下载与安装

相关资源

- SDK 压缩包下载地址：见帮助中心

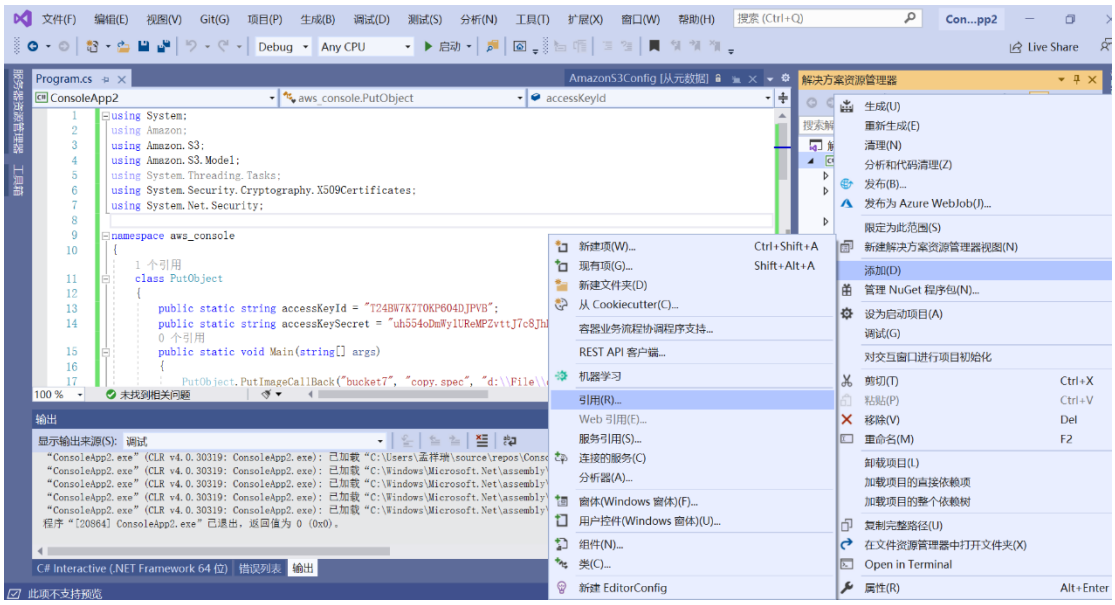
环境依赖

对象存储的 C# SDK 目前可以支持 Visual Studio 版本 2017 及更高版本(Windows 环境)，需要确保使用环境已经安装 Visual Studio。

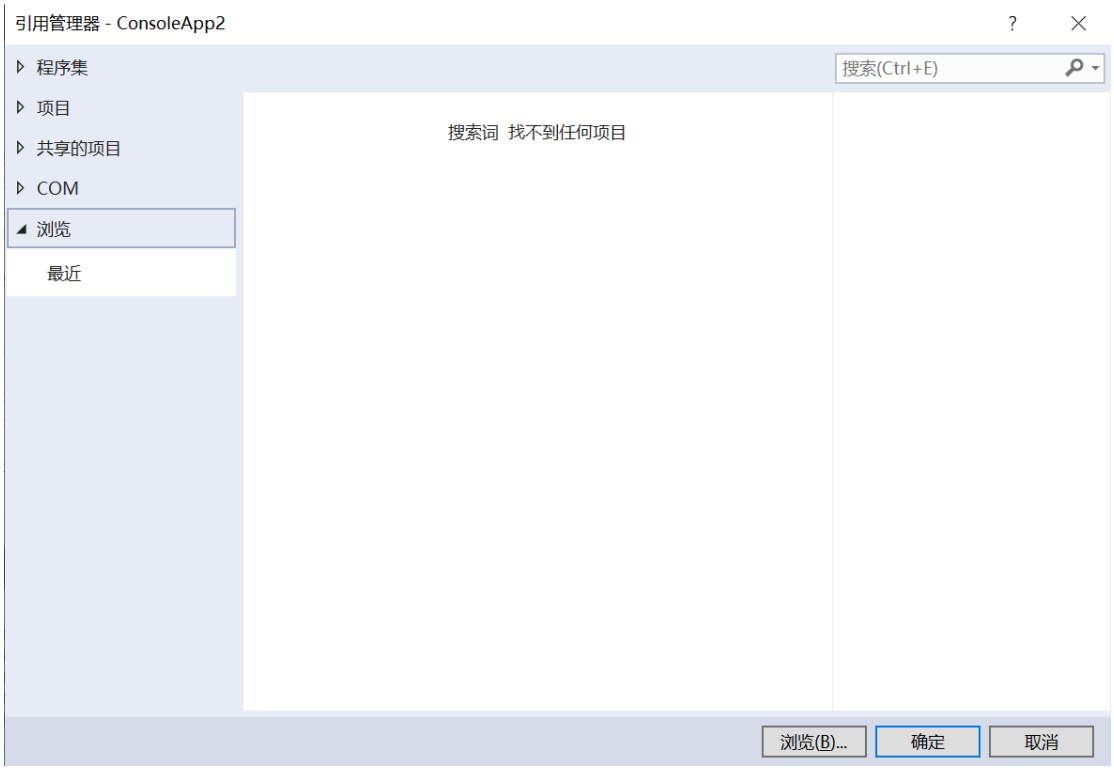
SDK 安装

1） 在 C:\Program Files (x86)\下创建路径 ZOS SDK for .NET\bin\Net45\，将下载好的 C# SDK 解压缩后的所有文件放入此路径下；

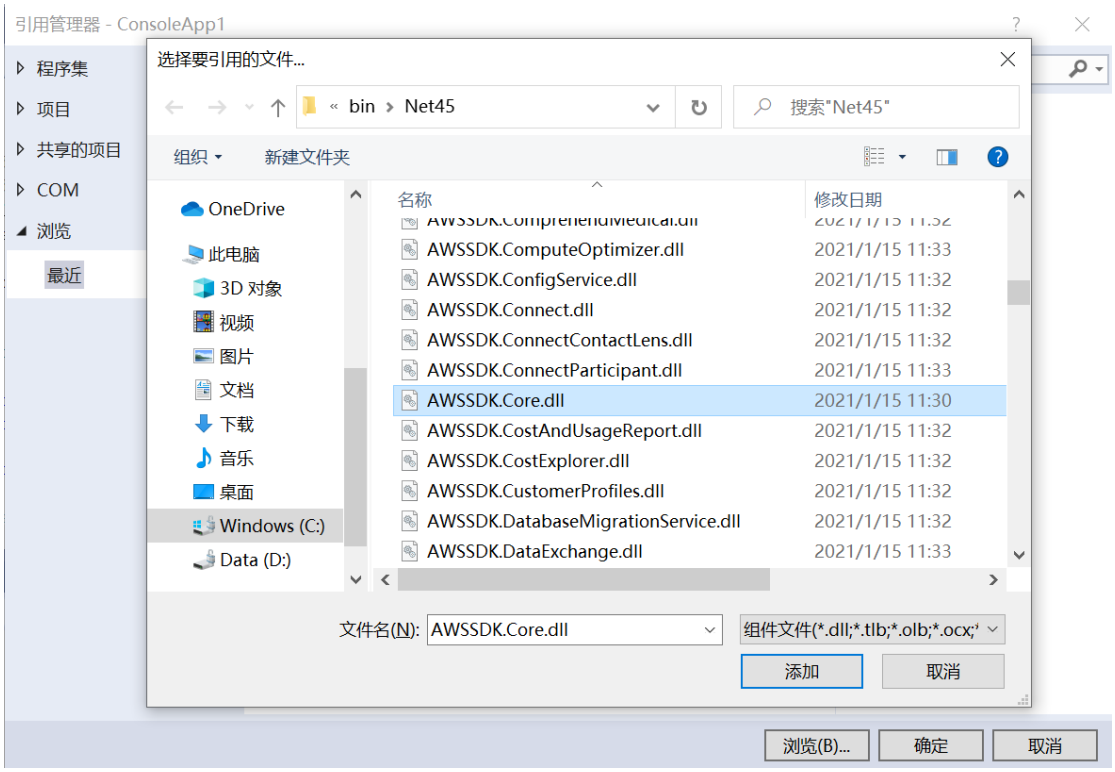
2） C#工程右键->添加->引用

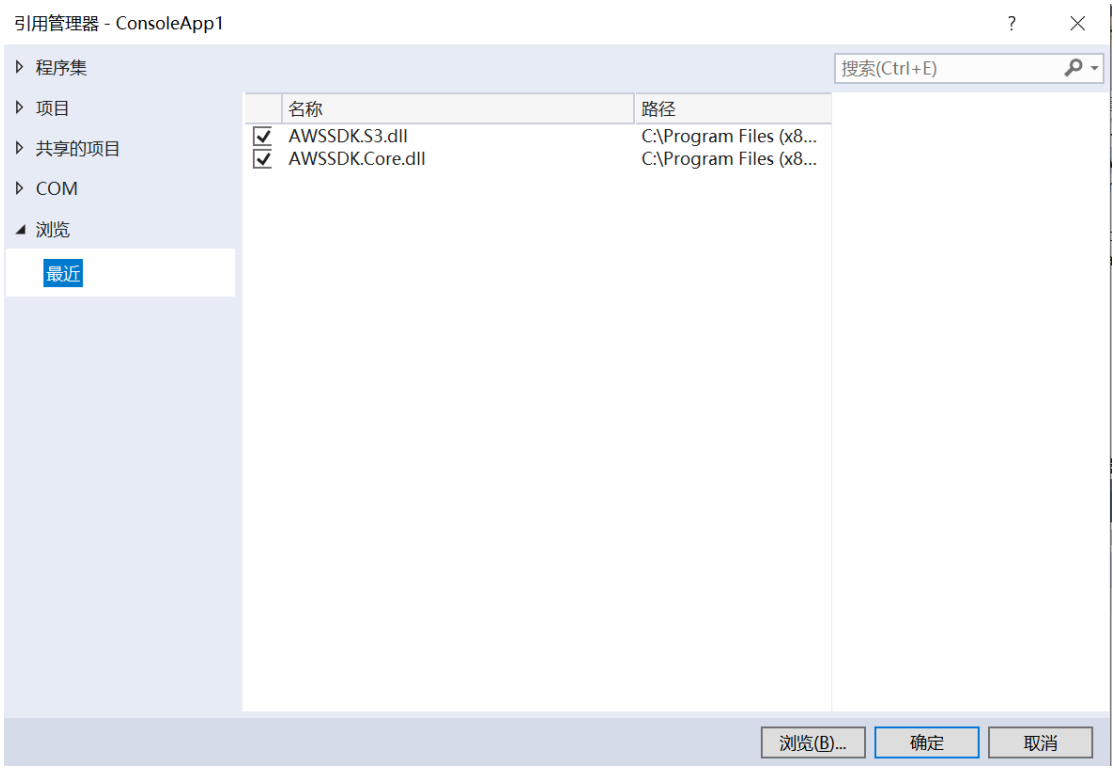


3） 点击右上角”浏览”

