "successful put bucket tagging " << endl;
    }

    return;
}
int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    put_bucket_tagging(client, bucket);
    Aws::ShutdownAPI(options);
}
```

获取桶标签

功能说明

返回桶的标签集。

方法原型

`GetBucketTaggingOutcome GetBucketTagging(const GetBucketTaggingRequest& request) const`

参数说明

- GetBucketTaggingRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名

返回结果说明

- GetBucketTaggingOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
const GetBucketTaggingResult& GetResult() const	获取请求结果

- GetBucketTaggingResult

获取结果方法	描述
const Aws::Vector<Tag>& GetTagSet() const	获取标签集

- Tag

设定参数方法	描述
const Aws::String& GetKey() const	获取标签key
const Aws::String& GetValue() const	获取标签value

代码示例

```
#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();
        cout << "ERROR: " << 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 << "bucket tagging:  " << 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[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    get_bucket_tagging(client, bucket);
    Aws::ShutdownAPI(options);
}
```

删除桶标签

功能说明

从指定的桶中删除整个标记集。

方法原型

`DeleteBucketTaggingOutcome DeleteBucketTagging(const DeleteBucketTaggingRequest& request
) const`

参数说明

- DeleteBucketTaggingRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名

返回结果说明

- DeleteObjectTaggingOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类

代码示例

```
#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();
        cout << "ERROR: " << 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 << "successful delete bucket tagging " << endl;
    }

    return;
}
int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    delete_bucket_tagging(client, bucket);
    Aws::ShutdownAPI(options);
}
```

设置桶日志转存配置

功能说明

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。

方法原型

`PutBucketLoggingOutcome PutBucketLogging (const PutBucketLoggingRequest& request) const`

参数说明

- PutBucketLoggingRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetBucketLoggingStatus(const BucketLoggingStatus& value)	是	设置日志转存配置

- BucketLoggingStatus

设定参数方法	是否必选	描述
void SetLoggingEnabled(const LoggingEnabled& value)	是	设置日志转存

- LoggingEnabled

设定参数方法	是否必选	描述
void SetTargetBucket(const Aws::String& value)	是	设置存储日志的桶
void SetTargetPrefix(const char* value)	是	设置日志目录的前缀
void SetTargetGrants(const Aws::Vector<TargetGrant>& value)	可选	设置权限授予

- TargetGrant

设定参数方法	是否必选	描述
void SetGrantee(const Grantee& value)	是	设置被授权用户
void SetPermission(const BucketLogsPermission& value)	是	设置权限： BucketLogsPermission::FULL_CONTROL，所有权限； BucketLogsPermission::READ，读权限； BucketLogsPermission::WRITE，写权限；

- Grantee

设定参数方法	是否必选	描述
void SetType(const Type& value)	是	设置被授权用户类型： Type::CanonicalUser，规范用户； Type::Group，IAM组。
void SetID(const Aws::String& value)	是（CanonicalUser）	设置ID
void SetURI(const Aws::String& value)	是（Group）	设置组URI
void SetDisplayName(const Aws::String& value)	可选	设置名字

返回结果说明

- PutBucketLoggingOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类

代码示例

```
#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.SetID("testid");

    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(bucket_name);
    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();
        cout << "ERROR: " << 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 << "successful put bucket logging: " << bucket_name <<
 "\n";
    }
    return;
}

int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    put_bucket_logging(client, bucket);
    Aws::ShutdownAPI(options);
}
```

获取桶日志转存配置

功能说明

获取存储桶的日志转存配置。

方法原型

`GetBucketLoggingOutcome GetBucketLogging(const GetBucketLoggingRequest& request) const`

参数说明

- GetBucketLoggingRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名

返回结果说明

- GetBucketLoggingOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
GetBucketLoggingResult& GetResult()	获取请求结果

- GetBucketLoggingResult

获取结果方法	描述
const LoggingEnabled& GetLoggingEnabled() const	获取日志转存配置

- LoggingEnabled

设定参数方法	描述
const Aws::String& GetTargetBucket() const	获取存储日志的桶名
const Aws::String& GetTargetPrefix() const	获取日志前缀
const Aws::Vector<TargetGrant>& GetTargetGrants() const	获取日志转存的目标授权信息

- TargetGrant

获取结果方法	描述
const Grantee& GetGrantee() const	获取授权用户信息
const BucketLogsPermission& GetPermission() const	获取权限类型： BucketLogsPermission::FULL_CONTROL，所有权限； BucketLogsPermission::READ，读权限； BucketLogsPermission::WRITE，写权限。

- Grantee

获取结果方法	描述
const Type& GetType() const	获取用户类型： Type::CanonicalUser，规范用户； Type::Group，IAM组。
const Aws::String& GetID() const	获取用户ID
const Aws::String& GetURI() const	获取用户组URI
const Aws::String& GetDisplayName() const	获取名字

代码示例

```
#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();
        cout << "ERROR: " << 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 << "successful get bucket logging: " << bucket_name << endl;
        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 id = grantee.GetID();
            cout << "GetID: " << id << endl;
        }
    }
    return;
}

int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);
```

```
    get_bucket_logging(client, bucket);
    Aws::ShutdownAPI(options);
}
```

设置服务端加密配置

功能说明

启用存储桶默认加密功能

方法原型

`PutBucketEncryptionOutcome PutBucketEncryption(const PutBucketEncryptionRequest& request) const`

参数说明

- PutBucketEncryptionRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetServerSideEncryptionConfiguration(const ServerSideEncryptionConfiguration& value)	是	设置加密配置

- ServerSideEncryptionConfiguration

设定参数方法	是否必选	描述
void SetRules(const Aws::Vector<ServerSideEncryptionRule>& value)	是	设置加密规则

- ServerSideEncryptionRule

设定参数方法	是否必选	描述
void SetApplyServerSideEncryptionByDefault(const ServerSideEncryptionByDefault& value)	是	设置加密方式

- ServerSideEncryptionByDefault

设定参数方法	是否必选	描述

void SetSSEAlgorithm(ServerSideEncryption& value)	是	设置加密算法：
		ServerSideEncryption::NOT_SET;
		ServerSideEncryption::AES256;
		ServerSideEncryption::aws_kms

void SetKMSMasterKeyID(const char* value)	可选（加密算法为kms时必选）	设置加密密钥，当选择"AES256"时，长度需为32位；当选择"aws:kms"时，需以cmkuuid:keyspec:userid的格式。

返回结果说明

- PutBucketEncryptionOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类

代码示例

```
#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::AES256);

    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();
        cout << "ERROR: " << 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 << "successful put bucket encryption: " << bucket_name
 << "\n";
    }
    return;
}

int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    put_bucket_encryption(client, bucket);
    Aws::ShutdownAPI(options);
}
```

获取服务端加密配置

功能说明

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

`GetBucketEncryptionOutcome PutBucketEncryption(const GetBucketEncryptionRequest& request) const`

参数说明

- GetBucketEncryptionRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名

返回结果说明

- GetBucketEncryptionOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
GetBucketEncryptionResult& GetResult()	获取请求结果

- GetBucketEncryptionResult

获取结果方法	描述
const ServerSideEncryptionConfiguration& GetServerSideEncryptionConfiguration() const	获取服务端加密配置

- ServerSideEncryptionConfiguration

获取结果方法	描述
const Aws::Vector<ServerSideEncryptionRule>& GetRules() const	获取服务端加密规则

- ServerSideEncryptionRule

获取结果方法	描述
const ServerSideEncryptionByDefault& GetApplyServerSideEncryptionByDefault() const	获取服务端加密规则

- ServerSideEncryptionByDefault

获取结果方法	描述
const ServerSideEncryption& GetSSEAlgorithm() const	获取服务端加密算法
const Aws::String& GetKMSMasterKeyID() const	获取密钥

代码示例

```
#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();
        cout << "ERROR: " << 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 << "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[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    get_bucket_encryption(client, bucket);
    Aws::ShutdownAPI(options);
}
```

删除服务端加密配置

功能说明

删除存储桶默认加密配置

方法原型

`DeleteBucketEncryptionOutcome DeleteBucketEncryption(const DeleteBucketEncryptionRequest& request) const`

参数说明

- DeleteBucketEncryptionRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名

返回结果说明

- DeleteBucketEncryptionOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类

代码示例

```
#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();
        cout << "ERROR: " << 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 << "successful delete bucket encryption: " << bucket_name <<
"\n";
    }
    return;
}

int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    delete_bucket_encryption(client, bucket);
    Aws::ShutdownAPI(options);
}
```

设置桶静态网站配置

功能说明

将桶设置成静态网站托管模式并设置跳转规则。

方法原型

PutBucketWebsiteOutcome PutBucketWebsite(const PutBucketWebsiteRequest& request)

参数说明

- PutBucketWebsiteRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetWebsiteConfiguration(const WebsiteConfiguration& value)	是	设置静态网站配置

- BucketWebsiteConfiguration，**ErrorDocument、IndexDocument、RoutingRules三项参数结合使用（不能全为空）并与RedirectAllRequestsTo互斥，当设置了这三个字段时，不能设置RedirectAllRequestsTo；反之，当设置了RedirectAllRequestsTo时，不能设置这三项。**

设定参数方法	是否必选	描述
void SetErrorDocument(const ErrorDocument& value)	可选	当4XX错误出现时使用的对象的名称。这个元素指定当错误出现时返回的页面。
void SetIndexDocument(const IndexDocument& value)	可选	设置索引文档(该字段被追加在对文件夹的请求的末尾（例如：参数设置为“index.html”，请求的是“samplebucket/images/”，返回的数据将是“samplebucket”桶内名为“images/index.html”的对象的内容）。
void SetRoutingRules(const Aws::Vector<RoutingRule>& value)	可选	设置重定向规则
void SetRedirectAllRequestsTo(const RedirectAllRequestsTo& value)	可选	设置所有请求重定向规则

- ErrorDocument

设定参数方法	是否必选	描述
void SetKey(const Aws::String& value)	必选	设置静态网站错误文档的key

- IndexDocument

设定参数方法	是否必选	描述
void SetSuffix(const Aws::String& value)	必选	设置静态网站索引文档名称

- RoutingRule

设定参数方法	是否必选	描述
void SetCondition(const Condition& value)	可选	设置重定向规则的匹配条件
void SetRedirect(const Redirect& value)	必选	重定向请求时的具体信息。

- Condition

设定参数方法	是否必选	描述
void SetKeyPrefixEquals(const Aws::String& value)	可选	如果对象名前缀等于这个值，那么重定向生效。
void SetHttpErrorCodeReturnedEquals(const Aws::String& value)	可选	当发生错误时，如果错误码等于这个值，那么重定向生效。（当跳转规则类型为镜像回源时，此项必须设置为404）

- Redirect

设定参数方法	是否必选	描述
void setType(const Aws::String& value)	可选	设置跳转的类型（不设置时为静态网站相关跳转，设置时则启用数据回源功能）： 空，为静态网站相关跳转 "MIRROR"，镜像回源； "REDIRECTION"，外部跳转，返回3XX请求，指定跳转到另一个地址；
void SetProtocol(const Protocol& value);	可选	设置重定向使用的协议，默认使用源请求的协议 Protocol::http，http协议； Protocol::https，https协议。
void SetHostName(const Aws::String& value);	必选	设置重定向主机名
void SetReplaceKeyPrefixWith(const Aws::String& value)	可选	指定重定向规则的具体重定向目标的对象键，替换方式为替换原始请求中所匹配到的前缀部分
void SetReplaceKeyWith(const Aws::String& value)	可选	指定重定向规则的具体重定向目标的对象键，替换方式为替换整个原始请求的对象键
void SetHttpRedirectCode(const Aws::String& value);	可选	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
void setMirrorFollowRedirect(bool value)	可选	默认false，只有Type设置为"MIRROR"时才生效，当镜像回源获取结果为3XX时，是否继续跳转到指定Location获取数据。例如发起镜像回源请求时，源站返回302，并且指定了location。如果此项设置为true，则ZOS会继续请求location对应的地址；如果为false，则ZOS会返回302，并透传location。

- RedirectAllRequestsTo

设定参数方法	是否必选	描述
void SetProtocol(const Protocol& value)	可选	设置重定向使用的协议，默认使用源请求的协议 Protocol::http，http协议； Protocol::https，https协议。
void SetHostName(const Aws::String& value);	必选	设置重定向主机名

返回结果说明

- PutBucketWebsiteOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类

代码示例

```
#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;


void PutWebsiteConfig(S3Client &client,const Aws::String& bucketName)
{
    IndexDocument index_doc;
    index_doc.SetSuffix("index.html");

    ErrorDocument error_doc;
    error_doc.SetKey("err.html");