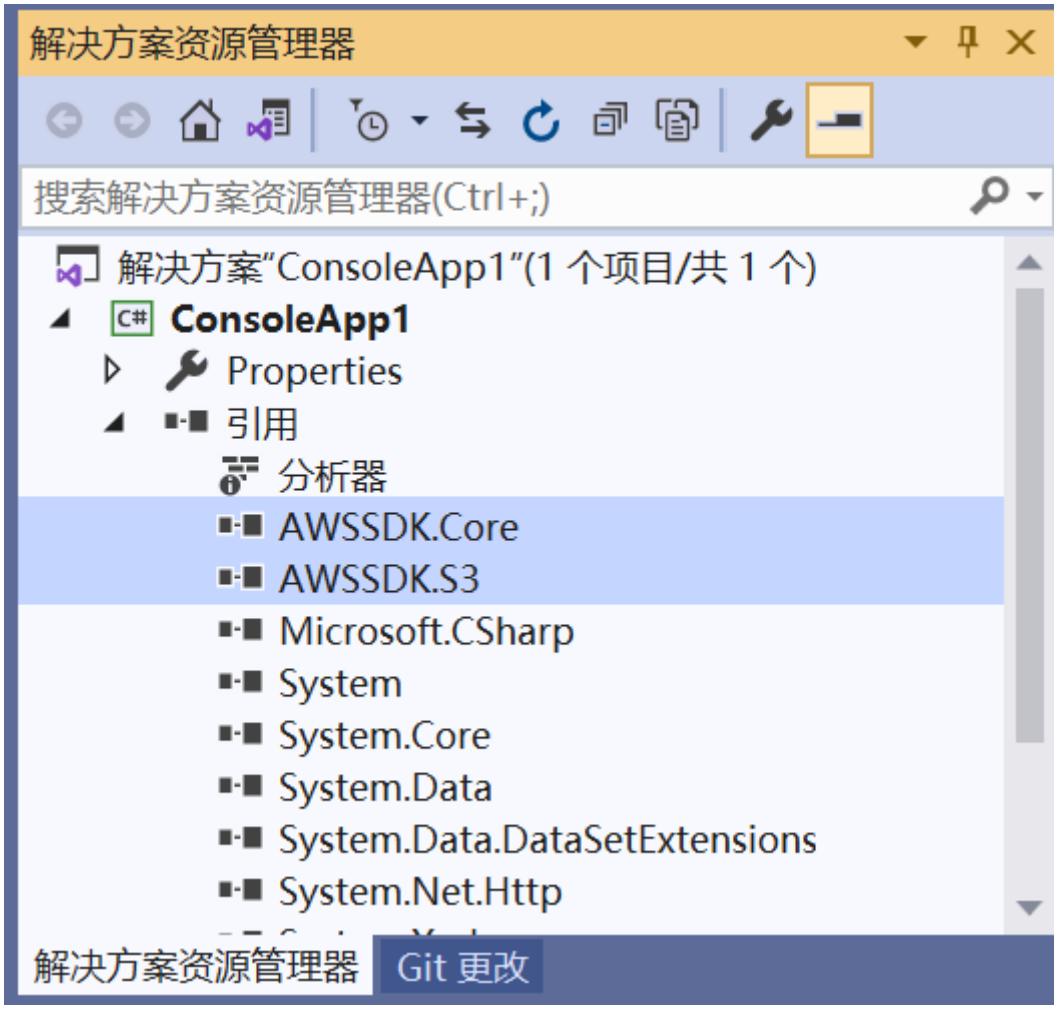


4） 在 C:\Program Files (x86)\ZOS SDK for .NET\bin\Net45\文件夹中找到 AWSSDK.Core.dll 和 AWSSDK.S3.dll，点击添加，两项添加之后确定即可





5） 在工程引用下即可看到 AWSSDK.Core.dll 和 AWSSDK.S3.dll



连接

在正式使用 SDK 接口之前，需要先连接 ZOS，使用以下的示例可以连接 ZOS。

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;

namespace aws_console
{
	class Zos
	{
		public static string accessKeyId = "your access key";
		public static string accessKeySecret = "your secret key";
		public static void Main(string[] args)
		{
			System.Net.ServicePointManager.ServerCertificateValidationC
allback +=
			delegate (
			object sender,
			X509Certificate certificate,
			X509Chain chain,
			SslPolicyErrors sslPolicyErrors) // 自签名忽略认证
		{
			return true;
		};
		AmazonS3Config config = new AmazonS3Config();
		config.ServiceURL = "http://192.168.218.130:7480"; // 指定 IP 和
Port
		config.ForcePathStyle = true; // 使用路径模式
		config.SignatureVersion = "4"; // 签名版本
		var client = new AmazonS3Client(accessKeyId, accessKeySecret,
config); // 建立连接
		}
	}
}
```

全局错误码及类定义

ResonseMetadata 定义方法如下:

```
 //获取本次请求 ID
string RequestId{get;set;}


//获取本次请求的一些元数据
IDictionary<string, string> Metadata { get;}
```

HttpStatusCode

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

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

412	PreconditionFailed
416	InvalidRange
422	UnprocessableEntity
500	InternalError

1、Bucket 操作

1.1、Put Bucket

功能说明

Put Bucket 请求可以在指定账号下创建一个新的 Bucket。

方法原型

```
PutBucketResponse PutBucket(PutBucketRequest request)
```

参数说明

- request:类型 PutBucketRequest 请求接口的参数，具体定义方法如下:

```
// 设置 Bcuekt 的名称，必须设置
string BucketName{get;set;}

//设置 Bucket 的文件锁定功能
bool ObjectLockEnabledForBucket { get; set; }

//设置 Bucket 的 ACL 规则，S3CannedACL 是个 class，内部包含几个 static readonly
类型为 S3CannedACL 的成员变量 NoACL、Private、PublicReadWrite、
PublicReadWrite、AuthenticatedRead 具体用法参考示例

S3CannedACL CannedACL { get; set; }
```

返回结果及说明

- PutBucketResponse:PutBucket 请求接口的返回参数，其具体定义方法如下:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义

HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
```

```
ResponseMetadata ResponseMetadata { get; set; }
```

示例

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                           delegate (
                           object sender,
                           X509Certificate certificate,
                           X509Chain chain,
                           SslPolicyErrors sslPolicyErrors) // 自签名忽
略认证
                           {
                               return true;
                           };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480"; // 指定
IP 和 Port
            config.ForcePathStyle = true; // 使用路径模式
            config.SignatureVersion = "2"; // "4" 签名版本
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接

            PutBucketRequest request = new PutBucketRequest();
            request.BucketName = "bucket-name";
            request.CannedACL = S3CannedACL.NoACL;

            try
            {
                PutBucketResponse response = client.PutBucket(request);
                Console.WriteLine((int)response.HttpStatusCode);
            }
            catch (Exception ex)
            {
                Console.WriteLine("requeset failed!" + ex.Message);
            }
        }
    }
```

```
}
```

1.2、Delete Bucket

功能说明

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

方法原型

```
DeleteBucketResponse DeleteBucket(DeleteBucketRequest request);
```

参数说明

- request:类型是个 class，具体定义方法如下:

```
// 设置要删除的 Bucket 名称，必须设置
string BucketName{get;set;}
```

返回结果说明

- DeleteBucketResponse:类型是个 class，具体定义方法如下:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义
HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }
```

示例

```
using System;
using Amazon.S3;
using Amazon.S3.Model;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;

namespace aws_console
```

```
{
    class Zos
    {
                public static string accessKeyId = "your access key";
            public static string accessKeySecret = "your secret key";
            public static void Main(string[] args)
            {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                            delegate (
                            object sender,
                            X509Certificate certificate,
                            X509Chain chain,
                            SslPolicyErrors sslPolicyErrors) // 自签名忽
略认证
                            {
                                return true;
                            };
               AmazonS3Config config = new AmazonS3Config();
               config.ServiceURL = "http://192.168.218.130:7480"; // 指定
IP 和 Port
               config.ForcePathStyle = true; // 使用路径模式
               config.SignatureVersion = "2"; // "4" 签名版本
               var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接

               DeleteBucketRequest requeset
                   = new DeleteBucketRequest();

               requeset.BucketName = "bucket-name";
               DeleteBucketResponse response;
               try
               {
                   response = client.DeleteBucket(requeset);
                   Console.WriteLine(response);
               } catch (Exception ex) {
                   Console.WriteLine("requeset failed!" + ex.Message);
               }
            }
    }
}
```

1.3、List Bucket

功能说明

List Bucket 可以列出请求用户拥有的所有的 bucket 列表。

方法原型

```
ListBucketsResponse ListBuckets()
```

参数说明

无

返回结果说明

- ListBucketsResponse :类型是个 class，具体定义方法如下:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义

HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }

//列出 Bucket 所属的用户，Owner 是个 Class，可以通过其定义的方法 string
DisplayName { get; set; }和 string Id { get; set; }得到用户的 Displayname 和 Id
Owner Owner { get; set; }

//获取 Bucket 列表，S3Bucket 是个 class，具体定义方法如下
List<S3Bucket> Buckets { get; set; }
```

S3Bucket 的定义方法如下:

```
//获取 Bucket 的创建时间，DateTime 是内置数据类型
DateTime CreationDate { get; set; }

//获取 Bucket 的名称
string BucketName { get; set; }
```

示例

```
using System;
using Amazon.S3;
using Amazon.S3.Model;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using System.Collections.Generic;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
```

```
        public static string accessKeySecret = "your secret key";
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                            delegate (
                            object sender,
                            X509Certificate certificate,
                            X509Chain chain,
                            SslPolicyErrors sslPolicyErrors) // 自签名忽
略认证
                            {
                                return true;
                            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480"; // 指定
IP 和 Port
            config.ForcePathStyle = true; // 使用路径模式
            config.SignatureVersion = "2"; // "4" 签名版本
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接
            ListBucketsResponse response;
            try
            {
                response = client.ListBuckets();
                List<S3Bucket> Buckets = response.Buckets;
                Console.WriteLine(response.Owner.DisplayName);
                Console.WriteLine(response.Owner.Id);
                for (int i = 0; i < Buckets.Count; i++)
                {
                    Console.WriteLine(Buckets[i].BucketName);
                    Console.WriteLine(Buckets[i].CreationDate);
                }
            } catch (Exception ex) {
                Console.WriteLine("requeset failed!" + ex.Message);
            }
        }
    }

}
```

1.4、ListObjects

功能说明

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

方法原型

```
```

```
ListObjectsResponse ListObjects(ListObjectsRequest request);
```

参数说明

- request:类型 ListObjectsRequest，ListObjects 请求接口的参数，具体定义方法如下:

```
// 设置 Bcuekt 的名称
string BucketName{get;set;}

// 设置一次返回的 key 的数量，默认和最大值为 1000
int MaxKeys { get; set; }

// 设置列出指定前缀的 key
string Prefix { get; set; }

//设置返回的 key 的编码方式，合法值为 EncodingType.url
EncodingType Encoding { get; set; }

//设置分组结果时使用的字符
 string Delimiter { get; set; }

//设置列出 key 时的起始 key
 string Marker { get; set; }
```

返回结果及说明

- ListObjectsResponse:ListObjects 请求接口的返回参数，其具体定义方法如下:

```
// 符合条件的 Object 列表
List<S3Object> S3Objects { get; set; }

// 请求的 BucketName
string Name { get; set; }

// 获取请求时指定的 Prefix
string Prefix { get; set; }

// 获取指定的 MaxKeys
int MaxKeys { get; set; }

// 获取请求时指定的 delimiter
string Delimiter { get; set; }

//返回结果是否发生了截断（当超过 MaxKeys 为 true)
bool IsTruncated { get; set; }
```

```
// 当指定了 delimiter 和 Prefix 时，获取根据 Prefix 和 delimieter 得到的公共前缀集合，例如指定 Prefix 为 “notes/” delimiter 为 “/”,则 notes/summer/july,和 notes/summer/august 折叠为一个 CommonPrefix “notes/summer/”。若指定 MaxKey，则折叠后，只 CommonPrefix 占一个计数
List<string> CommonPrefixes { get; set; }

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

S3Object：请求接口的返回参数，其具体定义方法如下：

```
// Bucket Name
string BucketName { get; set; }

// Object Key
string Key { get; set; }

// Object ETag
string ETag { get; set; }

// Object 创建时间
DateTime LastModified { get; set; }

// Object 的所有者，有属性 String DisplayName 和 string Id
Owner Owner { get; set; }

// Object 的大小
long Size { get; set; }

//Object 的存储级别
S3StorageClass StorageClass { get; set; }
```

示例

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;

namespace zos_sdk_demo
{
    class ListObjects
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
```

```
       static void Main(string[] args)
       {
           {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                    delegate (
                    object sender,
                    X509Certificate certificate,
                    X509Chain chain,
                    SslPolicyErrors sslPolicyErrors) // 自签名忽略认证
                    {
                        return true;
                    };
               AmazonS3Config config = new AmazonS3Config();
               config.ServiceURL = "http://192.168.218.130:7480"; //
指定 IP 和 Port
               config.ForcePathStyle = true; // 使用路径模式
               config.SignatureVersion = "2"; // "4" 签名版本
               var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接

               var request = new ListObjectsRequest();
               request.BucketName = "demo-bucket";
               request.MaxKeys = 100;
               request.Encoding = EncodingType.Url;
               request.Prefix = "test";
               request.Delimiter = "/";

               ListObjectsResponse response;
               try
               {
                   response = client.ListObjects(request);

               }
               catch (Exception ex)
               {
                   Console.WriteLine("requeset failed!" + ex.Message);
                   return;
               }

               Console.WriteLine("Bucket: {0}", response.Name);
               foreach (var obj in response.S3Objects)
               {
                   Console.Out.WriteLine("key: {0}, ETag: {1}, size:
{2}, StorageClass: {3}, Owner: {4}, Created Date: {5}"
                       , obj.Key, obj.ETag, obj.Size, obj.StorageClass,
obj.Owner.DisplayName, obj.LastModified);
               }
           }
       }
   }
}
```

1.5、Put Bucket Policy

功能说明

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

方法原型

```
PutBucketPolicyResponse PutBucketPolicy(PutBucketPolicyRequest
request)
```

参数说明

- request: Aws::S3::Model::PutBucketPolicyRequest 类型，该类包含的属性有:

```
//桶名称属性
string BucketName { get; set; }

//桶规则属性，可通过 JSON 格式字符串来设置，具体字段定义看下表
string Policy { get; set; }
```

桶规则 Policy 各字段描述如下：

字段	描述	类型	是否必须
Version	保持与 Amazon S3 一致，当前支持"2012-10-17"和"2006-03-01"	string	否
Id	桶策略 ID，桶策略的唯一标识	string	否
Statement	桶策略描述，定义完整的权限控制。每条桶策略的 Statement 可由多条描述组成，每条描述是一个 dict，每条描述可包含以下字段： Sid Effect Principal Action ReSource Condition	list	是
Sid	本条桶策略描述的 ID	string	否
Effect	桶策略的效果，即指定本条桶策略描述的权限是接受请求还是拒绝请求。 接受请求：配置为“Allow”， 拒绝请求：配置为“Deny”	string	是
Principal	被授权人，即指定本条桶策略描述所作用的用户，支持通配符“*”，表示所有用户。当对某个 user 进行授权时，Principal 格式为 "AWS": "arn:aws:s3:::user/userId"	map	否
Action	操作，即指定本条桶策略描述所作用的 ZOS 操作。以列表形式表示，可配置多条操作，以逗号间隔。支持通配符"*“，表示该资源能进行的所有操作。常用的 Action 有"s3:GetObject"，"s3:GetObjectAcl"，"s3:PutObject"，"s3:PutObjectAcl"等	list	否

Condition	条件语句，指定本条桶策略所限制的条件。可以通过 Condition 对 ZOS 资源设置防盗链，形如： "Condition": {"StringEquals":{"aws:Referer":["www.ctyun.com"]}， 此时如果 Effect 为“Allow”，则允许来自"www.ctyun.com"的请求； 如果为“Deny”，则拒绝。	map	否

返回结果说明

- 返回结果类型为 Aws::S3::Model::PutBucketPolicyResponse，其继承自 Aws::S3::Model::AmazonWebServiceResponse，该类型主要包含以下属性：

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义

HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }
```

示例

```
using System;

using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                            delegate (
                            object sender,
                            X509Certificate certificate,
                            X509Chain chain,
                            SslPolicyErrors sslPolicyErrors) // 自签名忽
略认证
                            {
                                return true;
                            };
            AmazonS3Config config = new AmazonS3Config();
```

```
            config.ServiceURL = "http://192.168.218.130:7480"; // 指定
IP 和 Port
            config.ForcePathStyle = true; // 使用路径模式
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接

            // Put sample bucket policy (overwrite an existing policy)
            string policy = @"{
                ""Statement"":[{
                ""Sid"":""C#Policy"",
                ""Effect"":""Allow"",
                ""Principal"": ""*"",
                ""Action"":[""s3:PutObject"",""s3:GetObject""],
                ""Resource"":[""arn:aws:s3:::bucket1/*""]
                }]}";
            PutBucketPolicyRequest putRequest = new
PutBucketPolicyRequest
            {
                BucketName = "bucket1",
                Policy = policy
            };
            try
            {
                PutBucketPolicyResponse response =
client.PutBucketPolicy(putRequest);
                Console.WriteLine((int)response.HttpStatusCode);
            }
            catch (Exception ex)
            {
                Console.WriteLine("requeset failed!" + ex.Message);
            }
        }
    }
}
```

1.6、Get Bucket Policy

功能说明

Get Bucket Policy 请求为获取设置在一个 bucket 上的策略

方法原型

```
GetBucketPolicyResponse GetBucketPolicy(GetBucketPolicyRequest
request)
```

参数说明

- request: Aws::S3::Model::GetBucketPolicyRequest 类型，该类定义的属性有:

```
// 指定获取哪一个 bucket 的 policy，必须要设置 BucketName

string BucketName { get; set; }
```

返回结果说明

- 返回结果类型为 Aws::S3::Model::GetBucketPolicyResponse，该类型定义的属性有:

```
// 桶策略，json 格式

public string Policy { get; set; }
```

示例

```
using System;

using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                            delegate (
                            object sender,
                            X509Certificate certificate,
                            X509Chain chain,
                            SslPolicyErrors sslPolicyErrors) // 自签名忽略认证
                            {
                                return true;
                            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480"; // 指定 IP 和 Port
            config.ForcePathStyle = true; // 使用路径模式
            var client = new AmazonS3Client(accessKeyId, accessKeySecret, config); // 建立连接

            // Retrieve current policy
```

```
            GetBucketPolicyRequest getRequest = new
GetBucketPolicyRequest
            {
                BucketName = "bucket1"
            };
            string get_policy =
client.GetBucketPolicy(getRequest).Policy;

            Console.WriteLine(get_policy);

        }
    }
}
```

1.7、Delete Bucket Policy

功能说明

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

方法原型

```
DeleteBucketPolicyResponse
DeleteBucketPolicy(DeleteBucketPolicyRequest request)
```

参数说明

- request: Aws::S3::Model::DeleteBucketPolicyRequest 类型，该类定义的属性有:

```
//指定删除哪一个 bucket 的 policy，必须要设置 BucketName
string BucketName { get; set; }
```

返回结果说明

- 返回结果类型为 Aws::S3::Model::DeleteBucketPolicyResponse，其继承自 Aws::S3::Model::AmazonWebServiceResponse，该类型主要包含以下属性：

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义

HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
```

```
ResponseMetadata ResponseMetadata { get; set; }
```

示例

```
using System;

using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                            delegate (
                            object sender,
                            X509Certificate certificate,
                            X509Chain chain,
                            SslPolicyErrors sslPolicyErrors) // 自签名忽
略认证
                            {
                                return true;
                            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480"; // 指定
IP 和 Port
            config.ForcePathStyle = true; // 使用路径模式
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接

            // Delete current policy
            DeleteBucketPolicyRequest deleteRequest = new
DeleteBucketPolicyRequest
            {
                BucketName = "bucket1"
            };
            try
            {
                DeleteBucketPolicyResponse response =
client.DeleteBucketPolicy(deleteRequest);
                Console.WriteLine((int)response.HttpStatusCode);
            }
            catch (Exception ex)
            {
                Console.WriteLine("requeset failed!" + ex.Message);
```

```
            }

        }
    }
}
```

1.8、Put Bucket ACL

功能说明

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

有两种方式设置 ACL，不可同时使用，每次只能给一种参数赋值。其中，设置预定义的固定的 ACL，不能针对特定用户进行授权，且该参数实现的效果，也可以借由另一种方式实现，该参数使用请求头进行传递；AccessControlList 参数方式则可以针对特定用户进行授权，该方式通过请求体传递。该接口会覆盖原有 ACL 属性，包括桶所有者自身的权限，如需保留原有 ACL 属性，应将需要保留的原 ACL 添加到本次操作的授权中（设置预定义的固定 ACL 会默认将桶所有者权限设为 FULL_CONTROL）。

方法原型

```
PutACLResponse PutACL(PutACLRequest request)
```

参数说明

- request: Aws::S3::Model::PutACLRequest 类型，该类包含的属性有:

```
//桶名称属性
string BucketName { get; set; }

// 指定预定 ACL，取值范围为
// Private
// PublicRead
// PublicReadWrite
// AuthenticatedRead
S3CannedACL CannedACL { get; set; }

// 指定 ACL 列表，与预定 ACL 不能同时使用
S3AccessControlList AccessControlList { get; set; }
```

S3AccessControlList 类型，该类包含的属性有:

```
// 桶所有者
Owner Owner { get; set; }

// 授权列表
List<S3Grant> Grants { get; set; }
```

Owner 类型，该类包含的属性有:

```
// 桶所有者 display name
string DisplayName { get; set; }

// 桶所有者用户 ID
string Id { get; set; }
```

S3Grant 类型，该类包含的属性有:

```
// 被授权用户
S3Grantee Grantee { get; set; }
// 被授权权限，取值范围
// READ
// WRITE
// READ_ACP
// WRITE_ACP
// FULL_CONTROL
S3Permission Permission { get; set; }
```

S3Grantee 类型，该类包含的属性有:

```
// 被授权用户类型，取值范围 CanonicalUser、Email、Group
GranteeType Type { get; set; }

// 被授权用户 ID，Type 为 CanonicalUser 必须指定该属性
string CanonicalUser { get; set; }

// 被授权用户 display name
string DisplayName { get; set; }

// 被授权用户邮箱，Type 为 Email 必须指定该属性
string EmailAddress { get; set; }

// 被授权组，Type 为 Group 必须指定该属性，取值范围
// http://acs.amazonaws.com/groups/global/AllUsers
// http://acs.amazonaws.com/groups/global/AuthenticatedUsers
string URI { get; set; }
```

返回结果说明

- PutBucketTaggingResponse，其具体定义方法如下:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义
```

```
HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }
```

示例

```
using System;

using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using System.Collections.Generic;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
        [Obsolete]
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
            delegate (
                object sender,
                X509Certificate certificate,
                X509Chain chain,
                SslPolicyErrors sslPolicyErrors)
            {
                return true;
            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480";
            config.ForcePathStyle = true;
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config);

            string bucket = "bucket-1";

            List<S3Grant> grants = new List<S3Grant>();
            grants.Add(new S3Grant {
                Grantee = new S3Grantee {
                    Type = GranteeType.Email,
                    EmailAddress = "abc@abc.com"
                },
                Permission = S3Permission.READ
            });
            grants.Add(new S3Grant
            {
```

```
                Grantee = new S3Grantee
                {
                    Type = GranteeType.CanonicalUser,
                    CanonicalUser = "test-2"
                },
                Permission = S3Permission.READ
            });
            grants.Add(new S3Grant
            {
                Grantee = new S3Grantee
                {
                    Type = GranteeType.Group,
                    URI =
"http://acs.amazonaws.com/groups/global/AuthenticatedUsers"
                },
                Permission = S3Permission.READ
            });
            PutACLRequest req = new PutACLRequest
            {
                BucketName = bucket,
                // CannedACL = S3CannedACL.PublicReadWrite
                AccessControlList = new S3AccessControlList
                {
                    Owner = new Owner
                    {
                        Id = "test-1"
                    },
                    Grants = grants
                }
            };
            try
            {
                 PutACLResponse res = client.PutACL(req);
                 Console.WriteLine(res.HttpStatusCode);
            }
            catch (Exception ex)
            {
                Console.WriteLine("requeset failed!" + ex.Message);
            }
        }
    }

}
```

1.9、Get Bucket ACL

功能说明

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

方法原型

```
GetACLResponse GetACL(GetACLRequest request)
```

参数说明

- request: Aws::S3::Model::GetACLRequest 类型，该类包含的属性有:

```
//桶名称属性
string BucketName { get; set; }
```

返回结果说明

- GetACLResponse 类型，该类包含的属性有:

```
// ACL 列表
S3AccessControlList AccessControlList { get; set; }
```

S3AccessControlList 类型，该类包含的属性有:

```
// 桶所有者
Owner Owner { get; set; }

// 授权列表
List<S3Grant> Grants { get; set; }
```

Owner 类型，该类包含的属性有:

```
// 桶所有者 display name
string DisplayName { get; set; }

// 桶所有者用户 ID
string Id { get; set; }
```

S3Grant 类型，该类包含的属性有:

```
// 被授权用户
S3Grantee Grantee { get; set; }

// 被授权权限
S3Permission Permission { get; set; }
```

S3Grantee 类型，该类包含的属性有:

```
// 被授权用户类型
GranteeType Type { get; set; }

// 被授权用户 ID
string CanonicalUser { get; set; }
```

```
// 被授权用户 display name
string DisplayName { get; set; }

// 被授权用户邮箱
string EmailAddress { get; set; }

// 被授权组
string URI { get; set; }
```

示例

```
using System;

using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using System.Collections.Generic;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
        [Obsolete]
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
            delegate (
                object sender,
                X509Certificate certificate,
                X509Chain chain,
                SslPolicyErrors sslPolicyErrors)
            {
                return true;
            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480";
            config.ForcePathStyle = true;
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config);

            string bucket = "bucket-1";
            GetACLRequest get_req = new GetACLRequest
            {
                BucketName = bucket,
            };
            try
            {
```

```
                GetACLResponse get_res = client.GetACL(get_req);
                Console.WriteLine(get_res.AccessControlList.Owner.Id);

Console.WriteLine(get_res.AccessControlList.Owner.DisplayName);
                foreach(S3Grant grant in
get_res.AccessControlList.Grants)
                {
                    Console.WriteLine("-------------------------");
                    Console.WriteLine("Type = " + grant.Grantee.Type);
                    if (grant.Grantee.Type == GranteeType.Group)
                    {
                        Console.WriteLine("URI = " + grant.Grantee.URI);
                    }
                    else
                    {
                        Console.WriteLine("CanonicalUser = " +
grant.Grantee.CanonicalUser);
                        Console.WriteLine("DisplayName = " +
grant.Grantee.DisplayName);
                        Console.WriteLine("EmailAddress = " +
grant.Grantee.EmailAddress);
                    }

                    Console.WriteLine("Permission = " +
grant.Permission);
                }           }
            catch (Exception ex)
            {
                Console.WriteLine("requeset failed!" + ex.Message);
            }
        }
    }

}
```

1.10、Put Bucket Lifecycle Configuration

功能说明

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

方法原型

```
PutLifecycleConfigurationResponse PutLifecycleConfiguration(PutLifecyc
leConfigurationRequest request)
```

参数说明

- request:类型 PutLifecycleConfiguration 请求接口的参数，具体定义方法如下:

```
// 设置 Bcuekt 的名称,必须设置
string BucketName {get;set;}

// 设置 Bucket 的 lifecycle 规则容器,必须设置
LifecycleConfiguration  Configuration { get; set; }
```

LifecycleConfiguration 的定义方法如下:

```
// 设置组成 LifecycleConfiguration 规则，必须设置
List<LifecycleRule>  Rules {get;set;}
```