    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;
    }
    else
    {
        auto err = outcome.GetError();
        cout << "ERROR: " << endl
            << "Http code: " << (int)err.GetResponseCode() << endl
            << "Error Type:" << (int)err.GetErrorType() << endl
            << "Error Msg: " << err.GetMessage() << endl
            << "Exception Name: " << err.GetExceptionName() << endl;
    }
}

int main()
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    PutWebsiteConfig(client, bucket);
    Aws::ShutdownAPI(options);
    return 0;
}
```

获取桶静态网站配置

功能说明

查询与存储桶关联的静态网站配置信息。

方法原型

`GetBucketWebsiteOutcome GetBucketWebsite(const GetBucketWebsiteRequest& request)`

参数说明

- GetBucketWebsiteRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value);	是	设置桶名

返回结果说明

- GetBucketWebsiteOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
GetBucketWebsiteResult& GetResult()	获取请求结果

- GetBucketWebsiteResult

获取结果方法	描述
ErrorDocument& GetErrorDocument()	获取错误文档配置
IndexDocument& GetIndexDocument()	获取索引文档
Aws::Vector<RoutingRule>& GetRoutingRules()	获取重定向规则配置
RedirectAllRequestsTo& GetRedirectAllRequestsTo()	获取对所有请求重定向的配置

- ErrorDocument

设定参数方法	描述
Aws::String& GetKey()	获取静态网站错误文档的key

- IndexDocument

设定参数方法	描述
Aws::String& GetSuffix()	获取静态网站索引文档名称

- RoutingRule

获取结果方法	描述
Condition& GetCondition()	获取重定向规则的匹配条件
Redirect& GetRedirect()	获取重定向规则配置信息

- Condition

获取结果方法	描述
Aws::String& GetKeyPrefixEquals()	获取前缀匹配条件
Aws::String& GetHttpErrorCodeReturnedEquals()	获取错误码匹配条件

- Redirect

获取结果方法	描述
Aws::String&getType()	获取跳转类型
Protocol& GetProtocol()	获取重定向协议配置
Aws::String& GetHostName()	获取重定向主机名
Aws::String& GetReplaceKeyPrefixWith()	获取重定向规则的具体重定向目标的对象键前缀
Aws::String& GetReplaceKeyWith()	获取重定向规则的具体重定向目标的对象键
Aws::String& GetHttpRedirectCode()	获取重定向规则的错误码匹配条件
bool getMirrorFollowRedirect()	获取是否继续跳转到指定Location获取数据

- RedirectAllRequestsTo

获取结果方法	描述
Protocol& GetProtocol()	获取前缀匹配条件
Aws::String& GetHostName()	获取错误码匹配条件

代码示例

```
#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);

    GetBucketWebsiteOutcome outcome = client.GetBucketWebsite(request);

    if (outcome.IsSuccess())
    {
        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();
        cout << "ERROR: " << endl
            << "Http code: " << (int)err.GetResponseCode() << endl
            << "Error Type:" << (int)err.GetErrorType() << endl
            << "Error Msg: " << err.GetMessage() << endl
            << "Exception Name: " << err.GetExceptionName() << endl;
    }
}

int main()
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    GetWebsiteConfigure(client, bucket);
    Aws::ShutdownAPI(options);
    return 0;
}
```

删除桶静态网站配置

功能说明

删除桶的静态网站配置。

方法原型

`DeleteBucketWebsiteOutcome DeleteBucketWebsite(const DeleteBucketWebsiteRequest& request)`

参数说明

- DeleteBucketWebsiteRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名

返回结果说明

- DeleteBucketWebsiteOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类

代码示例

```
#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)
{
    DeleteBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    DeleteBucketWebsiteOutcome outcome = client.DeleteBucketWebsite(request);

    if (!outcome.IsSuccess())
    {
        auto err = outcome.GetError();
        cout << "ERROR: " << 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 << "Success: Delete website configuration for bucket '"
            << bucketName << "'." << std::endl;
    }
}

int main()
```

```
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    DeleteBucketWebsiteConfig(client, bucket);
    Aws::ShutdownAPI(options);
    return 0;
}
```

对象操作

下载对象

功能说明

将一个文件（Object）下载至本地。

方法原型

`GetObjectOutcome GetObject(const GetObjectRequest &request) const`

参数说明

- GetObjectRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetKey(const Aws::String& value)	是	设置对象名
void SetIfMatch(const Aws::String& value)	可选	当对象的ETag和值相同时才下载
void SetIfNoneMatch(const Aws::String& value)	可选	当对象的ETag和值不同时才下载
void SetIfModifiedSince(const Aws::Utils::DateTime& value)	可选	只有指定时间之后有修改记录的对象才会返回
void SetIfUnmodifiedSince(const Aws::Utils::DateTime& value)	可选	只有指定时间之后没有修改记录的对象才会返回
void SetRange(const Aws::String& value)	可选	指定下载对象的字节范围，"bytes=0-9"表示下载开头的10个字节
void SetVersionId(const Aws::String& value)	可选	指定版本号

返回结果说明

- GetObjectOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
GetObjectResult& GetResultWithOwnership()	获取结果

- GetObjectResult

获取结果方法	描述
Aws::IOStream& GetBody()	获取对象内容
const Aws::Map<Aws::String, Aws::String>& GetMetadata() const	获取对象元数据
const Aws::Utils::DateTime& GetLastModified() const	获取对象创建时间
const Aws::String& GetETag() const	获取对象ETag
const Aws::String& GetVersionId()	获取对象版本号
bool GetDeleteMarker() const	获取的对象是否为删除标记
const ObjectLockMode& GetObjectLockMode() const	获取对象的合规保留模式
const StorageClass& GetStorageClass() const	获取对象存储类型

代码示例

```
#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;
using namespace Aws::Client;
using namespace Aws::S3;
using namespace Aws::S3::Model;
using namespace Aws::Auth;
using namespace std;

void GetObject(S3Client &client, const Aws::String &bucket, Aws::String
objectName, Aws::String fname)
{
    // 设置请求
    GetObjectRequest request;

    request.SetBucket(bucket);
    request.SetKey(objectName);

    auto outcome = client.GetObject(request);

    if (!outcome.IsSuccess())
    {
        auto err = outcome.GetError();
        cout << "ERROR: " << 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 << "GetObject request success " << std::endl;
        auto result = outcome.GetResultWithOwnership();
        auto &file = result.GetBody();
        Aws::Map<Aws::String, Aws::String> meta = result.GetMetadata();

        ofstream outFile;
        outFile.open(fname, ios::out | ios::binary);
        string bytes(5 * 1024 * 1024, '\0');
        while (file.read(&bytes[0], sizeof(bytes)).gcount() > 0) {
            outFile.write(&bytes[0], file.gcount());
        }
        outFile.close();
    }
}

int main(int argc, char *argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";
    String savePath = "<your-savePath>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    GetObject(client, bucket, key, savePath);
    Aws::ShutdownAPI(options);
}
```

上传对象

功能说明

将一个文件（Object）上传至指定 Bucket。

方法原型

`PutObjectOutcome PutObject(const PutObjectRequest &request) const`

参数说明

- PutObjectRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetKey(const Aws::String& value)	是	设置对象名
void SetACL(const ObjectCannedACL& value)	可选	设置对象ACL，可选项： ObjectCannedACL::private_，私有（默认）； ObjectCannedACL::public_read，公共读，但只有对象所有者可以写入和更改ACL； ObjectCannedACL::public_read_write，公共读写，但只有对象所有者可以写入和更改ACL； ObjectCannedACL::bucket_owner_read，对象所有者和存储桶所有者可以读取该对象，只有对象所有者可以写入和更改ACL； ObjectCannedACL::bucket_owner_full_control，对象所有者和存储桶所有者可以读取和写入该对象以及更改ACL。
void SetMetadata(const Aws::Map<Aws::String, Aws::String>& value)	可选	设置对象元数据
void SetObjectLockMode(const ObjectLockMode& value)	可选	设置合规保留模式： ObjectLockRetentionMode::COMPLIANCE，设置后不允许修改； ObjectLockRetentionMode::GOVERNANCE，设置后允许修改。
void SetObjectLockRetainUntilDate(const Aws::Utils::DateTime& value)	可选	设置保留到期时间
void SetTagging(const Aws::String& value)	可选	设置对象标签，格式："key=value"
void SetStorageClass(const StorageClass& value)	可选	设置对象存储类型，可选项： "STANDARD"，标准存储； "STANDARD_IA"，低频存储； "GLACIER"，归档存储。

返回结果说明

- PutObjectOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
PutObjectResult &GetResult()	获取结果

- PutObjectResult

获取结果方法	描述
const Aws::String& GetETag() const	获取对象ETag

代码示例

```
#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;
using namespace Aws::Client;
using namespace Aws::S3;
using namespace Aws::S3::Model;
using namespace Aws::Auth;
using namespace std;

void PutObject(S3Client &client, const Aws::String &bucket, Aws::String
objectName, Aws::String fname)
{
    // 设置请求
    PutObjectRequest request;
    std::shared_ptr<Aws::IOStream> inputData =
        Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
            fname.c_str(),
            std::ios_base::in | std::ios_base::binary);

    request.SetBody(inputData);
    request.SetBucket(bucket);
    request.SetKey(objectName);

    auto outcome = client.PutObject(request);

    if (!outcome.IsSuccess())
    {
        auto err = outcome.GetError();
        cout << "ERROR: " << 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;
    }
}

int main(int argc, char *argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";
    String filePath = "<your-localFilePath>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    PutObject(client, bucket, key, filePath);
    Aws::ShutdownAPI(options);
}
```

判断对象是否存在

功能说明

用于判断一个对象是否存在，同时可用于获取对象元数据。

方法原型

`HeadObjectOutcome HeadObject(const HeadObejctRequest& request) const`

参数说明

- HeadObejctRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetKey(const Aws::String& value)	是	设置对象名
void SetIfMatch(const Aws::String& value)	可选	当对象的ETag和值相同时才下载
void SetIfNoneMatch(const Aws::String& value)	可选	当对象的ETag和值不同时才下载
void SetIfModifiedSince(const Aws::Utils::DateTime& value)	可选	只有指定时间之后有修改记录的对象才会返回
void SetIfUnmodifiedSince(const Aws::Utils::DateTime& value)	可选	只有指定时间之后没有修改记录的对象才会返回
void SetRange(const Aws::String& value)	可选	指定下载对象的字节范围，"0-9"表示下载开头的10个字节
void SetVersionId(const Aws::String& value)	可选	指定版本号

返回结果说明

- HeadObjectOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
const HeadObjectResult& GetResult()	获取结果

- HeadObjectResult

获取结果方法	描述
Aws::Utils::DateTime& GetLastModified()	获取最后修改时间
const Aws::String& GetETag() const	获取对象ETag
Aws::String& GetVersionId()	获取版本号
Aws::Map<Aws::String, Aws::String>& GetMetadata()	获取对象自定义元数据
StorageClass& GetStorageClass()	获取对象存储类型
Aws::String& GetContentType()	获取对象类型
long long GetContentLength()	获取对象大小

代码示例

```cpp
#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();
        if ((int)err.GetResponseCode() == 404) {
            cout << "object: " << object_name << " is not exist." << endl;
        }
        else {
            cout << "ERROR: " << 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_name << " is exist." << endl;
    }
    return;
}

int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    head_object(client, bucket, key);
    Aws::ShutdownAPI(options);
}
```

删除对象

功能说明

将一个文件（Object）删除。

方法原型

`DeleteObjectOutcome DeleteObject(const DeleteObjectRequest &request) const`

参数说明

- DeleteObjectRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetKey(const Aws::String& value)	是	设置对象名
void SetBypassGovernanceRetention(bool value)	可选	设置是否忽视合规保留的Governance模式
void SetVersionId(const Aws::String& value)	可选	指定版本号

返回结果说明

- DeleteObjectOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类

代码示例

```
#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;
using namespace Aws::Client;
using namespace Aws::S3;
using namespace Aws::S3::Model;
using namespace Aws::Auth;
using namespace std;

void DeleteObject(S3Client &client, const Aws::String &bucket, const Aws::String
&object)
{
    // 设置请求
    DeleteObjectRequest request;
    request.SetBucket(bucket);
    request.SetKey(object);

    auto outcome = client.DeleteObject(request);

    if (!outcome.IsSuccess())
    {
        auto err = outcome.GetError();
        cout << "ERROR: " << 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[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    DeleteObject(client, bucket, key);
    Aws::ShutdownAPI(options);
}
```

批量删除对象

功能说明

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

`DeleteObjectsOutcome DeleteObjects(const DeleteObjectsRequest &request)`

参数说明

- DeleteObjectsRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetDelete(const Aws::S3::Model::Delete& value)	是	设置要删除的对象集合
void setBypassGovernanceRetention(boolean bypassGovernanceRetention)	可选	设置是否绕过GOVERNANCE对象锁的限制

- Delete

设定参数方法	是否必选	描述
void SetObjects(const Aws::Vector<ObjectIdentifier>& value)	是	设置对象集合
Delete& AddObjects(const ObjectIdentifier& value)	可选	添加对象
void SetQuiet(bool value)	可选	是否开启quiet模式

- ObjectIdentifier

设定参数方法	是否必选	描述
void SetKey(const Aws::String& value)	是	设置对象名
void SetVersionId(const Aws::String& value)	可选	设置版本号

返回结果说明

- DeleteObjectsOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
DeleteObjectResult &GetResult()	获取请求结果

- DeleteObjectsResult

获取结果方法	描述
Aws::Vector<DeletedObject>& GetDeleted()	获取已删除的对象列表

- DeletedObject

获取结果方法	描述
Aws::String& GetKey()	获取对象名
Aws::String& GetVersionId()	获取对象版本号

代码示例

```
#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;
using namespace Aws::Client;
using namespace Aws::S3;
using namespace Aws::S3::Model;
using namespace Aws::Auth;
using namespace std;

void DeleteObject(S3Client &client, const Aws::String &bucket, const Aws::String
&key)
{
    // 设置请求
    DeleteObjectsRequest request;
    Aws::S3::Model::Delete delete_objects;

    Aws::Vector<ObjectIdentifier> idnetfiers;
    ObjectIdentifier ident;
    ident.SetKey(key);
    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: " << 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[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    DeleteObject(client, bucket, key);
    Aws::ShutdownAPI(options);
}
```

列出对象

功能说明

列出该 Bucekt 下部分或者所有Object。

方法原型

`ListObjectsOutcome ListObjects(const ListObjectsRequest &request) const`

参数说明

- ListObjectsRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetPrefix(const Aws::String& value)	可选	只列出特定前缀的对象
void SetMaxKeys(int value)	可选	设置一次返回的对象数，默认1000
void SetMarker(const Aws::String& value)	可选	列举对象的起始位置，返回的对象列表将是对象名按照字典序排序后该参数以后的所有对象
void SetDelimiter(const Aws::String& value)	可选	对象名按照此标识符进行分组。通常与prefix参数搭配使用，如果指定了prefix，从prefix到第一次出现delimiter间具有相同字符串的对象名会被分成一组

返回结果说明

- ListObjectsOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
ListObjectsResult& GetResult()	获取返回结果

- ListObjectsResult

获取结果方法	描述
const Aws::Vector<Object>& GetContents()	获取对象集合
Aws::String& GetNextMarker()	下次列举对象请求的起始位置。如果本次没有返回全部结果，响应请求中将包含该元素，用于标明接下来请求的Marker值
bool GetIsTruncated()	获取是否返回了所有满足要求的Key
Aws::String& GetName()	获取桶名

- Object

获取结果方法	描述
const Aws::String& GetETag()	获取ETag
const Aws::String& GetKey()	获取对象名
const Aws::Utils::DateTime& GetLastModified()	获取修改时间
const Owner& GetOwner()	获取所有者
long long GetSize()	获取大小
const ObjectStorageClass& GetStorageClass()	获取存储类型
const Aws::String& GetType()	获取对象类型： "Appendable"，可追加写对象； "Symlink"，软链接对象； "Normal"，普通对象

代码示例

```
#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;
using namespace Aws::Client;
using namespace Aws::S3;
using namespace Aws::S3::Model;
using namespace Aws::Auth;
using namespace std;

void ListObjects(S3Client &client, const Aws::String &bucket)
{
    // 设置请求
    ListObjectsRequest request;
    request.SetBucket(bucket);

    String marker = "";
    bool istruncated = true;
    // 循环请求直到获取全部对象
    while (istruncated) {
        request.SetMarker(marker);
        auto outcome = client.ListObjects(request);

        if (!outcome.IsSuccess())
        {
            auto err = outcome.GetError();
            cout << "ERROR: " << 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() << "\tETag: " << content-
>GetETag() << endl;
            }
        }
        marker = outcome.GetResult().GetNextMarker();
        istruncated = outcome.GetResult().GetIsTruncated();
    }
}

int main(int argc, char *argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    ListObjects(client, bucket);
    Aws::ShutdownAPI(options);
}
```

列出对象V2

功能说明

列出该 Bucekt 下部分或者所有Object。

方法原型

`ListObjectsV2Outcome ListObjectsV2(const ListObjectsV2Request& request) const`

参数说明

- ListObjectsV2Request

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetPrefix(const Aws::String& value)	可选	只列出特定前缀的对象
void SetMaxKeys(int value)	可选	设置一次返回的对象数，默认1000
void SetContinuationToken(const Aws::String& value)	可选	列举对象的起始位置，返回的对象列表将是对象名按照字典序排序后该参数以后的所有对象
void SetDelimiter(const Aws::String& value)	可选	对象名按照此标识符进行分组。通常与prefix参数搭配使用，如果指定了prefix，从prefix到第一次出现delimiter间具有相同字符串的对象名会被分成一组

返回结果说明

- ListObjectsV2Outcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
ListObjectsV2Result& GetResult()	获取返回结果

- ListObjectsV2Result

获取结果方法	描述
const Aws::Vector<Object>& GetContents()	获取对象集合
Aws::String& GetNextContinuationToken()	下次列举对象请求的起始位置。如果本次没有返回全部结果，响应请求中将包含该元素，用于标明接下来请求的token值
bool GetIsTruncated()	获取是否返回了所有满足要求的Key
Aws::String& GetName()	获取桶名

- Object

获取结果方法	描述
const Aws::String& GetETag()	获取ETag
const Aws::String& GetKey()	获取对象名
const Aws::Utils::DateTime& GetLastModified()	获取修改时间
const Owner& GetOwner()	获取所有者
long long GetSize()	获取大小
const ObjectStorageClass& GetStorageClass()	获取存储类型
const Aws::String& GetType()	获取对象类型： "Appendable"，可追加写对象； "Symlink"，软链接对象； "Normal"，普通对象

代码示例

```
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/ListObjectsV2Request.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 ListObjectsV2(S3Client &client, const Aws::String &bucket)
{
    // 设置请求
    ListObjectsV2Request request;
    request.SetBucket(bucket);

    String token = "";
    bool istruncated = true;
    while (istruncated) {
        request.SetContinuationToken(token);
        auto outcome = client.ListObjectsV2(request);

        if (!outcome.IsSuccess())
        {
            auto err = outcome.GetError();
            cout << "ERROR: " << 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() << "\tETag: " << content-
>GetETag() << endl;
            }
        }
        token = outcome.GetResult().GetNextContinuationToken();
        istruncated = outcome.GetResult().GetIsTruncated();
    }
}

int main(int argc, char *argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    ListObjectsV2(client, bucket);
    Aws::ShutdownAPI(options);
}
```

列出版本

功能说明

列出该 Bucekt 下部分或者所有Object的版本。

方法原型

`ListObjectVersionsOutcome ListObjectVersions(const ListObjectVersionsRequest& request) const`

参数说明

- ListObjectVersionsRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetPrefix(const Aws::String& value)	可选	只列出特定前缀的对象
void SetMaxKeys(int value)	可选	设置一次返回的对象数
void SetKeyMarker(const Aws::String& value)	可选	列举多版本对象的起始位置，返回的对象列表将是对象名按照字典序排序后该参数以后的所有对象
void SetVersionIdMarker(const Aws::String& value)	可选	与keyMarker配合使用，返回的对象列表将是对象名和版本号按照字典序排序后该参数以后的所有对象
void SetDelimiter(const Aws::String& value)	可选	对象名按照此标识符进行分组。通常与prefix参数搭配使用，如果指定了prefix，从prefix到第一次出现delimiter间具有相同字符串的对象名会被分成一组

返回结果说明

- ListObjectVersionsOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
ListObjectVersionsResult& GetResult()	获取返回结果

- ListObjectVersionsResult

获取结果方法	描述
Aws::Vector<ObjectVersion>& GetVersions()	获取版本集合
Aws::Vector<DeleteMarkerEntry>& GetDeleteMarkers()	获取删除标记集合
Aws::String& GetNextKeyMarker()	下次列举对象请求的起始位置。如果本次没有返回全部结果，响应请求中将包含该元素，用于标明接下来请求的keyMarker值
Aws::String& GetNextVersionIdMarker()	下次列举多版本对象请求的起始位置，与nextKeyMarker配合使用。如果本次没有返回全部结果，响应请求中将包含该元素，用于标明接下来请求的versionIdMarker值。
bool GetIsTruncated()	获取是否返回了所有满足要求的Key
Aws::String& GetName()	获取桶名

- ObjectVersion

获取结果方法	描述
const Aws::String& GetETag()	获取ETag
const Aws::String& GetKey()	获取对象名
const Aws::Utils::DateTime& GetLastModified()	获取修改时间
const Owner& GetOwner()	获取所有者
long long GetSize()	获取大小
const ObjectStorageClass& GetStorageClass()	获取存储类型
Aws::String& GetVersionId()	获取版本号
const Aws::String& GetType()	获取对象类型： "Appendable"，可追加写对象； "Symlink"，软链接对象； "Normal"，普通对象

- DeleteMarkerEntry

获取结果方法	描述
const Aws::String& GetETag()	获取ETag
const Aws::String& GetKey()	获取对象名
const Aws::Utils::DateTime& GetLastModified()	获取修改时间
const Owner& GetOwner()	获取所有者
long long GetSize()	获取大小
const ObjectStorageClass& GetStorageClass()	获取存储类型
Aws::String& GetVersionId()	获取版本号

代码示例

```
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/ListObjectVersionsRequest.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 ListObjectVersions(S3Client &client, const Aws::String &bucket)
{
    // 设置请求
    ListObjectVersionsRequest request;
    request.SetBucket(bucket);

    String keyMarker = "";
    String versionIdMarker = "";
    bool istruncated = true;
    while (istruncated) {
        request.SetKeyMarker(keyMarker);
        request.SetVersionIdMarker(versionIdMarker);
        auto outcome = client.ListObjectVersions(request);

        if (!outcome.IsSuccess())
        {
            auto err = outcome.GetError();
            cout << "ERROR: " << 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.GetVersions();
            cout << "Bucket:" << result.GetName() << endl;

            for (auto content = contents.begin(); content != contents.end();
++content)
            {
                cout << "\tKey: " << content->GetKey() << "\tVersionId: " <<
content->GetVersionId() << endl;
            }
        }
        keyMarker = outcome.GetResult().GetNextKeyMarker();
        versionIdMarker = outcome.GetResult().GetNextVersionIdMarker();
        istruncated = outcome.GetResult().GetIsTruncated();
    }
}

int main(int argc, char *argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    ListObjectVersions(client, bucket);
    Aws::ShutdownAPI(options);
}
```

设置对象ACL

功能说明

设置对象的ACL，控制对对象的访问权限。该操作需要用户具有WRITE_ACP权限。

设置方法有三种：

1. BucketCannedACL，整体的权限设置；
2. AccessControlPolicy，根据给出的配置，具体的给某些用户授予某些权限；
3. Grant方式，直接赋予某个用户某种权限。

方法原型

`PutObjectAclOutcome S3Client::PutObjectAcl(const PutObjectAclRequest& request) const`

参数说明

- PutObjectAclRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetKey(const Aws::String& value)	是	设置对象名
void SetVersionId(const Aws::String& value)	可选	设置版本号
void SetACL(const ObjectCannedACL& value)	可选	设置ACL（方法1），可选项：ObjectCannedACL::private_，私有（默认）；ObjectCannedACL::public_read，公共读，但只有对象所有者可以写入和更改ACL；ObjectCannedACL::public_read_write，公共读写，但只有对象所有者可以写入和更改ACL；ObjectCannedACL::bucket_owner_read，对象所有者和存储桶所有者可以读取该对象，只有对象所有者可以写入和更改ACL；ObjectCannedACL::bucket_owner_full_control，对象所有者和存储桶所有者可以读取和写入该对象以及更改ACL。
void SetAccessControlPolicy(const AccessControlPolicy& value)	两种方法选一种	设置ACL（方法2）
void SetGrantFullControl(const Aws::String& value)	三种方法选一种	设置ACL（方法3），给与用户所有权限，格式："id=xxx, uri=xxx, emailAddress=xxx"，只给出id即可。
void SetGrantRead(const Aws::String& value)	三种方法选一种	设置ACL（方法3），给与用户读权限，格式："id=xxx, uri=xxx, emailAddress=xxx"，只给出id即可。
void SetGrantReadACP(const Aws::String& value)	三种方法选一种	设置ACL（方法3），给与用户读ACL权限，格式："id=xxx, uri=xxx, emailAddress=xxx"，只给出id即可。
void SetGrantWrite(const Aws::String& value)	三种方法选一种	设置ACL（方法3），给与用户写权限，格式："id=xxx, uri=xxx, emailAddress=xxx"，只给出id即可。
void SetGrantWriteACP(const Aws::String& value)	三种方法选一种	设置ACL（方法3），给与用户写ACL权限，格式："id=xxx, uri=xxx, emailAddress=xxx"，只给出id即可。

- AccessControlPolicy

设定参数方法	是否必选	描述
void SetOwner(const Owner& value)	是	设置桶所有者
void SetGrants(const Aws::Vector<Grant>& value)	是	设置授权列表

- Owner

设定参数方法	是否必选	描述
void SetID(const Aws::String& value)	是	设置ID
void SetDisplayName(const Aws::String& value)	否	设置名字

- Grant

设定参数方法	是否必选	描述
void SetGrantee(const Grantee& value)	是	设置被授权用户
void SetPermission(const Permission& value)	是	设置权限： Permission::FULL_CONTROL，所有权限； Permission::READ，读权限； Permission::READ_ACP，读ACL权限； Permission::WRITE，写权限； Permission::WRITE_ACP，写ACL权限。

- Grantee

设定参数方法	是否必选	描述
void SetType(const Type& value)	是	设置被授权用户类型： Type::CanonicalUser，规范用户； Type::Group，IAM组。
void SetID(const Aws::String& value)	是（CanonicalUser）	设置ID
void SetURI(const Aws::String& value)	是（Group）	设置组URI

返回结果说明

- PutObjectAclOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类

代码示例