LifecycleRule 的定义方法如下:

```
// 设置当前是否应用生命周期规则，可选值
[LifecycleRuleStatus.Enabled|Lifecycle// RuleStatus.Disabled]，必须设置
LifecycleRuleStatus  Status { get; set; }

// 设置生命周期规则的特征 ID
string  Id { get; set; }

// 设置前缀过滤条件
string  Prefix { get; set; }

// 设置对象的到期删除时间
LifecycleRuleExpiration  Expiration { get; set; }

// 设置历史版本对象到期删除时间
LifecycleRuleNoncurrentVersionExpiration  NoncurrentVersionExpiration { get;
set; }

// 设置生命周期规则的过滤条件
LifecycleFilter  Filter { get; set; }

//设置对象到期转存规则
List<LifecycleTransition>  Transitions { get; set; }

// 设置历史版本对象到期转存规则
List<LifecycleRuleNoncurrentVersionTransition>  NoncurrentVersionTransitions
{ get; set; }

//设置一次分段上传最多持续时间
LifecycleRuleAbortIncompleteMultipartUpload  AbortIncompleteMultipartUpload
{get;set;}
```

LifecycleRuleExpiration 的定义方法如下:

```
// 设置对象的到期删除时间，日期为 ISO8601 格式，必须为 UTC 午夜 0 时
DateTime  Date {get;set;}

// 设置对象受规则约束的天数，与 Date 只能设置一个
int  Days { get; set; }
```

```
// 设置是否删除“删除标记”
bool  ExpiredObjectDeleteMarker { get; set; }
```

LifecycleRuleNoncurrentVersionExpiration 的定义方法如下:

```
// 设置历史版本受规则约束的天数
int  NoncurrentDays {get;set;}
```

LifecycleFilter 的定义方法如下:

```
// 设置筛选条件，LifecycleFilterPredicate 是个抽象类，具体实现类包括 Lifecycle
// AndOperator，LifecyclePrefixPredicate，LifecycleTagPredicate

LifecycleFilterPredicate  LifecycleFilterPredicate {get;set;}
```

LifecycleFilterPredicate 的实现类定义方法：

```
// LifecyclePrefixPredicate 的定义方法
string  Prefix  {  get;  set;  }

// LifecycleTagPredicate 的定义方法
Tag  Tag  {  get;  set;  }

// Tag 的定义方法，设置 tag 的 key
string  Key  {  get;  set;  }

// Tag 的定义方法，设置 tag 的 value
string  Value  {  get;  set;  }

// LifecycleAndOperator 的定义方法，可以包含一个 LifecyclePrefixPredicate 或多
// 个 LifecycleTagPredicate
List<LifecycleFilterPredicate>  Operands  {  get;  set;  }
```

LifecycleTransition 的定义方法如下:

```
// 设置对象的转存时间，日期为 ISO8601 格式，必须为 UTC 午夜 0 时
DateTime  Date {get;set;}


// 设置对象受规则约束的天数，与 Date 只能设置一个
int  Days { get; set; }

// 设置转存的存储级别，共有 3 个级别，S3StorageClass.STANDARD(标准型),S3Storag
// eClass.INFREQUENT-ACCESS(低频型),S3StorageClass.ARCHIVE(归档型，该存储级别
/// 暂未启用)
S3StorageClass  StorageClass { get; set; }
```

LifecycleRuleNoncurrentVersionTransition 的定义方法如下:

```


// 设置历史版本对象受规则约束的天数
int  NoncurrentDays { get; set; }
```

```
// 设置转存的存储级别，共有 3 个级别，S3StorageClass.STANDARD(标准型),S3Storag
// eClass.INFREQUENT-ACCESS(低频型),S3StorageClass.ARCHIVE(归档型，该存储级别
暂// 未启用)
S3StorageClass  StorageClass { get; set; }
```

LifecycleRuleAbortIncompleteMultipartUpload 的定义方法如下:

```
// 设置分段上传最大持续天数
int  DaysAfterInitiation {get;set;}
```

返回结果及说明

- PutLifecycleConfigurationResponse:PutLifecycleConfiguration 请求接口的返回参数，返回参数无具体意义字段，无需处理。

示例

```
using System;
using Amazon.S3;
using Amazon.S3.Model;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using System.Collections.Generic;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                           delegate (
                           object sender,
                           X509Certificate certificate,
                           X509Chain chain,
                           SslPolicyErrors sslPolicyErrors) // 自签名忽
略认证
                           {
                               return true;
                           };
           AmazonS3Config config = new AmazonS3Config();
           config.ServiceURL = "http://192.168.168.130:7480"; // 指定
IP 和 Port
           config.ForcePathStyle = true; // 使用路径模式
           config.SignatureVersion = "2"; // "4" 签名版本
           var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接

           PutLifecycleConfigurationRequest putlcrequest = new
PutLifecycleConfigurationRequest
```

```
            {
                BucketName = "test",
                Configuration = new LifecycleConfiguration
                {
                    Rules = new List<LifecycleRule>
                    {
                        new LifecycleRule
                        {
                            Expiration = new LifecycleRuleExpiration
                            {
                                Days = 365
                            },
                            Id = "for c#",
                            Status = LifecycleRuleStatus.Enabled,
                            Filter = new LifecycleFilter
                            {
                                LifecycleFilterPredicate = new
LifecycleAndOperator
                                {
                                    Operands = new
List<LifecycleFilterPredicate>
                                    {
                                        new LifecycleTagPredicate
                                        {
                                            Tag = new Tag
                                            {
                                                Key = "tag1",
                                                Value = "val1"
                                            }
                                        },
                                        new LifecycleTagPredicate
                                        {
                                            Tag = new Tag
                                            {
                                                Key = "tag2",
                                                Value = "val2"
                                            }
                                        }
                                    }
                                }
                            }

                        }
                    }
                }
            };

            try
            {
                client.PutLifecycleConfiguration(putlcrequest);

            }
            catch (Exception ex)
            {
                Console.WriteLine("requeset failed!" + ex.Message);
            }
```

```
        }
    }
}
```

1.11、Get Bucket Lifecycle Configuration

功能说明

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

方法原型

```
GetLifecycleConfigurationResponse GetLifecycleConfiguration(GetLifecyc
leConfigurationRequest request)
```

参数说明

- request:类型 GetLifecycleConfiguration 请求接口的参数，具体定义方法如下:

```
// 设置 Bcuekt 的名称,必须设置

string BucketName {get;set;}
```

返回结果及说明

- GetLifecycleConfigurationResponse:GetLifecycleConfiguration 请求接口的返回参数，其具体定义方法如下:

```
// 获取指定桶的生命周期规则，LifecycleConfiguration 具体定义方法同 1.10 参数说
// 明

LifecycleConfiguration  Configuration{get;set;}
```

示例

```
using System;
using Amazon.S3;
using Amazon.S3.Model;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using System.Collections.Generic;

namespace aws_console
{
    class Zos
```

```
   {
       public static string accessKeyId = "your access key";
       public static string accessKeySecret = "your secret key";
public static void Main(string[] args)
       {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                        delegate (
                        object sender,
                        X509Certificate certificate,
                        X509Chain chain,
                        SslPolicyErrors sslPolicyErrors) // 自签名忽
略认证
                        {
                            return true;
                        };
           AmazonS3Config config = new AmazonS3Config();
           config.ServiceURL = "http://192.168.168.130:7480"; // 指定
IP 和 Port
           config.ForcePathStyle = true; // 使用路径模式
           config.SignatureVersion = "2"; // "4" 签名版本
           var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接
           GetLifecycleConfigurationRequest getlcrequest = new
GetLifecycleConfigurationRequest
           {
               BucketName = "bk01"
           };

           try
           {
               GetLifecycleConfigurationResponse response =
client.GetLifecycleConfiguration(getlcrequest);

               for (int i = 0; i < response.Configuration.Rules.Count;
i++)
               {
                   Console.WriteLine("Rule Start!");
                   Console.WriteLine("Id: " +
response.Configuration.Rules[i].Id);
                   Console.WriteLine("Status: " +
response.Configuration.Rules[i].Status);
                   Console.WriteLine("Expiration Days: " +
response.Configuration.Rules[i].Expiration.Days);
                   LifecycleFilter filter =
response.Configuration.Rules[i].Filter;
                   LifecyclePrefixPredicate prefixPredicate;
                   List<LifecycleTagPredicate> tagPredicates = new
List<LifecycleTagPredicate>();
                   if (filter != null)
                   {
                       if
(filter.LifecycleFilterPredicate.ToString().Contains("LifecyclePrefixP
redicate"))
                       {
```

```
                        prefixPredicate =
(LifecyclePrefixPredicate)filter.LifecycleFilterPredicate;
                        Console.WriteLine("Prefix: " +
prefixPredicate.Prefix);
                    }
                    else if
(filter.LifecycleFilterPredicate.ToString().Contains("LifecycleTagPred
icate"))
                    {

tagPredicates.Add((LifecycleTagPredicate)filter.LifecycleFilterPredica
te);
                    }
                    else if
(filter.LifecycleFilterPredicate.ToString().Contains("LifecycleAndOper
ator"))
                    {
                        LifecycleAndOperator andoperator =
(LifecycleAndOperator)filter.LifecycleFilterPredicate;
                        List<LifecycleFilterPredicate>
current_operands = andoperator.Operands;
                        for (int j = 0; j < current_operands.Count;
j++)
                        {
                            if
(current_operands[j].ToString().Contains("LifecyclePrefixPredicate"))
                            {
                                prefixPredicate =
(LifecyclePrefixPredicate)current_operands[j];
                                Console.WriteLine("Prefix" +
prefixPredicate.Prefix);
                            }
                            if
(current_operands[j].ToString().Contains("LifecycleTagPredicate"))
                            {

tagPredicates.Add((LifecycleTagPredicate)current_operands[j]);
                            }
                        }
                    }
                    for (int k = 0; k < tagPredicates.Count; k++)
                    {
                        Console.WriteLine("Key:" +
tagPredicates[k].Tag.Key);
                        Console.WriteLine("Value:" +
tagPredicates[k].Tag.Value);
                    }
                }
                else
                {
                    Console.WriteLine("Prefix: " +
response.Configuration.Rules[i].Prefix);
                }
                Console.WriteLine("Rule End!\n");
            }
        }
```

```
            catch (Exception ex)
            {
                Console.WriteLine("requeset failed!" + ex.Message);
            }
        }
    }
}
```