```
#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)
{
    PutObjectAclRequest request;
    request.SetBucket(bucket_name);
    request.SetKey(key);

    // 第一种方式
    request.SetACL(ObjectCannedACL::public_read);

    // 第二种方式
    //Owner owner;
    //owner.SetID("testid");
    //
    //Grantee grantee;
    //grantee.SetType(Type::CanonicalUser);
    //grantee.SetID("testid");
    //grantee.SetEmailAddress("abc@ctyun.cn");
    //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);

    // 第三种方式
    //request.SetGrantFullControl("id=xxx");

    auto outcome = client.PutObjectAcl(request);
    if (!outcome.IsSuccess())
    {
        auto err = outcome.GetError();
        cout << "ERROR: " << 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 << "successful" << endl;
    }
    return;
}

int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    put_object_acl(client, bucket, key);
    Aws::ShutdownAPI(options);
}
```

获取对象ACL

功能说明

获取指定Object的 ACL。该操作需要READ_ACP权限。

方法原型

`GetObjectAclOutcome S3Client::GetObjectAcl(const GetObjectAclRequest& request) const`

参数说明

- GetObjectAclRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetKey(const Aws::String& value)	是	设置对象名
void SetVersionId(const Aws::String& value)	可选	设置版本号

返回结果说明

- GetObjectAclOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
const GetObjectAclResult& GetResult() const	获取请求结果

- GetObjectAclResult

获取结果方法	描述
const Owner& GetOwner() const	获取桶Owner
const Aws::Vector<Grant>& GetGrants() const	获取授权列表

- Owner

设定参数方法	描述
const Aws::String& GetID() const	获取ID
const Aws::String& GetDisplayName() const	获取名字

- Grant

获取结果方法	描述
const Grantee& GetGrantee() const	获取授权用户信息
const Permission& GetPermission()	获取权限类型： Permission::FULL_CONTROL，所有权限； Permission::READ，读权限； Permission::READ_ACP，读ACL权限； Permission::WRITE，写权限； Permission::WRITE_ACP，写ACL权限。

- Grantee

获取结果方法	描述
const Type& GetType() const	获取用户类型： Type::CanonicalUser，规范用户； Type::Group，IAM组。
const Aws::String& GetID() const	获取用户ID
const Aws::String& GetURI() const	获取用户组URI

代码示例

```
#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)
{
    GetObjectAclRequest request;
    request.SetBucket(bucket_name);
    request.SetKey(key);

    auto outcome = client.GetObjectAcl(request);
    if (!outcome.IsSuccess())
    {
        auto err = outcome.GetError();
        cout << "ERROR: " << 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 << "successful : " << endl;
        auto result = outcome.GetResult();
        auto owner = result.GetOwner();
        cout << "Owner ID = " << owner.GetID() << endl;
        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();
            cout << "Type = " << GetNameForType(type) << endl;
            if (type == Type::Group) {
                auto uri = grantee.GetURI();
                cout << "URI = " << uri << endl;
            } else {
                auto id = grantee.GetID();
                auto display_name = grantee.GetDisplayName();
                auto email = grantee.GetEmailAddress();
                cout << "ID = " << id << endl;
                cout << "DisplayName = " << display_name << endl;
                cout << "Email = " << email << endl;
            }

            auto permission = it->GetPermission();
            cout << "Permission = " << GetNameForPermission(permission) << endl;
        }
    }
    return;
}

int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    get_object_acl(client, bucket, key);
    Aws::ShutdownAPI(options);
}
```

设置对象标签

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，key最大128字节，value最大256字节，value可以为空。

方法原型

`PutObjectTaggingOutcome S3Client::PutObjectTagging(const PutObjectTaggingRequest& request) const`

参数说明

- PutObjectTaggingRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetKey(const Aws::String& value)	是	设置对象名
void SetVersionId(const Aws::String& value)	可选	设置版本号
void SetTagging(const Tagging& value)	是	设置标签

- Tagging

设定参数方法	是否必选	描述
void SetTagSet(const Aws::Vector<Tag>& value)	是	设置标签集

- Tag

设定参数方法	是否必选	描述
void SetKey(const Aws::String& value)	是	设置标签key
void SetValue(const Aws::String& value)	是	设置标签value

返回结果说明

- PutObjectTaggingOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类

代码示例

```
#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 )
{
    PutObjectTaggingRequest request;
    request.SetBucket(bucket_name);
    request.SetKey(object_key);

    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();
        cout << "ERROR: " << 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 << "successful" << endl;
    }

    return;
}

int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    put_object_tagging(client, bucket, key);
    Aws::ShutdownAPI(options);
}
```

获取对象标签

功能说明

返回对象的标签集。

方法原型

`GetObjectTaggingOutcome S3Client::GetObjectTagging(const GetObjectTaggingRequest& request) const`

参数说明

- GetObjectTaggingRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetKey(const Aws::String& value)	是	设置对象名
void SetVersionId(const Aws::String& value)	可选	设置版本号

返回结果说明

- GetObjectTaggingOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
const GetObjectTaggingResult& GetResult() const	获取请求结果

- GetObjectTaggingResult

获取结果方法	描述
const Aws::Vector<Tag>& GetTagSet() const	获取标签集

- Tag

设定参数方法	描述
const Aws::String& GetKey() const	获取标签key
const Aws::String& GetValue() const	获取标签value

代码示例

```
#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)
{
    GetObjectTaggingRequest request;
    request.SetBucket(bucket_name);
    request.SetKey(object_key);

    auto outcome = client.GetObjectTagging(request);
    //处理请求结果
    if (!outcome.IsSuccess())
    {
        auto err = outcome.GetError();
        cout << "ERROR: " << 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 << "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[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    get_object_tagging(client, bucket, key);
    Aws::ShutdownAPI(options);
}
```

删除对象标签

功能说明

从指定的对象中删除整个标记集。

方法原型

`DeleteObjectTaggingOutcome S3Client::DeleteObjectTagging(const DeleteObjectTaggingRequest& request) const`

参数说明

- DeleteObjectTagging

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetKey(const Aws::String& value)	是	设置对象名
void SetVersionId(const Aws::String& value)	可选	设置版本号

返回结果说明

- DeleteObjectTaggingOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类

代码示例

```
#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)
{
    DeleteObjectTaggingRequest request;
    request.SetBucket(bucket_name);
    request.SetKey(object_key);

    auto outcome = client.DeleteObjectTagging(request);
    //处理请求结果
    if (!outcome.IsSuccess())
    {
        auto err = outcome.GetError();
        cout << "ERROR: " << 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 << "successful delete tagging" << endl;
    }

    return;
}
int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    delete_object_tagging(client, bucket, key);
    Aws::ShutdownAPI(options);
}
```

设置对象合规保留配置

功能说明

在指定的对象上增加合规保留配置，桶在创建时需开启对象锁。

方法原型

`PutObjectRetentionOutcome PutObjectRetention (const PutObjectRetentionRequest& request) const`

参数说明

- PutObjectRetentionRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetKey(const Aws::String& value)	是	设置对象名
void SetVersionId(const Aws::String& value)	可选	设置版本号
void SetRetention(const ObjectLockRetention& value)	可选	设置合规保留规则
void SetBypassGovernanceRetention(bool value)	可选	设置是否忽视Governance模式

- ObjectLockRetention

设定参数方法	是否必选	描述
void SetRetainUntilDate(const Aws::Utils::DateTime& value)	是	设置到期时间
void SetMode(const ObjectLockRetentionMode& value)	是	设置模式：ObjectLockRetentionMode::COMPLIANCE，设置后不允许修改；ObjectLockRetentionMode::GOVERNANCE，设置后允许修改。

返回结果说明

- PutObjectRetentionOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类

代码示例

```
#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) {
    // 设置请求参数
    PutObjectRetentionRequest request;
    request.SetBucket(bucket_name);
    request.SetKey(object_name);

    Aws::Utils::DateTime date_time;
    date_time = "2024-03-06T16:45:19Z";

    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();
        cout << "ERROR: " << 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 << "successful put object retention: " << object_name << endl;
    }
    return;
}

int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    put_object_retention(client, bucket, key);
    Aws::ShutdownAPI(options);
}
```

获取对象合规保留配置

功能说明

获取对象的合规保留配置。

方法原型

`GetObjectRetentionOutcome GetObjectRetention (const GetObjectRetentionRequest& request) const`

参数说明

- GetObjectRetentionRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetKey(const Aws::String& value)	是	设置对象名
void SetVersionId(const Aws::String& value)	可选	设置版本号

返回结果说明

- GetObjectRetentionOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
GetObjectRetentionResult& GetResult()	获取结果

- GetObjectRetentionResult

获取结果方法	描述
const ObjectLockRetention& GetRetention() const	获取合规保留配置

- ObjectLockRetention

获取结果方法	描述
const ObjectLockRetentionMode& GetMode() const	获取保留模式
const Aws::Utils::DateTime& GetRetainUntilDate() const	获取到期时间

代码示例

```
#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) {
    // 设置请求参数
    GetObjectRetentionRequest request;
    request.SetBucket(bucket_name);
    request.SetKey(object_name);

    // 发出请求
    GetObjectRetentionOutcome outcome = client.GetObjectRetention(request);

    //处理请求结果
    if (!outcome.IsSuccess())
    {
        auto err = outcome.GetError();
        cout << "ERROR: " << 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 << "successful get object retention: " << object_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[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    get_object_retention(client, bucket, key);
    Aws::ShutdownAPI(options);
}
```

设置对象依法保留配置

功能说明

在指定的对象上增加合规保留配置，桶在创建时需开启对象锁。该配置仅可开启或关闭，没有时限的设置，如果同时设置了合规保留配置，则以合规保留配置为准，**建议使用合规保留。**

方法原型

`PutObjectLegalHoldOutcome PutObjectLegalHold(const PutObjectLegalHoldRequest& request)`

参数说明

- PutObjectLegalHoldRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetKey(const Aws::String& value)	是	设置对象名
void SetVersionId(const Aws::String& value)	可选	设置版本号
void SetLegalHold(const ObjectLockLegalHold& value)	可选	设置依法保留配置

- ObjectLockLegalHold

设定参数方法	是否必选	描述
void SetStatus(ObjectLockLegalHoldStatus& value)	是	设置是否开启：ObjectLockLegalHoldStatus::OFF, 关闭 ObjectLockLegalHoldStatus::ON, 开启

返回结果说明

- PutObjectLegalHoldOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类

代码示例

```
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/PutObjectLegalHoldRequest.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_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
    PutObjectLegalHoldRequest request;
    request.SetBucket(bucket_name);
    request.SetKey(object_name);
    request.SetVersionId(object_version);

    ObjectLockLegalHold stat;
    stat.SetStatus(ObjectLockLegalHoldStatus::OFF);
    request.SetLegalHold(stat);

    // 发出请求
    PutObjectLegalHoldOutcome outcome = client.PutObjectLegalHold(request);

    //处理请求结果
    if (!outcome.IsSuccess())
    {
        auto err = outcome.GetError();
        cout << "ERROR: " << 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 << "successful Put object legal hold: " << object_name << endl;
    }
    return;
}

int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

    get_object_legal_hold(client, bucket, key, "<your-versionId>");
    Aws::ShutdownAPI(options);
}
```

获取对象依法保留配置

功能说明

获取对象的依法保留配置。

方法原型

`GetObjectLegalHoldOutCome GetObjectLegalHold(const GetObjectLegalHoldRequest& request)`

参数说明

- GetObjectLegalHoldRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetKey(const Aws::String& value)	是	设置对象名
void SetVersionId(const Aws::String& value)	可选	设置版本号

返回结果说明

- GetObjectLegalHoldOutCome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
GetObjectLegalHoldResult& GetResult()	获取结果

- GetObjectLegalHoldResult

获取结果方法	描述
const ObjectLockLegalHold& GetLegalHold() const	获取依法保留配置

- ObjectLockLegalHold

获取结果方法	描述
const ObjectLockLegalHoldStatus& GetStatus() const	获取保留开启状态

代码示例