1.12、Delete Bucket Lifecycle

功能说明

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

方法原型

```
DeleteLifecycleConfigurationResponse DeleteLifecycleConfiguration(Dele
teLifecycleConfigurationRequest request)
```

参数说明

- request:类型 DeleteLifecycleConfiguration 请求接口的参数，具体定义方法如下:

```
// 设置 Bcuekt 的名称,必须设置

string BucketName {get;set;}
```

返回结果及说明

- DeleteLifecycleConfigurationResponse:DeleteLifecycleConfiguration 请求接口的返回参数，返回参数无具体意义字段，无需处理。

示例

```
using System;
using Amazon.S3;
using Amazon.S3.Model;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using System.Collections.Generic;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
```

```
       public static string accessKeySecret = "your secret key";
public static void Main(string[] args)
       {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                           delegate (
                           object sender,
                           X509Certificate certificate,
                           X509Chain chain,
                           SslPolicyErrors sslPolicyErrors) // 自签名忽
略认证
                           {
                               return true;
                           };
           AmazonS3Config config = new AmazonS3Config();
           config.ServiceURL = "http://192.168.168.130:7480"; // 指定
IP 和 Port
           config.ForcePathStyle = true; // 使用路径模式
           config.SignatureVersion = "2"; // "4" 签名版本
           var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接
           DeleteLifecycleConfigurationRequest dellcrequest = new
DeleteLifecycleConfigurationRequest
           {
               BucketName = "test"
           };

           try
           {
               client.DeleteLifecycleConfiguration(dellcrequest);
           }
           catch (Exception ex)
           {
               Console.WriteLine("requeset failed!" + ex.Message);
           }
       }
   }

}
```

1.13、Put Bucket Website

功能说明

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

方法原型

```

PutBucketWebsiteResponse PutBucketWebsite(PutBucketWebsiteRequest
request)
```

参数说明

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

```
//静态网站关联的存储桶名称，必须设置
string BucketName { get; set; }

//静态网站配置参数的容器
WebsiteConfiguration WebsiteConfiguration { get; set; }
```

WebsiteConfiguration 类的方法如下：

```
//错误文档配置
string ErrorDocument { get; set; }

//索引文档配置
string IndexDocumentSuffix { get; set; }

//重定向所有请求配置，该规则与其他规则互斥，也就是说使用了重定向规则就不能配置其他规则
RoutingRuleRedirect RedirectAllRequestsTo { get; set; }

//重定向规则配置列表
List<RoutingRule> RoutingRules { get; set; }
```

RoutingRuleRedirect 类方法如下：

```
//重定向机器名
string HostName { get; set; }

//http 返回码规则
string HttpRedirectCode { get; set; }

//重定向请求要用的协议，默认使用原请求所是应用的协议。
string Protocol { get; set; }

//指定重定向规则的具体重定向目标的对象键，替换方式为替换原始请求中所匹配到的前//缀部分，仅可在 Condition 为 KeyPrefixEquals 时设置,ReplaceKeyWith 与 //ReplaceKeyPrefixWith 必选其一
string ReplaceKeyPrefixWith { get; set; }

//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,ReplaceKeyWith 与 ReplaceKeyPrefixWith 必选其一
string ReplaceKeyWith { get; set; }
```

```

```

RoutingRule 类方法如下：

```
//重定向规则的条件配置
RoutingRuleCondition Condition { get; set; }

//重定向规则，可以配置规则重定向其他主机、页面或其他协议，当发生错误时，也可以//配置错误码。RoutingRules 中的必要配置。
RoutingRuleRedirect Redirect { get; set; }
```

RoutingRuleCondition 类方法如下：

```
//指定重定向规则的错误码匹配条件，只支持配置 4XX 返回码，例如 403 或 404。当、、//condition 配置后，HttpErrorCodeReturnedEquals 和 KeyPrefixEquals 两者只//能配置一个。
string HttpErrorCodeReturnedEquals { get; set; }

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

返回结果说明

- PutBucketWebsiteResponse :PutBucketWebsite 请求接口的返回参数，其具体定义方法如下:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考 HttpStatusCode 定义

HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考 ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }
```

示例

```
using System;
using System.Collections.Generic;
using System.Linq;
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
```

```
namespace sdk_test
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
        static void Main(string[] args)
        {
            Console.WriteLine("hello world");

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                delegate (
                object sender,
                X509Certificate certificate,
                X509Chain chain,
                SslPolicyErrors sslPolicyErrors) // 自签名忽略认证
                {
                    return true;
                };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480"; // 指定
IP 和 Port
            config.ForcePathStyle = true; // 使用路径模式
            config.SignatureVersion = "2"; // "4" 签名版本
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接
            PutBucketWebsiteRequest req = new PutBucketWebsiteRequest
            {
                BucketName = "rgwuser01-testbucket03",
                WebsiteConfiguration = new WebsiteConfiguration
                {
                    ErrorDocument = "err.html",
                    IndexDocumentSuffix = "index.html",
                    RedirectAllRequestsTo = new RoutingRuleRedirect
                    {
                        HostName = "adfada",
                        Protocol = "https",
                    }
                }
            };

            PutBucketWebsiteResponse res =
client.PutBucketWebsite(req);
        }
    }
}
```

1.14、Get Bucket Website

功能说明

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

方法原型

```
GetBucketWebsiteResponse GetBucketWebsite(GetBucketWebsiteRequest
request)
```

参数说明

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

```
//存储桶名称，必须设置
string BucketName { get; set; }
```

返回结果说明

- 返回结果为 GetBucketWebsiteResponse 类型,具体方法如下：

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义

HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }

//静态网站配置参数的容器,其具体方法参见 Put Bucket WebSite 部分的参数说明
WebsiteConfiguration WebsiteConfiguration { get; set; }
```

示例

```
using System;
using System.Collections.Generic;
using System.Linq;
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
namespace sdk_test
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
```

```
       static void Main(string[] args)
       {
           Console.WriteLine("hello world");

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
               delegate (
               object sender,
               X509Certificate certificate,
               X509Chain chain,
               SslPolicyErrors sslPolicyErrors) // 自签名忽略认证
               {
                   return true;
               };
           AmazonS3Config config = new AmazonS3Config();
           config.ServiceURL = "http://192.168.218.130:7480"; // 指定
IP 和 Port
           config.ForcePathStyle = true; // 使用路径模式
           config.SignatureVersion = "2"; // "4" 签名版本
           var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接
           GetBucketWebsiteRequest req = new GetBucketWebsiteRequest
           {
               BucketName = "rgwuser01-testbucket03"
           };
           GetBucketWebsiteResponse res =
client.GetBucketWebsite(req);
           Console.WriteLine(res.WebsiteConfiguration.ErrorDocument);

Console.WriteLine(res.WebsiteConfiguration.IndexDocumentSuffix);

Console.WriteLine(res.WebsiteConfiguration.RedirectAllRequestsTo.HostN
ame);

Console.WriteLine(res.WebsiteConfiguration.RedirectAllRequestsTo.Proto
col);

       }
   }
}
```

1.15、Delete Bucket Website

功能说明

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

方法原型

```
```

```
DeleteBucketWebsiteResponse
DeleteBucketWebsite(DeleteBucketWebsiteRequest request)
```

参数说明

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

```
//存储桶名称，必须设置
string BucketName { get; set; }
```

返回结果说明

- PutBucketWebsiteResponse :PutBucketWebsite 请求接口的返回参数，其具体定义方法如下：

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义
HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }
```

示例

```
using System;
using System.Collections.Generic;
using System.Linq;
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;


namespace sdk_test
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
        static void Main(string[] args)
        {
            Console.WriteLine("hello world");

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                delegate (
                object sender,
```

```
                X509Certificate certificate,
                X509Chain chain,
                SslPolicyErrors sslPolicyErrors) // 自签名忽略认证
                {
                    return true;
                };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480"; // 指定
IP 和 Port
            config.ForcePathStyle = true; // 使用路径模式
            config.SignatureVersion = "2"; // "4" 签名版本
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接
            DeleteBucketWebsiteRequest req = new
DeleteBucketWebsiteRequest
            {
                BucketName = "rgwuser01-testbucket03"
            };
            DeleteBucketWebsiteResponse res =
client.DeleteBucketWebsite(req);

            Console.Read();
        }
    }
}
```

1.16、Put Bucket Request Payment

功能说明

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

方法原型

```
PutBucketRequestPaymentResponse PutBucketRequestPayment(
        PutBucketRequestPaymentRequest request)
```

参数说明

- request: Aws::S3::Model::PutBucketRequestPaymentRequest 类型，该类定义的属性有:

```
//BucketName 属性
string BucketName { get; set; }
```

```
//设置请求付费配置，RequestPaymentConfiguration 是个 class 类型，具体定义如下
RequestPaymentConfiguration RequestPaymentConfiguration { get; set; }
```

RequestPaymentConfiguration 定义属性如下:

```
//RequestPaymentConfiguration 类中封装有 Payer 属性：
string Payer { get; set; }
```

返回结果说明

- 返回结果类型为 Aws::S3::Model::PutBucketRequestPaymentResponse，其继承自 Aws::S3::Model::AmazonWebServiceResponse，该类型主要包含以下属性：

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义

HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }
```

示例

```
using System;

using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                            delegate (
                            object sender,
                            X509Certificate certificate,
                            X509Chain chain,
                            SslPolicyErrors sslPolicyErrors) // 自签名忽
略认证
                            {
                                return true;
```

```
                        };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480"; // 指定
IP 和 Port
            config.ForcePathStyle = true; // 使用路径模式
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接