```
#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();
        cout << "ERROR: " << 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 << "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[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

    get_object_legal_hold(client, bucket, key, "<your-versionId>");
    Aws::ShutdownAPI(options);
}
```

复制对象

功能说明

将一个文件从源路径复制到目标路径。

方法原型

`CopyObjectOutcome CopyObject(const CopyObjectRequest &request) const`

参数说明

- CopyObjectRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置目的桶名
void SetKey(const Aws::String& value)	是	设置目的对象名
void SetCopySource(const Aws::String& value)	是	设置源桶名及对象名，格式为"bucketName/objectName"
void SetCopySourceIfMatch(const Aws::String& value)	可选	仅当指定的Etag和Copy Source指定的object的Etag匹配时才复制
void SetCopySourceIfNoneMatch(const Aws::String& value)	可选	仅当指定的Etag和Copy Source指定的object的Etag不匹配时才复制
void SetCopySourceIfModifiedSince(const Aws::Utils::DateTime& value)	可选	仅当CopySource在指定时间后更新过才复制
void SetCopySourceIfUnmodifiedSince(const Aws::Utils::DateTime& value)	可选	仅当CopySource在指定时间后未更新过才复制
void SetACL(const ObjectCannedACL& value)	可选	设置对象ACL，可选项： ObjectCannedACL::private_，私有（默认）； ObjectCannedACL::public_read，公共读，但只有对象所有者可以写入和更改ACL； ObjectCannedACL::public_read_write，公共读写，但只有对象所有者可以写入和更改ACL； ObjectCannedACL::bucket_owner_read，对象所有者和存储桶所有者可以读取该对象，只有对象所有者可以写入和更改ACL； ObjectCannedACL::bucket_owner_full_control，对象所有者和存储桶所有者可以读取和写入该对象以及更改ACL。
void SetMetadata(const Aws::Map<Aws::String, Aws::String>& value)	可选	设置对象元数据
void SetMetadataDirective(const MetadataDirective& value)	可选	是否沿用元数据： MetadataDirective::COPY，复制源对象的元数据； MetadataDirective::REPLACE，替换新的元数据。
void SetTagging(const Aws::String& value)	可选	设置标签
void SetTaggingDirective(const TaggingDirective& value)	可选	是否沿用标签： TaggingDirective::COPY，复制源对象的标签； TaggingDirective::REPLACE，替换新的标签。
void SetObjectLockMode(const ObjectLockMode& value)	可选	设置合规保留模式： ObjectLockRetentionMode::COMPLIANCE，设置后不允许修改； ObjectLockRetentionMode::GOVERNANCE，设置后允许修改。

设定参数方法	是否必选	描述
void SetObjectLockRetainUntilDate(const Aws::Utils::DateTime& value)	可选	设置保留到期时间
void SetStorageClass(const StorageClass& value)	可选	设置对象存储类型，可选项： "STANDARD"，标准存储； "STANDARD_IA"，低频存储； "GLACIER"，归档存储。

返回结果说明

- CopyObjectOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
CopyObjectResult &GetResult()	获取请求结果

- CopyObjectResult

获取结果方法	描述
const Aws::String& GetCopySourceVersionId() const	获取源对象的版本号
const Aws::String& GetVersionId() const	获取复制后对象的版本号
const CopyObjectResultDetails& GetCopyObjectResultDetails() const	获取详细结果

- CopyObjectResultDetails

获取结果方法	描述
const Aws::String& GetETag() const	获取ETag
const Aws::Utils::DateTime& GetLastModified() const	获取复制对象的创建时间

代码示例

```
#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;
using namespace Aws::Client;
using namespace Aws::S3;
using namespace Aws::S3::Model;
using namespace Aws::Auth;
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.SetACL(ObjectCannedACL::public_read_write);
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetCopySource(source);

    auto outcome = client.CopyObject(request);

    if (!outcome.IsSuccess())
    {
        auto err = outcome.GetError();
        cout << "ERROR: " << 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: " << eTag << endl;
    }
}

int main(int argc, char *argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";
    String source = "<your-source-bucket>/<your-source-key>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);
```

```
    CopyObject(client, bucket, key, source);
    Aws::ShutdownAPI(options);
}
```

生成预签名链接

功能说明

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

`String GeneratePresignedUrl(const String& bucket, const String& key, Aws::Http::HttpMethod method, long long expirationInSeconds = MAX_EXPIRATION_SECONDS)`

参数说明

参数	类型	是否必选	描述
bucket	String	是	设置桶名
key	String	是	设置对象名
method	Aws::Http::HttpMethod	是	设置HTTP方法： HTTP_GET HTTP_POST HTTP_DELETE HTTP_PUT HTTP_HEAD HTTP_PATCH
expirationInSeconds	long long	是	设置有效期

返回结果说明

预签名链接字符串。

代码示例

- 仅获取链接

```
#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[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    String url = client.GeneratePresignedUrl(bucket, key,
Aws::Http::HttpMethod::HTTP_PUT, 300);
    cout << url << endl;

    Aws::ShutdownAPI(options);
}
```

- 获取上传链接并上传对象，需要依赖外部库libcurl。

```
#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 <curl/curl.h>
#include <iostream>
#include <iomanip>
#include <string>
#include <sstream>
#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;

static size_t read_callback(void* ptr, size_t size, size_t nmemb, void* stream)
{
    size_t retcode;
    curl_off_t nread;
    retcode = fread(ptr, size, nmemb, (FILE*)stream);
    nread = (curl_off_t)retcode;
    fprintf(stderr, "*** We read %" CURL_FORMAT_CURL_OFF_T" bytes from file\n",
nread);
    return retcode;
}

int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";
    String filePath = "<your-filePath>";
    FILE* file = fopen(filePath.c_str(), "rb");

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    String url = client.GeneratePresignedUrl(bucket, key,
Aws::Http::HttpMethod::HTTP_PUT, 300);
    cout << url << endl;

    curl_global_init(CURL_GLOBAL_DEFAULT);
    CURL* curl = curl_easy_init();
    if (curl)
    {
        curl_easy_setopt(curl, CURLOPT_URL, url.c_str());
        curl_easy_setopt(curl, CURLOPT_READFUNCTION, read_callback);
        curl_easy_setopt(curl, CURLOPT_READDATA, file);
        curl_easy_setopt(curl, CURLOPT_UPLOAD, 1L);
        curl_easy_setopt(curl, CURLOPT_VERBOSE, 1L);
        //设置acl，"private"/"public-read"/"public-read-write"
        /*struct curl_slist* headers = nullptr;
        headers = curl_slist_append(headers, "x-amz-acl: private");
        curl_easy_setopt(curl, CURLOPT_HTTPHEADER, headers);*/

        CURLcode res = curl_easy_perform(curl);
        if (res != CURLE_OK) {
            std::cerr << "curl_easy_perform() failed: " <<
curl_easy_strerror(res) << std::endl;
        } else {
            long httpCode = 0;
            curl_easy_getinfo(curl, CURLINFO_RESPONSE_CODE, &httpCode);
            std::cout << "HTTP Response Code: " << httpCode << std::endl;
        }
        curl_easy_cleanup(curl);
    }
    // 清理libcurl
    curl_global_cleanup();

    Aws::ShutdownAPI(options);
}
```

- 获取下载链接并下载对象，需要依赖外部库libcurl。

```
#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 <curl/curl.h>
#include <iostream>
#include <iomanip>
#include <string>
#include <sstream>
#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;

static size_t write_callback(void* ptr, size_t size, size_t nmemb, FILE* stream)
{
    return fwrite(ptr, size, nmemb, stream);
}

int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";
    String filePath = "<your-filePath>";
    FILE* file = fopen(filePath.c_str(), "wb");

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    String url = client.GeneratePresignedUrl(bucket, key,
Aws::Http::HttpMethod::HTTP_GET, 300);
    cout << url << endl;

    curl_global_init(CURL_GLOBAL_DEFAULT);
    CURL* curl = curl_easy_init();
    if (curl)
    {
        curl_easy_setopt(curl, CURLOPT_URL, url.c_str());
        curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, write_callback);
        curl_easy_setopt(curl, CURLOPT_WRITEDATA, file);
        curl_easy_setopt(curl, CURLOPT_VERBOSE, 1L);

        CURLcode res = curl_easy_perform(curl);
        if (res != CURLE_OK) {
            std::cerr << "curl_easy_perform() failed: " <<
curl_easy_strerror(res) << std::endl;
        } else {
            long httpCode = 0;
            curl_easy_getinfo(curl, CURLINFO_RESPONSE_CODE, &httpCode);
            std::cout << "HTTP Response Code: " << httpCode << std::endl;
        }
        curl_easy_cleanup(curl);
    }
    // 清理libcurl
    curl_global_cleanup();

    Aws::ShutdownAPI(options);
}
```

上传软链接

功能说明

使用接口对目标对象创建软链接，可以通过访问软链接来获取目标对象。删除软链接时，仅删除软链接本身，不会影响指
向的目标对象。

方法原型

`PutSymlinkOutcome PutSymlink(PutSymlinkRequest& request)`

参数说明

- PutSymlinkRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetKey(const Aws::String& value)	是	设置软链接名
void SetSymlinkTarget(const Aws::String& value)	是	设置链接的对象名

返回结果说明

- PutSymlinkOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
PutSymlinkResult &GetResult()	获取请求结果

- PutSymlinkResult

获取结果方法	描述
Aws::String& GetVersionId()	获取版本号

代码示例

```
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/PutSymlinkRequest.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 PutSymlink(S3Client& client, const Aws::String& bucket_name, const
Aws::String& object_name, const Aws::String& target_name) {
    // 设置请求参数
    PutSymlinkRequest request;
    request.SetBucket(bucket_name);
    request.SetKey(object_name);
    request.SetSymlinkTarget(target_name);

    // 发出请求
    PutSymlinkOutcome outcome = client.PutSymlink(request);

    //处理请求结果
    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        cout << "ERROR: " << 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 << "successful PutSymlink: " << object_name << std::endl;
        cout << "VersionId: " << outcome.GetResult().GetVersionId() << endl;
    }
    return;
}

int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";
    String target = "<your-target>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

    PutSymlink(client, bucket, key, target);
    Aws::ShutdownAPI(options);
}
```

获取软链接

功能说明

获取软链接，得到版本号和链接所指向的对象名。

方法原型

`GetSymlinkOutcome GetSymlink(GetSymlinkRequest& request)`

参数说明

- GetSymlinkRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetKey(const Aws::String& value)	是	设置软链接名
void setVersionId(const Aws::String&versionId)	可选	设置版本号

返回结果说明

- GetSymlinkOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
GetSymlinkResult &GetResult()	获取请求结果

- GetSymlinkResult

获取结果方法	描述
Aws::String& GetVersionId()	获取版本号
Aws::String& GetSymlinkTarget()	获取链接的对象名

代码示例

```
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/GetSymlinkRequest.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 GetSymlink(S3Client& client, const Aws::String& bucket_name, const
Aws::String& object_name) {
    // 设置请求参数
    GetSymlinkRequest request;
    request.SetBucket(bucket_name);
    request.SetKey(object_name);

    // 发出请求
    GetSymlinkOutcome outcome = client.GetSymlink(request);

    // 处理请求结果
    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        cout << "ERROR: " << 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 << "successful GetSymlink: " << object_name << std::endl;
        GetSymlinkResult result = outcome.GetResult();
        cout << "VersionId: " << result.GetVersionId() << endl;
        cout << "SymlinkTarget: " << result.GetSymlinkTarget() << endl;
    }
    return;
}

int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";

    // 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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

    GetSymlink(client, bucket, key);
    Aws::ShutdownAPI(options);
}
```

解冻对象

功能说明

用于解冻归档对象，归档对象无法直接读取，需要先解冻。解冻操作会生成一份带有生存期的标准类型副本数据，副本数据可设置有效期1~31天。

方法原型

`RestoreObjectOutcome restoreObjectV2(RestoreObjectRequest request)`

参数说明

- RestoreObjectRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetKey(const Aws::String& value)	是	设置软链接名
void SetRestoreRequest(const RestoreRequest& value)	是	设置解冻配置
void setVersionId(const Aws::String&versionId)	可选	设置版本号

- RestoreRequest

设定参数方法	是否必选	描述
void SetDays(int value)	是	设置有效期

返回结果说明

- RestoreObjectOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类

代码示例

```cpp
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/RestoreObjectRequest.h>
#include <aws/s3/model/RestoreRequest.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 RestoreObject(S3Client& client, const Aws::String& bucket_name, const
Aws::String& object_name) {
    // 设置请求参数
    RestoreObjectRequest request;
    RestoreRequest rr;
    rr.SetDays(2);
    request.SetBucket(bucket_name);
    request.SetKey(object_name);
    request.SetRestoreRequest(rr);

    auto outcome = client.RestoreObject(request);

    if (!outcome.IsSuccess())
    {
        auto err = outcome.GetError();
        cout << "ERROR: " << 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 << "request success" << endl;
    }
}
```

```
int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";

    // 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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