            RequestPaymentConfiguration payment_config = new
RequestPaymentConfiguration
            {
                Payer = "Requester"
            };
            PutBucketRequestPaymentRequest request = new
PutBucketRequestPaymentRequest
            {
                BucketName = "bucket1",
                RequestPaymentConfiguration = payment_config
            };
            try
            {
                PutBucketRequestPaymentResponse response =
client.PutBucketRequestPayment(request);
                Console.WriteLine((int)response.HttpStatusCode);
            }
            catch (Exception e)
            {
                Console.WriteLine("put bucket request payment failed!"
+ e.Message);
            }
        }
    }
}
```

1.17、Get Bucket Request Payment

功能说明

返回 bucket 的请求支付配置。

方法原型

```

GetBucketRequestPaymentResponse GetBucketRequestPayment(
        GetBucketRequestPaymentRequest request)
```

参数说明

- request: Aws::S3::Model::GetBucketRequestPaymentRequest 类型，该类定义的属性有:

```
//BucketName 属性
string BucketName { get; set; }
```

返回结果说明

- 返回结果类型为 Aws::S3::Model::GetBucketRequestPaymentResponse 该类型主要包含以下属性：

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义

HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }

// Payer，即具体的请求者
string Payer { get; set; }
```

示例

```
using System;

using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                            delegate (
                            object sender,
                            X509Certificate certificate,
                            X509Chain chain,
                            SslPolicyErrors sslPolicyErrors) // 自签名忽
略认证
                            {
                                return true;
                            };
            AmazonS3Config config = new AmazonS3Config();
```

```
            config.ServiceURL = "http://192.168.218.130:7480"; // 指定
IP 和 Port
            config.ForcePathStyle = true; // 使用路径模式
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接

            GetBucketRequestPaymentRequest request = new
GetBucketRequestPaymentRequest
            {
                BucketName = "bucket1"
            };
            try
            {
                GetBucketRequestPaymentResponse res =
client.GetBucketRequestPayment(request);
                string payer = res.Payer;
                Console.WriteLine("Payer: " + payer);
            } catch (Exception e)
            {
                Console.WriteLine("get bucket request payment failed!"
+ e.Message);
            }
        }
    }
}
```

1.18、Put Bucket Tagging

功能说明

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

方法原型

```
PutBucketTaggingResponse PutBucketTagging(PutBucketTaggingRequest
request)
```

参数说明

- request: Aws::S3::Model::PutBucketTaggingRequest 类型，该类定义的属性有:

```
// 桶名称
string BucketName { get; set; }
```

```
// 标签集
List<Tag> TagSet {get; set;}
```

Tag 类型，该类定义的属性有：

```
// 标签 Key，最大 128 字节
string Key {get; set;}

// 标签 value， 最大 256 字节
string Value {get; set;}
```

返回结果说明

- PutBucketTaggingResponse，其具体定义方法如下：

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义
HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }
```

示例

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using System.Collections.Generic;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";

        [Obsolete]
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
            delegate (
                object sender,
                X509Certificate certificate,
                X509Chain chain,
                SslPolicyErrors sslPolicyErrors)
```

```
            {
                return true;
            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480";
            config.ForcePathStyle = true;
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config);

            string bucket = "bucket-1";
            List<Tag> tags = new List<Tag>();
            tags.Add(new Tag
            {
                Key = "key1",
                Value = "value1"
            });
            PutBucketTaggingRequest tag_req = new
PutBucketTaggingRequest
            {
                BucketName = bucket,
                TagSet = tags
            };

            try
            {
                 PutBucketTaggingResponse tag_res =
client.PutBucketTagging(tag_req);
                 Console.WriteLine(tag_res.HttpStatusCode);
            }
            catch (Exception ex)
            {
                Console.WriteLine("requeset failed!" + ex.Message);
            }
        }
    }

}
```

1.19、Get Bucket Tagging

功能说明

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

方法原型

```
GetBucketTaggingResponse GetBucketTagging(GetBucketTaggingRequest
request)
```

参数说明

- request: Aws::S3::Model::GetBucketTaggingRequest 类型，该类定义的属性有:

```
// 桶名称
string BucketName { get; set; }
```

返回结果说明

- GetBucketTaggingResponse 类型，该类定义的属性有:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义

HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }

// 标签集
List<Tag> TagSet { get; set; }
```

Tag 类型，该类定义的属性有:

```
// 标签 Key

string Key {get; set;}

// 标签 value
string Value {get; set;}
```

示例

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using System.Collections.Generic;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";

        [Obsolete]
        public static void Main(string[] args)
        {
```

```
System.Net.ServicePointManager.ServerCertificateValidationCallback +=
          delegate (
              object sender,
              X509Certificate certificate,
              X509Chain chain,
              SslPolicyErrors sslPolicyErrors)
          {
              return true;
          };
          AmazonS3Config config = new AmazonS3Config();
          config.ServiceURL = "http://192.168.218.130:7480";
          config.ForcePathStyle = true;
          var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config);

          string bucket = "bucket-1";
          GetBucketTaggingRequest get_tag_req =
              new GetBucketTaggingRequest
          {
              BucketName = bucket
          };

          try
          {
              GetBucketTaggingResponse get_tag_res =
client.GetBucketTagging(get_tag_req);
              Console.WriteLine(get_tag_res.HttpStatusCode);
              foreach(Tag tag in get_tag_res.TagSet)
              {
                  Console.WriteLine(tag.Key + "=" + tag.Value);
              }
          }
          catch (Exception ex)
          {
              Console.WriteLine("requeset failed!" + ex.Message);
          }
      }
  }
}
```

1.20、Delete Bucket Tagging

功能说明

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

方法原型

```
```

```
DeleteBucketTaggingResponse
DeleteBucketTagging(DeleteBucketTaggingRequest request)
```

参数说明

- request: Aws::S3::Model::DeleteBucketTaggingRequest 类型，该类定义的属性有:

```
// 桶名称

string BucketName { get; set; }
```

返回结果说明

- DeleteBucketTaggingResponse，其具体定义方法如下:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义

HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }
```

示例

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using System.Collections.Generic;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";

        [Obsolete]
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
            delegate (
                object sender,
                X509Certificate certificate,
                X509Chain chain,
```

```
                SslPolicyErrors sslPolicyErrors)
            {
                return true;
            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480";
            config.ForcePathStyle = true;
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config);

            string bucket = "bucket-1";
            DeleteBucketTaggingRequest del_tag_req
                = new DeleteBucketTaggingRequest
            {
                BucketName = bucket
            };
            try
            {
                DeleteBucketTaggingResponse del_tag_res =
client.DeleteBucketTagging(del_tag_req);
                Console.WriteLine(del_tag_res.HttpStatusCode);
            }
            catch (Exception ex)
            {
                Console.WriteLine("requeset failed!" + ex.Message);
            }

            try
            {
                GetBucketTaggingResponse get_tag_res =
client.GetBucketTagging(get_tag_req);
                Console.WriteLine(get_tag_res.HttpStatusCode);
                foreach (Tag tag in get_tag_res.TagSet)
                {
                    Console.WriteLine(tag.Key + "=" + tag.Value);
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine("requeset failed!" + ex.Message);
            }
        }
    }

}
```

1.21、Put Bucket Encrytion

功能说明

put bucket encryption 请求可以启用存储桶默认加密功能

方法原型

```
PutBucketEncryptionResponse
PutBucketEncryption(PutBucketEncryptionRequest request)
```

参数说明

- request:类型 PutBucketEncryptionRequest 请求接口的参数，具体定义方法如下:

```
// 设置 Bcuekt 的名称，必须设置
string BucketName{get;set;}

//设置 Bucket 的默认加密功能
ServerSideEncryptionConfiguration ServerSideEncryptionConfiguration { get; set; }
```

类型 ServerSideEncryptionConfiguration，设置 Bucket 默认加密配置，定义的方法如下:

```
//设置 Bucket 的默认加密配置
List<ServerSideEncryptionRule> ServerSideEncryptionRules { get; set; }
```

类型 ServerSideEncryptionRule，设置 Bucket 默认加密规则，定义的方法如下:

```
//设置 Bucket 的默认加密规则
ServerSideEncryptionByDefault ServerSideEncryptionByDefault { get; set; }
```

类型 ServerSideEncryptionByDefault，设置 Bucket 默认加密，定义的方法如下:

```
//设置 Bucket 的默认加密算法，可以是 AES256 或 AWSKMS
ServerSideEncryptionMethod ServerSideEncryptionAlgorithm { get; set; }

//设置 Bucket 的默认加密密钥，当算法是 AES256 时，可以调用此函数设置加密密钥，但字符长度需为 32，也可以不设置加密密钥，系统会自动生成；当算法是 AWSKMS 时，此项必须设置，且按照 cmkuuid:keyspec:userid 模式配置，其中 cmkuuid 是 CMKID，keyspec 是指定生成的数据密钥长度，userid 是用户 id。
string ServerSideEncryptionKeyManagementServiceKeyId { get; set; }
```

返回结果说明

- PutBucketEncryptionResponse:PutBucketEncryption 请求接口的返回参数，其具体定义方法如下:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义
HttpStatusCode HttpStatusCode{get;set;}
```

```
//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }
```

示例

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using System.Collections.Generic;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";


        [Obsolete]
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
            delegate (
                object sender,
                X509Certificate certificate,
                X509Chain chain,
                SslPolicyErrors sslPolicyErrors)
            {
                return true;
            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480";
            config.ForcePathStyle = true;
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config);
            Zos.PutBucketEncryptionCallBack(client, "bucket1");
        }

        public static void PutBucketEncryptionCallBack(AmazonS3Client
client, string bucketName)
        {
            ServerSideEncryptionConfiguration Configuration = new
ServerSideEncryptionConfiguration
            {
                ServerSideEncryptionRules = new
List<ServerSideEncryptionRule>
                {
                    new ServerSideEncryptionRule
                    {
```

```
                    ServerSideEncryptionByDefault = new
ServerSideEncryptionByDefault
                    {
                        ServerSideEncryptionAlgorithm = new
ServerSideEncryptionMethod("aws:kms"),

ServerSideEncryptionKeyManagementServiceKeyId = "6b1f657c-816b-4534-
a41a-903e7a60e703:AES_256:e3d16fba6ae84e33a1d386dd880696c0"
                    }
                }
            }
        };
        PutBucketEncryptionRequest request = new
PutBucketEncryptionRequest
        {
            BucketName = bucketName,
            ServerSideEncryptionConfiguration = Configuration
        };
        try
        {
            PutBucketEncryptionResponse response =
client.PutBucketEncryption(request);
            Console.WriteLine("ZOS 存储桶默认加密启动成功! ");
        }
        catch (Exception ex)
        {
            Console.WriteLine("put bucket encryption failed!" +
ex.Message);
        }
    }
  }
}
```