    RestoreObject(client, bucket, key);
    Aws::ShutdownAPI(options);
}
```

分段上传

创建分段上传

功能说明

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 用于后续的上传分段请求。

方法原型

`CreateMultipartUploadOutcome CreateMultipartUpload(const CreateMultipartUploadRequest& request) const`

参数说明

- CreateMultipartUploadRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetKey(const Aws::String& value)	是	设置对象名
void SetStorageClass(const StorageClass& value)	可选	设置对象存储类型，可选项： "STANDARD"，标准存储； "STANDARD_IA"，低频存储； "GLACIER"，归档存储。
void SetTagging(const Tagging& value)	可选	设置标签
void SetACL(const ObjectCannedACL& value)	可选	设置对象ACL，可选项： ObjectCannedACL::private_，私有（默认）； ObjectCannedACL::public_read，公共读，但只有对象所有者可以写入和更改ACL； ObjectCannedACL::public_read_write，公共读写，但只有对象所有者可以写入和更改ACL； ObjectCannedACL::bucket_owner_read，对象所有者和存储桶所有者可以读取该对象，只有对象所有者可以写入和更改ACL； ObjectCannedACL::bucket_owner_full_control，对象所有者和存储桶所有者可以读取和写入该对象以及更改ACL。
oid SetMetadata(const Aws::Map<Aws::String, Aws::String>& value)	可选	设置对象元数据

返回结果说明

- CreateMultipartUploadOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
CreateMultipartUploadResult& GetResult()	获取请求结果

- CreateMultipartUploadResult

获取结果方法	描述
const Aws::String& GetUploadId()	获取upload id

代码示例

```
#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);

    // 发出请求
    auto outcome = client.CreateMultipartUpload(request);

      //处理请求结果
    if (!outcome.IsSuccess())
    {
        auto err = outcome.GetError();
        cout << "ERROR: " << 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 outresult = outcome.GetResult();
        cout << "upload ID: " << outresult.GetUploadId() << endl;
        cout << "request success " << endl;
    }
    return;
}

int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg, 
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    create_mulripart_upload(client, bucket, key);
    Aws::ShutdownAPI(options);
}
```

上传分段

功能说明

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

`UploadPartOutcome UploadPart(const UploadPartRequest& request)`

参数说明

- UploadPartRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetKey(const Aws::String& value)	是	设置对象名
void SetUploadId(const Aws::String& value)	是	设置Upload ID
void SetBody(const std::shared_ptr<Aws::IOStream>& body)	是	设置文件数据
void SetPartNumber(int value)	是	设置分段号，从1开始

返回结果说明

- UploadPartOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
UploadPartResult& GetResult()	获取请求结果

- UploadPartResult

获取结果方法	描述
const Aws::String& GetETag()	获取ETag

代码示例

```
#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, String &bucket_name, String &key_name, const
String &file, int partnumber, const String &upload_id) {

    //确保文件存在

    std::shared_ptr<Aws::IOStream> input_data =
        Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
            file.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(upload_id);
    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();
        cout << "ERROR: " << 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 outresult = outcome.GetResult();
        std::cout << "Etag: " << outresult.GetETag() << std::endl;
        std::cout << "request success " << std::endl;
    }
    return;
}

int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";
    String file = "<your-filePath>";
    int partNumber = "<your-partNumber>";
    String uploadId = "<your-uploadId>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    upload_part(client, bucket, key, file, partNumber, uploadId);
    Aws::ShutdownAPI(options);
}
```

完成分段上传

功能说明

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

`CompleteMultipartUploadOutcome CompleteMultipartUpload(const CompleteMultipartUploadRequest& request)`

参数说明

- CompleteMultipartUploadRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetKey(const Aws::String& value)	是	设置对象名
void SetUploadId(const Aws::String& value)	是	设置upload id
void SetMultipartUpload(const CompletedMultipartUpload& value)	是	设置各分段信息

- CompletedMultipartUpload

设定参数方法	是否必选	描述
void SetParts(const Aws::Vector<CompletedPart>& value)	是	设置分段列表

- CompletedPart

设定参数方法	是否必选	描述
void SetETag(const Aws::String& value)	是	设置分段Etag
void SetPartNumber(int value)	是	设置分段号

返回结果说明

- CompleteMultipartUploadOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
CompleteMultipartUploadResult& GetResult()	获取请求结果

- CompleteMultipartUploadResult

获取结果方法	描述
const Aws::String& GetETag()	获取整个文件Etag

代码示例

```
#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>
```

```cpp
#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, const Aws::String& upload_id) {
    // 设置请求参数
    CompleteMultipartUploadRequest request;
    request.SetBucket(bucket_name);
    request.SetKey(key_name);
    request.SetUploadId(upload_id);

    CompletedPart part1;
    part1.SetPartNumber(1);
    part1.SetETag("<your-etag>");

    Aws::Vector<CompletedPart> vec_value;
    vec_value.push_back(part1);

    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();
        cout << "ERROR: " << 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 outresult = outcome.GetResult();
        std::cout<< "GetETag:" << outresult.GetETag() <<std::endl;
        std::cout << "request success " << std::endl;
    }
    return;
}

int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";
    String uploadId = "<your-uploadId>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    complete_multipart_upload(client, bucket, key, uploadId);
    Aws::ShutdownAPI(options);
}
```

列出分段

功能说明

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

`ListPartsOutcome ListParts(const ListPartsRequest& request) const`

参数说明

- ListPartsRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetKey(const Aws::String& value)	是	设置对象名
void SetUploadId(const Aws::String& value)	是	设置upload id
void SetMaxParts(int value)	可选	设置返回的最多分段数
void SetPartNumberMarker(int value)	可选	列举分段的起始位置，返回的分段列表是分段号大于此参数的分段

返回结果说明

- ListPartsOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
ListPartsResult& GetResult()	获取请求结果

- ListPartsResult

获取结果方法	描述
const Aws::Vector<Part>& GetParts() const	获取分段信息
const Aws::String& GetBucket()	获取文件存储的桶名
int GetNextPartNumberMarker()	下次列举分段请求的起始位置。如果本次没有返回全部结果，响应请求中将包含该元素，用于标明接下来请求的分段标记
bool GetIsTruncated() const	是否返回全部结果
int GetPartNumberMarker() const	获取当前list的分段起始编号
int GetMaxParts() const	获取当前list返回的最大分段数目
const Aws::String& GetUploadId() const	获取分段上传的UploadID
const StorageClass& GetStorageClass() const	获取文件的存储类型
const Aws::String& GetKey() const	获取文件key

- Part

获取结果方法	描述
int GetPartNumber() const	获取当前分段的分段号
const Aws::String& GetETag() const	获取该分段的Etag
long long GetSize() const	获取当前分段的大小，单位字节

代码示例

```
#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 String &bucket_name, const String
&key_name, const String upload_id) {
    // 设置请求参数
    ListPartsRequest request;
    request.SetBucket(bucket_name);
    request.SetKey(key_name);
    request.SetUploadId(upload_id);

    // 发出请求
    auto outcome = client.ListParts(request);

    //处理请求结果
    if (!outcome.IsSuccess())
    {
        auto err = outcome.GetError();
        cout << "ERROR: " << 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 outresult = outcome.GetResult();
        cout<< "GetBucket:" << outresult.GetBucket() <<endl;
        cout<< "GetNextPartNumberMarker:" << outresult.GetNextPartNumberMarker()
<<endl;
        cout<< "GetIsTruncated:" << outresult.GetIsTruncated() <<endl;
        cout<< "GetMaxParts:" << outresult.GetMaxParts() <<endl;
        cout<< "GetUploadId:" << outresult.GetUploadId() <<endl;
        cout<< "GetStorageClass:" << (int)outresult.GetStorageClass() << endl;
        auto parts = outresult.GetParts();
        for(auto part: parts){
            cout<< "part:GetPartNumber:" << part.GetPartNumber() <<endl;
            cout<< "part:GetLastModified:" << part.GetLastModified().Millis()
<<endl;
            cout<< "part:GetETag:" << part.GetETag() <<endl;
            cout<< "part:GetSize:" << part.GetSize() <<endl;
        }
        cout << "request success " << endl;
    }
    return;
}

int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";
    String uploadId = "<your-uploadId>";
    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    list_parts(client, bucket, key, uploadId);
    Aws::ShutdownAPI(options);
}
```

列出所有分段上传

功能说明

查询正在进行中的分段上传，也就是已经 Created但是还没有Aborted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

`ListMultipartUploadsOutcome ListMultipartUploads(const ListMultipartUploadsRequest& request) const`

参数说明

- ListMultipartUploadsRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetPrefix(const Aws::String& value)	可选	只有以Prefix为开头的Key的分段上传数据才会返回
void SetMaxUploads(int value)	可选	单次最多返回的分段上传数据，大小是1-1000
void SetDelimiter(const Aws::String& value)	可选	对象名按照此标识符进行分组。通常与prefix参数搭配使用，如果指定了prefix，从prefix到第一次出现delimiter间具有相同字符串的对象名会被分成一组
void SetKeyMarker(const Aws::String& value)	可选	列举分段上传的起始位置，返回的对象列表将是对象名按照字典序排序后该参数以后的所有对象
void SetUploadIdMarker(const Aws::String& value)	可选	与keyMarker配合使用，返回的对象列表将是对象名和uploadId按照字典序排序后该参数以后的所有对象

返回结果说明

- ListMultipartUploadsOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
ListMultipartUploadsResult& GetResult()	获取请求结果

- ListMultipartUploadsResult

获取结果方法	描述
const Aws::Vector<MultipartUpload>& GetUploads()	获取分段上传数据
bool GetIsTruncated()	获取是否返回了所有满足要求的分段上传
Aws::String& GetNextKeyMarker()	下次列举分段上传请求的起始位置。如果本次没有返回全部结果，响应请求中将包含该元素，用于标明接下来请求的keyMarker值
Aws::String& GetNextUploadIdMarker()	下次列举分段上传请求的起始位置，与nextKeyMarker配合使用。如果本次没有返回全部结果，响应请求中将包含该元素，用于标明接下来请求的uploadIdMarker值。
Aws::String& GetPrefix()	获取请求设置的前缀
Aws::String& GetDelimiter()	获取请求设置的分隔符

- MultipartUpload

获取结果方法	描述
const Aws::String& GetUploadId()	获取分段上传uploadId
const Aws::String& GetKey()	获取文件key
const StorageClass& GetStorageClass()	获取文件存储类型
const Aws::S3::Model::Owner& GetOwner()	获取分段上传所属用户
const Aws::Utils::DateTime& GetInitiated() const	获取分段上传的初始化时间

代码示例

```
#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();
        cout << "ERROR: " << 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 outresult = outcome.GetResult();
        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 << "request success " << std::endl;
    }
    return;
}

int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    list_multi_uploads(client, bucket);
    Aws::ShutdownAPI(options);
}
```

取消分段上传

功能说明

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

`AbortMultipartUploadOutcome AbortMultipartUpload(const AbortMultipartUploadRequest& request) const`

参数说明

- AbortMultipartUploadRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetKey(const Aws::String& value)	是	设置对象名
void SetUploadId(const Aws::String& value)	是	设置Upload ID

返回结果说明

- AbortMultipartUploadOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
UploadPartResult& GetResult()	获取请求结果

代码示例

```
#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, const Aws::String& upload_id) {
    // 设置请求参数
    AbortMultipartUploadRequest request;
    request.SetBucket(bucket_name);
    request.SetKey(key_name);
    request.SetUploadId(upload_id);

    // 发出请求
    auto outcome = client.AbortMultipartUpload(request);

    //处理请求结果
    if (!outcome.IsSuccess())
    {
        auto err = outcome.GetError();
        cout << "ERROR: " << 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 << "request success " << std::endl;
    }
    return;
}

int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";
    String uploadId = "<your-uploadId>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    abort_multipart_upload(client, bucket, key, uploadId);
    Aws::ShutdownAPI(options);
}
```

分段复制

功能说明

请求适用于将大文件分段复制到目标路径（桶）。

方法原型

UploadPartCopyOutcome UploadPartCopy(const UploadPartCopyRequest& request)

参数说明

- UploadPartCopyRequest

设定参数方法	是否必选	描述
void SetBucket(Aws::String&& value)	是	设置目的桶名
void SetKey(const Aws::String& value)	是	设置目的对象名
void SetCopySource(const Aws::String& value)	是	设置源桶名及对象名，格式为"bucketName/objectName"
void SetCopySourceRange(const Aws::String& value)	是	设置当前分段字节范围，"bytes=0-9"表示开头的10个字节
void SetPartNumber(int value)	是	设置分段编号
void SetUploadId(const Aws::String& value)	是	设置uploadId
void setSourceVersionId(String sourceVersionId)	可选	设置源对象版本号
void SetCopySourceIfNoneMatch(const Aws::String& value)	可选	仅当指定的Etag和Copy Source指定的object的Etag匹配时才复制
void setNonmatchingETagConstraints(List<String> eTagList)	可选	仅当指定的Etag和Copy Source指定的object的Etag不匹配时才复制
void SetCopySourceIfModifiedSince(const Aws::Utils::DateTime& value)	可选	仅当CopySource在指定时间后更新过才复制
void SetCopySourceIfUnmodifiedSince(const Aws::Utils::DateTime& value)	可选	仅当CopySource在指定时间后未更新过才复制

返回结果说明

- UploadPartCopyOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
UploadPartCopyResult& GetResult()	获取请求结果

- UploadPartCopyResult

获取结果方法	描述
const CopyPartResult& GetCopyPartResult() const	获取分段复制结果
const Aws::String& GetCopySourceVersionId() const	获取复制后对象的版本号

- CopyPartResult

获取结果方法	描述
const Aws::String& GetETag() const	获取分段ETag
const Aws::Utils::DateTime& GetLastModified() const	获取本次Copy的上传时间

代码示例

```
#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;
using namespace Aws::Client;
using namespace Aws::S3;
using namespace Aws::S3::Model;
using namespace Aws::Auth;
using namespace std;

void CopyPart(S3Client &client)
{
    Aws::String srcBucekt = "<your-bucket>";
    Aws::String srcKey = "<your-key>";
    Aws::String dstBucket = "<your-bucket>";
    Aws::String dstKey = "<your-key>";

    // 获取源文件长度
    HeadObjectRequest headRequest;
    headRequest.WithBucket(srcBucekt);
    headRequest.WithKey(srcKey);
    auto headOutcome = client.HeadObject(headRequest);
    if (!headOutcome.IsSuccess())
    {
        auto err = headOutcome.GetError();
        cout << "ERROR: " << 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 = headOutcome.GetResult().GetContentLength();
    // 创建目的桶的分段上传请求获得uploadId
    CreateMultipartUploadRequest initRequest;
    initRequest.SetBucket(dstBucket);
    initRequest.SetKey(dstKey);
    auto initResponse = client.CreateMultipartUpload(initRequest);
    if (!initResponse.IsSuccess())
    {
        auto err = initResponse.GetError();
        cout << "ERROR: " << endl
            << "Http code: " << (int)err.GetResponseCode() << endl
            << "Error Type:" << (int)err.GetErrorType() << endl
            << "Error Msg: " << err.GetMessage() << endl
            << "Exception Name: " << err.GetExceptionName() << endl;
        return;
    }
    String uploadId = initResponse.GetResult().GetUploadId();

    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;
    UploadPartCopyRequest request;
    while (bytePosition < size)
    {
        stringstream ss;
        long lastByte = std::min(bytePosition + partSize - 1, size - 1);
        // 复制一段
        ss << "bytes=" << bytePosition <<"-" << lastByte;
        request.WithBucket(dstBucket)
            .WithKey(dstKey)
            .WithCopySource(srcBucekt + "/" + srcKey)
            .WithUploadId(uploadId)
            .WithPartNumber(partNum)
            .WithCopySourceRange(ss.str().c_str());

        auto uploadOutCome = client.UploadPartCopy(request);
        if (!uploadOutCome.IsSuccess())
        {
            auto err = uploadOutCome.GetError();
            cout << "ERROR: " << 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++;
    }

    CompleteMultipartUploadRequest  finishRequest;
    CompletedMultipartUpload uploadResult;
    uploadResult.SetParts(completedPart);
    // 完成 multiPartUpload
    finishRequest.WithBucket(dstBucket)
        .WithKey(dstKey)
        .WithUploadId(uploadId)
        .WithMultipartUpload(uploadResult);

    auto finishOutcom = client.CompleteMultipartUpload(finishRequest);
    if (!finishOutcom.IsSuccess())
    {
        auto err = finishOutcom.GetError();
        cout << "ERROR: " << 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[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    CopyPart(client);
    Aws::ShutdownAPI(options);
}
```

客户端加密

只支持AES256加密算法。

生成加密密钥

```
// 生成Key的要妥善保管，如果该key丢失了，那么意味着通过该key加密的数据将没法解密
Aws::Utils::CryptoBuffer GetKey() {
    return Aws::Utils::Crypto::SymmetricCipher::GenerateKey();
}
```

保存加密密钥

```
void SaveKey(Aws::Utils::CryptoBuffer originalKey, Aws::String path) {
    Aws::Utils::Base64::Base64 myBase64;
    Aws::String keyEncoded = myBase64.Encode(originalKey);
    std::ofstream file(path, std::ios::binary | std::ios::out);
    if (!file.is_open()) {
        throw std::runtime_error("Cannot open file: " + path);
    }
    file << keyEncoded;
    file.close();
}
```

加载加密密钥

```
Aws::Utils::CryptoBuffer LoadKey(Aws::String path) {
    std::ifstream fileStream(path, std::ios::in | std::ios::binary);
    if (!fileStream) {
        throw std::runtime_error("Cannot open file: " + path);
    }
    std::stringstream buffer;
    buffer << fileStream.rdbuf();
    fileStream.close();
    Aws::String keyString = Aws::String(buffer.str());
    Aws::Utils::Base64::Base64 myBase64;
    Aws::Utils::CryptoBuffer keyDecoded = myBase64.Decode(keyString);
    return keyDecoded;
}
```

加密上传示例

请求参数与返回和普通上传相同。

```
#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>
#include <aws/core/utils/base64/Base64.h>
#include <aws/s3-encryption/S3EncryptionClient.h>
#include <aws/s3-encryption/CryptoConfiguration.h>
#include <aws/s3-encryption/materials/SimpleEncryptionMaterials.h>
using namespace Aws;
using namespace Aws::Client;
using namespace Aws::S3;
using namespace Aws::S3::Model;
using namespace Aws::Auth;
using namespace Aws::S3Encryption;
using namespace Aws::S3Encryption::Materials;
using namespace std;

void PutObject(S3EncryptionClientV2 &client, const Aws::String &bucket,
Aws::String objectName, Aws::String fname)
```

```
{
    // 设置请求
    PutObjectRequest request;
    std::shared_ptr<Aws::IOStream> inputData =
        Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
            fname.c_str(),
            std::ios_base::in | std::ios_base::binary);

    request.SetBody(inputData);
    request.SetBucket(bucket);
    request.SetKey(objectName);

    auto outcome = client.PutObject(request, {});

    if (!outcome.IsSuccess())
    {
        auto err = outcome.GetError();
        cout << "ERROR: " << endl
            << "Http code: " << (int)err.GetResponseCode() << 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;
    }
}

Aws::Utils::CryptoBuffer GetKey() {
    return Aws::Utils::Crypto::SymmetricCipher::GenerateKey();
}

void SaveKey(Aws::Utils::CryptoBuffer originalKey, Aws::String path) {
    Aws::Utils::Base64::Base64 myBase64;
    Aws::String keyEncoded = myBase64.Encode(originalKey);
    std::ofstream file(path, std::ios::binary | std::ios::out);
    if (!file.is_open()) {
        throw std::runtime_error("Cannot open file: " + path);
    }
    file << keyEncoded;
    file.close();
}

int main(int argc, char *argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";
    String filePath = "<your-localFilePath>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk

    auto symmetricKey = GetKey();
    SaveKey(symmetricKey, "<your-key-path>);
    const auto simpleEncryptionMaterialsWithGCMAAD =
Aws::MakeShared<SimpleEncryptionMaterialsWithGCMAAD>("example", symmetricKey);
    CryptoConfigurationV2 cryptoConfig(simpleEncryptionMaterialsWithGCMAAD);
    S3EncryptionClientV2 s3EncryptionClient(cryptoConfig, cred, cfg);

    PutObject(s3EncryptionClient, bucket, key, savePath);
    Aws::ShutdownAPI(options);
}
```

解密下载示例

请求参数与返回和普通下载相同。

```
#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>
#include <aws/core/utils/base64/Base64.h>
#include <aws/s3-encryption/S3EncryptionClient.h>
#include <aws/s3-encryption/CryptoConfiguration.h>
#include <aws/s3-encryption/materials/SimpleEncryptionMaterials.h>
using namespace Aws;
using namespace Aws::Client;
using namespace Aws::S3;
using namespace Aws::S3::Model;
using namespace Aws::Auth;
using namespace Aws::S3Encryption;
using namespace Aws::S3Encryption::Materials;
using namespace std;

void GetObject(S3EncryptionClientV2 &client, const Aws::String &bucket,
Aws::String objectName, Aws::String fname)
{
    // 设置请求
    GetObjectRequest request;

    request.SetBucket(bucket);
    request.SetKey(objectName);

    auto outcome = client.GetObject(request);

    if (!outcome.IsSuccess())
    {
        auto err = outcome.GetError();
        cout << "ERROR: " << endl
```

```
                << "Http code: " << (int)err.GetResponseCode() << endl
                << "Error Msg: " << err.GetMessage() << endl
                << "Exception Name: " << err.GetExceptionName() << endl;
    }
    else
    {
        std::cout << "GetObject request success " << std::endl;
        auto result = outcome.GetResultWithOwnership();
        auto &file = result.GetBody();
        Aws::Map<Aws::String, Aws::String> meta = result.GetMetadata();

        ofstream outFile;
        outFile.open(fname, ios::out | ios::binary);
        string bytes(5 * 1024 * 1024, '\0');
        while (file.read(&bytes[0], sizeof(bytes)).gcount() > 0) {
            outFile.write(&bytes[0], file.gcount());
        }
        outFile.close();
    }
}

Aws::Utils::CryptoBuffer LoadKey(Aws::String path) {
    std::ifstream fileStream(path, std::ios::in | std::ios::binary);
    if (!fileStream) {
        throw std::runtime_error("Cannot open file: " + path);
    }
    std::stringstream buffer;
    buffer << fileStream.rdbuf();
    fileStream.close();
    Aws::String keyString = Aws::String(buffer.str());
    Aws::Utils::Base64::Base64 myBase64;
    Aws::Utils::CryptoBuffer keyDecoded = myBase64.Decode(keyString);
    return keyDecoded;
}

int main(int argc, char *argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";
    String filePath = "<your-localFilePath>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk

    auto symmetricKey = LoadKey("<your-key-path>");
    const auto simpleEncryptionMaterialsWithGCMAAD =
Aws::MakeShared<SimpleEncryptionMaterialsWithGCMAAD>("example", symmetricKey);
    CryptoConfigurationV2 cryptoConfig(simpleEncryptionMaterialsWithGCMAAD);
    S3EncryptionClientV2 s3EncryptionClient(cryptoConfig, cred, cfg);

    GetObject(s3EncryptionClient, bucket, key, savePath);
    Aws::ShutdownAPI(options);
}
```

图片处理

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

`ProcessObjectOutcome ProcessObject(const ProcessObjectRequest& request)`

参数说明

- ProcessObjectRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置持久化的目的桶
void SetKey(const Aws::String& value)	是	设置持久化存储的目的名称（不能和原图相同）
void SetProcessSource(const Aws::String& value)	是	设置待处理的图片所在桶和对象名，格式："bucket/key"
void SetBody(const std::shared_ptr<Aws::IOStream>& body)	是	设置图片处理方式，用字符串传递，每一项参数格式均为"参数名称_取值"，之间用逗号分隔具体见下表

- 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，相对原点的横坐标（默认左上角为原点）。	[0,图片边界]	否
y	指定裁剪起点，相对原点的纵坐标（默认左上角为原点）。	[0,图片边界]	否
g	设置裁剪的原点位置。原点按照整幅图的九宫格形式分布，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,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 north：中上 ne：右上 west：左中 center：中部 east：右中 sw：左下 south：中下 se：右下 (默认)	否

- 格式转化(e.g. "image/format,png")

参数名称	参数用途	取值	是否必须
无	将原图转换成指定格式	jpg、png、webp、bmp、tiff	是

返回结果说明

- ProcessObjectOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
const ProcessObjectResult& GetResult()	获取请求结果

- ProcessObjectResult

获取结果方法	描述
const Aws::String& GetETag()	获取ETag
const Aws::String& GetVersionId()	获取版本号

代码示例

```
#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) {
    // 设置请求参数
    ProcessObjectRequest request;
    request.SetBucket(dest_bucket);
    request.SetKey(dest_name);
    request.SetProcessSource(process_source);
    Aws::String zosProcess = "image/resize,w_300,h_1000,m_fixed";
    request.SetBody(std::make_shared<std::stringstream>(zosProcess));
    // 发出请求
    auto outcome = client.ProcessObject(request);

        //处理请求结果
    if (!outcome.IsSuccess())
    {
        auto err = outcome.GetError();
        cout << "ERROR: " << 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 << "ETag: " << outcome.GetResult().GetETag() << endl;
        cout << "ProcessObject success" << endl;
    }
    return;
}
int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String srcBucket = "<your-srcBucket>";
    String dstBucket = "<your-dstBucket>";
    String srcKey = "<your-srcKey>";
    String dstKey = "<your-dstKey>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

    process_object(client, dstBucket, dstKey, srcBucket + "/" + srcKey);
    Aws::ShutdownAPI(options);
}
```

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

`S3Object getObject(GetObjectRequest request)`

参数说明

- GetObjectRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetKey(const Aws::String& value)	是	设置对象名
void SetZosProcess(const Aws::String& value)	可选	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)，具体见上节

返回结果说明

- S3Object

获取结果方法	描述
S3ObjectInputStream getObjectContent()	获取对象内容
String getBucketName()	获取对象所在桶
String getKey()	获取对象key
ObjectMetadata getObjectMetadata()	获取对象元数据

- ObjectMetadata

获取结果方法	描述
Map<String, String> getUserMetadata()	获取用户元数据
Date getLastModified()	获取对象创建时间
String getETag()	获取对象ETag
String getVersionId()	获取对象版本号
Long[] getContentRange()	若请求时指定了Range或partnumber，则返回结果中包含content range
String getObjectLockMode()	获取对象的合规保留模式
Date getObjectLockRetainUntilDate()	获取Retention 过期日期
String getStorageClass()	获取对象存储类型

代码示例

```
#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;
using namespace Aws::Client;
using namespace Aws::S3;
using namespace Aws::S3::Model;
using namespace Aws::Auth;
using namespace std;

void GetObject(S3Client &client, const Aws::String &bucket, Aws::String
objectName, Aws::String fname)
{
    // 设置请求
    GetObjectRequest request;

    request.SetBucket(bucket);
    request.SetKey(objectName);
    request.SetZosProcess("image/resize,w_100,h_100,m_fixed");

    auto outcome = client.GetObject(request);

    if (!outcome.IsSuccess())
    {
        auto err = outcome.GetError();
        cout << "ERROR: " << 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 << "GetObject request success " << std::endl;
        auto result = outcome.GetResultWithOwnership();
        auto &file = result.GetBody();
        Aws::Map<Aws::String, Aws::String> meta = result.GetMetadata();

        ofstream outFile;
        outFile.open(fname, ios::out | ios::binary);
        string bytes(5 * 1024 * 1024, '\0');
        while (file.read(&bytes[0], sizeof(bytes)).gcount() > 0) {
            outFile.write(&bytes[0], file.gcount());
        }
        outFile.close();
    }
}

int main(int argc, char *argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";
    String savePath = "<your-savePath>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    GetObject(client, bucket, key, savePath);
    Aws::ShutdownAPI(options);
}
```

解压缩操作

压缩打包对象

功能说明

发送一个多文件打包任务到zos，并返回任务ID，异步打包完成后，形成压缩包对象。

方法原型

`PostObjectsPackageOutcome PostObjectsPackage(const PostObjectsPackageRequest& request)`

参数说明

- PostObjectsPackageRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetKey(const Aws::String& value)	是	设置目标压缩包名
void SetSourceConfigPolicy(const SourceConfigPolicy& value)	是	设置打包策略

- SourceConfigPolicy

设定参数方法	是否必选	描述
void SetMode(int value)	是	设置打包规则： 2：按规则列表打包； 4：按规则文件打包（json文件）。
void SetSourceRules(const Aws::Vector<SourceRule>& value)	可选	设置打包规则列表，Mode为2时必填
void SetSourceIndex(const Aws::String& value)	可选	设置打包规则文件路径（桶内路径，"bucket/xxx.json"），Mode为4时必填
SourceConfigPolicy& AddSourceRules(const SourceRule& value)	可选	添加打包规则到打包规则列表

- SourceRule

设定参数方法	是否必选	描述
void SetResource(const Aws::String& value)	可选	设置源文件在桶中路径
void SetRenamePath(const Aws::String& value)	可选	设置打包后源文件在压缩包中的路径

- sourceindex

取值为桶内已有的json规则文件，需提前写好并上传，内容示例如下，逻辑与mode2相同：

```
{
    "Rules": [
        {
            "Resource": "bucket/file1",
            "renamePath": "path1/file1"
        },
        {
            "Resource": "bucket/file2",
            "renamePath": "path2/file2"
        }
    ]
}
```

返回结果说明

- PostObjectsPackageOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
const PostObjectsPackageResult& GetResult()	获取请求结果

- PostObjectsPackageResult

获取结果方法	描述
String& GetPersistentId()	获取打包任务id

代码示例

```
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/PostObjectsPackageRequest.h>
#include <aws/s3/model/SourceConfigPolicy.h>
#include <aws/s3/model/SourceRule.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 postObjectsPackage(S3Client& client, const Aws::String& bucket_name, const
Aws::String& object_name) {
    // 设置请求参数
    PostObjectsPackageRequest request;
    request.SetBucket(bucket_name);
    request.SetKey(object_name);

    SourceRule rule1;
    rule1.SetResource(bucket_name + "/" + "file1");
    rule1.SetRenamePath("path1/file1");
    SourceRule rule2;
    rule2.SetResource(bucket_name + "/" + "file2");
    rule2.SetRenamePath("path2/file2");

    SourceConfigPolicy policy;
    policy.SetMode(2);
    policy.AddSourceRules(rule1);
    policy.AddSourceRules(rule2);

    request.SetSourceConfigPolicy(policy);

    // 发出请求
    PostObjectsPackageOutcome outcome = client.PostObjectsPackage(request);

    //处理请求结果
    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        cout << "ERROR: " << 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 << "successful PostObjectsPackage: " << object_name <<
std::endl;
        cout << "PersistentId: " << outcome.GetResult().GetPersistentId() <<
endl;
    }
    return;
}

int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

    postObjectsPackage(client, bucket, key);
    Aws::ShutdownAPI(options);
}
```

获取打包任务

功能说明

通过persistent id获取打包任务的状态。

方法原型

`GetObjectsPackageOutcome GetObjectsPackage(const GetObjectsPackageRequest& request)`

参数说明

- GetObjectsPackageRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	可选	设置桶名
void SetKey(const Aws::String& value)	可选	设置目标压缩包名
void SetPersistentId(const Aws::String& value)	可选	设置打包任务id

返回结果说明

- GetObjectsPackageOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
const GetObjectsPackageResult& GetResult()	获取请求结果

- GetObjectsPackageResult

获取结果方法	描述
String& GetDescInfo()	获取描述信息
int GetMode()	获取打包模式
String& GetPackageStatus()	获取打包任务状态
String& GetSourceIndex()	获取打包规则文件
Vector<SourceRule>& GetSourceRules()	获取打包规则列表
String& GetPersistentId()	获取打包任务id

代码示例

```
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/GetObjectsPackageRequest.h>
#include <aws/s3/model/SourceConfigPolicy.h>
#include <aws/s3/model/SourceRule.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;
```

```cpp
void getObjectsPackage(S3Client& client, const Aws::String& bucket_name, const
Aws::String& object_name) {
    // 设置请求参数
    GetObjectsPackageRequest request;
    request.SetBucket(bucket_name);
    request.SetKey(object_name);
    request.SetPersistentId("xxx");

    // 发出请求
    GetObjectsPackageOutcome outcome = client.GetObjectsPackage(request);

    //处理请求结果
    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        cout << "ERROR: " << 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 << "successful GetObjectsPackage: " << object_name <<
std::endl;
        GetObjectsPackageResult result = outcome.GetResult();
        cout << "DescInfo: " << result.GetDescInfo() << endl;
        cout << "PackageStatus: " << result.GetPackageStatus() << endl;
    }
    return;
}

int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

    getObjectsPackage(client, bucket, key);
    Aws::ShutdownAPI(options);
}
```

解压对象

功能说明

发送一个解压任务到zos，并返回任务ID，异步解压完成后，形成解压后的多个对象。

方法原型

`PostObjectsUnzipOutcome PostObjectsUnzip(const PostObjectsUnzipRequest& request)`

参数说明

- PostObjectsUnzipRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	是	设置桶名
void SetKey(const Aws::String& value)	是	设置目标压缩包名
void SetUnzipConfigPolicy(const UnzipConfigPolicy& value)	是	设置解压策略

- UnzipConfigPolicy

设定参数方法	是否必选	描述
void SetDestBucket(const Aws::String& value)	可选	设置解压到目标桶，默认源桶
void SetDestPath(const Aws::String& value)	可选	设置解压到的路径，默认与压缩包同名文件夹
void SetPassword(const Aws::String& value)	可选	设置解压密码（如果压缩包有密码）
void SetKeepZipFileName(bool value)	可选	是否保留压缩包名的文件夹路径（默认否）
void SetOverwrite(int value)	可选	设置当解压路径存在重名文件时的处理方式（默认2）： 0：覆盖； 1：跳过； 2：重命名（以随机字符串+对象名的形式）。

返回结果说明

- PostObjectsUnzipOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
const PostObjectsUnzipResult& GetResult()	获取请求结果

- PostObjectsPackageResult

获取结果方法	描述
String& GetPersistentId()	获取解压任务id

代码示例

```
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/PostObjectsUnzipRequest.h>
#include <aws/s3/model/UnzipConfigPolicy.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 postObjectsUnzip(S3Client& client, const Aws::String& bucket_name, const
Aws::String& object_name) {
    // 设置请求参数
    PostObjectsUnzipRequest request;
    request.SetBucket(bucket_name);
    request.SetKey(object_name);

    UnzipConfigPolicy policy;
    policy.SetDestBucket(bucket_name);

    request.SetUnzipConfigPolicy(policy);

    // 发出请求
    PostObjectsUnzipOutcome outcome = client.PostObjectsUnzip(request);

    //处理请求结果
    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        cout << "ERROR: " << 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 << "successful PostObjectsUnzip: " << object_name <<
std::endl;
```

```
        cout << "PersistentId: " << outcome.GetResult().GetPersistentId() <<
endl;
    }
    return;
}

int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

    postObjectsUnzip(client, bucket, key);
    Aws::ShutdownAPI(options);
}
```

获取解压任务

功能说明

通过persistent id获取解压任务的状态。

方法原型

`GetObjectsUnzipOutcome GetObjectsUnzip(const GetObjectsUnzipRequest& request)`

参数说明

- GetObjectsUnzipRequest

设定参数方法	是否必选	描述
void SetBucket(const Aws::String& value)	可选	设置桶名
void SetKey(const Aws::String& value)	可选	设置目标压缩包名
void SetPersistentId(const Aws::String& value)	可选	设置解压任务id

返回结果说明

- GetObjectsUnzipOutcome

获取结果方法	描述
bool IsSuccess()	本次请求是否成功
const Aws::S3::S3Error& GetError()	获取本次请求错误类
const GetObjectsUnzipResult& GetResult()	获取请求结果

- GetObjectsUnzipResult

获取结果方法	描述
String& GetDescInfo()	获取描述信息
String& GetUnzipStatus()	获取解压任务状态
String& GetPersistentId()	获取解压任务id
String& GetProcess()	获取解压进度
UnzipConfigPolicy& GetUnzipConfig()	获取解压配置

- UnzipConfigPolicy

获取结果方法	描述
String& GetDestBucket()	获取目标桶
String& GetDestPath()	获取解压路径
String& GetPassword()	获取解压密码
bool GetKeepZipFileName()	获取保留压缩包名的文件夹路径
int GetOverwrite()	获取重名策略

代码示例

```
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/GetObjectsUnzipRequest.h>
#include <aws/s3/model/UnzipConfigPolicy.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 getObjectsUnzip(S3Client& client, const Aws::String& bucket_name, const
Aws::String& object_name) {
    // 设置请求参数
    GetObjectsUnzipRequest request;
    request.SetBucket(bucket_name);
    request.SetKey(object_name);
    request.SetPersistentId("xxx");

    // 发出请求
    GetObjectsUnzipOutcome outcome = client.GetObjectsUnzip(request);

    // 处理请求结果
    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        cout << "ERROR: " << 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 << "successful GetObjectsUnzip: " << object_name << std::endl;
        GetObjectsUnzipResult result = outcome.GetResult();
        cout << "DescInfo: " << result.GetDescInfo() << endl;
        cout << "UnzipStatus: " << result.GetUnzipStatus() << endl;
    }
    return;
}

int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";
    String key = "<your-key>";

    // 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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Never, false);

    getObjectsUnzip(client, bucket, key);
    Aws::ShutdownAPI(options);
}
```

特定场景

批量碎片清理

功能说明

清除桶内的所有文件碎片（分段上传的未完成部分）。

代码示例

```
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/ListMultipartUploadsRequest.h>
#include <aws/s3/model/AbortMultipartUploadRequest.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 defragment(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();
        cout << "ERROR: " << 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 outresult = outcome.GetResult();
        auto uploads = outresult.GetUploads();
        for(auto upload: uploads){
            AbortMultipartUploadRequest abortRequest;
            request.SetBucket(bucket_name);
            abortRequest.SetKey(upload.GetKey());
            abortRequest.SetUploadId(upload.GetUploadId());
            auto abortOutcome = client.AbortMultipartUpload(abortRequest);
            if (!outcome.IsSuccess())
            {
                auto err = abortOutcome.GetError();
                cout << "ERROR: " << endl
                    << "Http code: " << (int)err.GetResponseCode() << endl
                    << "Error Type:" << (int)err.GetErrorType() << endl
                    << "Error Msg: " << err.GetMessage() << endl
                    << "Exception Name: " << err.GetExceptionName() << endl;
                }
            }
            std::cout << "request success " << std::endl;
        }
    return;
}

int main(int argc, char* argv[])
{
    String ep = "<your-ep>";
    String ak = "<your-ak>";
    String sk = "<your-sk>";
    String bucket = "<your-bucket>";

    //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 = ep;
    cfg.scheme = Aws::Http::Scheme::HTTP;
    cfg.verifySSL = false;
    AWSCredentials cred(ak, sk);  // ak,sk
    S3Client client(cred, cfg,
Aws::Client::AWSAuthV4Signer::PayloadSigningPolicy::Always, false);

    defragment(client, bucket);
    Aws::ShutdownAPI(options);
}
```