1.22、Get Bucket Encrytion

功能说明

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

方法原型

```
GetBucketEncryptionResponse
GetBucketEncryption(GetBucketEncryptionRequest request)
```

参数说明

- request:类型 GetBucketEncryptionRequest 请求接口的参数，具体定义方法如下:

```
// 设置 Bcuekt 的名称，必须设置
string BucketName{get;set;}
```

返回结果说明

- GetBucketEncryptionResponse:GetBucketEncryption 请求接口的返回参数，其具体定义方法如下:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义
HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }

//获取 Bucket 默认加密配置
ServerSideEncryptionConfiguration ServerSideEncryptionConfiguration { get;
set; }
```

类型 ServerSideEncryptionConfiguration，获取 Bucket 默认加密配置，定义的方法如下:

```
//获取 Bucket 的默认加密配置
List<ServerSideEncryptionRule> ServerSideEncryptionRules { get; set; }
```

类型 ServerSideEncryptionRule，获取 Bucket 默认加密规则，定义的方法如下:

```
//获取 Bucket 的默认加密规则
ServerSideEncryptionByDefault ServerSideEncryptionByDefault { get; set; }
```

类型 ServerSideEncryptionByDefault，获取 Bucket 默认加密，定义的方法如下:

```
//获取 Bucket 的默认加密算法，可以是 AES256 或 AWSKMS
ServerSideEncryptionMethod ServerSideEncryptionAlgorithm { get; set; }

//获取 Bucket 的默认加密密钥，当算法是 AES256 时，可以获取到加密密钥，字符长度
应为 32；当算法是 AWSKMS 时，获取到的是按照 cmkuuid:keyspec:userid 模式配置的字
符串，其中 cmkuuid 是 CMKID，keyspec 是指定生成的数据密钥长度，userid 是用户
id。
string ServerSideEncryptionKeyManagementServiceKeyId { get; set; }
```

示例

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
```

```
using System.Net.Security;
using System.Collections.Generic;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";


        [Obsolete]
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
            delegate (
                object sender,
                X509Certificate certificate,
                X509Chain chain,
                SslPolicyErrors sslPolicyErrors)
            {
                return true;
            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480";
            config.ForcePathStyle = true;
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config);
            Zos.GetBucketEncryptionCallBack(client, "bucket1");
        }

        public static void GetBucketEncryptionCallBack(AmazonS3Client
client, string bucketName)
        {
            GetBucketEncryptionRequest request = new
GetBucketEncryptionRequest
            {
                BucketName = bucketName,
            };
            try
            {
                ServerSideEncryptionConfiguration Configuration =
client.GetBucketEncryption(request).ServerSideEncryptionConfiguration;
                Console.WriteLine("Configuration contains {0} rules",
Configuration.ServerSideEncryptionRules.Count);
                foreach (ServerSideEncryptionRule rule in
Configuration.ServerSideEncryptionRules)
                {
                    Console.WriteLine("Rule");
                    Console.WriteLine(" ServerSideEncryptionAlgorithm =
" + rule.ServerSideEncryptionByDefault.ServerSideEncryptionAlgorithm);
                    Console.WriteLine("
ServerSideEncryptionKeyManagementServiceKeyId = " +
rule.ServerSideEncryptionByDefault.ServerSideEncryptionKeyManagementSe
rviceKeyId);
```

```
                }
                Console.WriteLine("ZOS 存储桶获取加密配置成功! ");
            }
            catch (Exception ex)
            {
                Console.WriteLine("get bucket encryption failed!" +
ex.Message);
            }
        }
    }
}
```

1.23、Delete Bucket Encrytion

功能说明

delete bucket encryption 请求删除存储桶默认加密配置

方法原型

```
DeleteBucketEncryptionResponse
DeleteBucketEncryption(DeleteBucketEncryptionRequest request)
```

参数说明

- request:类型 DeleteBucketEncryptionRequest 请求接口的参数，具体定义方法如下:

```
// 设置 Bcuekt 的名称，必须设置
string BucketName{get;set;}
```

返回结果说明

- DeleteBucketEncryptionResponse:DeleteBucketEncryption 请求接口的返回参数，其具体定义方法如下:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义
HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }
```

示例

```
using System;
```

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using System.Collections.Generic;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";


        [Obsolete]
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
            delegate (
                object sender,
                X509Certificate certificate,
                X509Chain chain,
                SslPolicyErrors sslPolicyErrors)
            {
                return true;
            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480";
            config.ForcePathStyle = true;
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config);
            Zos.DeleteBucketEncryptionCallBack(client, "bucket1");
        }

        public static void
DeleteBucketEncryptionCallBack(AmazonS3Client client, string
bucketName)
        {
            DeleteBucketEncryptionRequest request = new
DeleteBucketEncryptionRequest
            {
                BucketName = bucketName,
            };
            try
            {
                DeleteBucketEncryptionResponse response =
client.DeleteBucketEncryption(request);
                Console.WriteLine("ZOS 存储桶删除加密配置成功! ");
            }
            catch (Exception ex)
            {
                Console.WriteLine("delete bucket encryption failed!" +
ex.Message);
```

```
            }
        }
    }
}
```

1.24、Put Bucket Object Lock Configuration

功能说明

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

方法原型

```
PutObjectLockConfigurationResponse
PutObjectLockConfiguration(PutObjectLockConfigurationRequest request)
```

参数说明

- request:类型 PutObjectLockConfigurationRequest 请求接口的参数，具体定义方法如下:

```
// 设置 Bcuekt 的名称，必须设置
string BucketName{get;set;}

//设置 Bucket 的对象锁定功能
ObjectLockConfiguration ObjectLockConfiguration { get; set; }
```

类型 ObjectLockConfiguration，设置 Bucket 对象锁定，定义的方法如下:

```
//设置 Bucket 的对象锁定，ObjectLockEnabled 取值为 Enabled
ObjectLockEnabled ObjectLockEnabled { get; set; }

//设置 Bucket 对象锁定规则
ObjectLockRule Rule { get; set; }
```

类型 ObjectLockRule，设置 Bucket 对象锁定规则，定义的方法如下:

```
//设置 Bucket 对象锁定的默认保留期限
DefaultRetention DefaultRetention { get; set; }
```

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

```
//设置 Bucket 对象锁定默认保留天数，不能与设置年数同时使用
int Days { get; set; }

//设置对象锁定保留期限模式，Mode 取值为 Governance 或 Compliance
ObjectLockRetentionMode Mode { get; set; }
```

```
//设置 Bucket 对象锁定默认保留年数，不能与设置天数同时使用
int Years { get; set; }
```

返回结果说明

- PutObjectLockConfigurationResponse: PutObjectLockConfiguration 请求接口的返回参数，其具体定义方法如下:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义
HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }
```

示例

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using System.Collections.Generic;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";


        [Obsolete]
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
            delegate (
                object sender,
                X509Certificate certificate,
                X509Chain chain,
                SslPolicyErrors sslPolicyErrors)
            {
                return true;
            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480";
            config.ForcePathStyle = true;
```

```
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config);
            Zos.PutObjectLockConfigurationCallBack(client, "bucket30");
        }

        public static void
PutObjectLockConfigurationCallBack(AmazonS3Client client, string
bucketName)
        {
            ObjectLockConfiguration Configuration = new
ObjectLockConfiguration
            {
                ObjectLockEnabled = new ObjectLockEnabled("Enabled"),
                Rule = new ObjectLockRule
                {
                    DefaultRetention = new DefaultRetention
                    {
                        Days = 1,
                        Mode = new ObjectLockRetentionMode("Governance")
                    }
                }
            };
            PutObjectLockConfigurationRequest request = new
PutObjectLockConfigurationRequest
            {
                BucketName = bucketName,
                ObjectLockConfiguration = Configuration
            };
            try
            {
                PutObjectLockConfigurationResponse response =
client.PutObjectLockConfiguration(request);
                Console.WriteLine("ZOS 存储桶对象锁定功能启动成功! ");
            }
            catch (Exception ex)
            {
                Console.WriteLine("put object lock failed!" +
ex.Message);
            }
        }
    }
}
```

1.25、Get Bucket Object Lock Configuration

功能说明

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

方法原型

```
GetObjectLockConfigurationResponse
GetObjectLockConfiguration(GetObjectLockConfigurationRequest request)
```

参数说明

- request:类型 GetObjectLockConfigurationRequest 请求接口的参数，具体定义方法如下:

```
// 设置 Bcuekt 的名称，必须设置
string BucketName{get;set;}
```

返回结果说明

- GetObjectLockConfigurationResponse:GetObjectLockConfiguration 请求接口的返回参数，其具体定义方法如下:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义
HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }

//获取 Bucket 对象锁定配置
ObjectLockConfiguration ObjectLockConfiguration { get; set; }
```

类型 ObjectLockConfiguration，获取 Bucket 对象锁定，定义的方法如下:

```
//获取 Bucket 的对象锁定，ObjectLockEnabled 取值为 Enabled
ObjectLockEnabled ObjectLockEnabled { get; set; }

//获取 Bucket 对象锁定规则
ObjectLockRule Rule { get; set; }
```

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

```
//获取 Bucket 对象锁定的默认保留期限
DefaultRetention DefaultRetention { get; set; }
```

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

```
//获取 Bucket 对象锁定默认保留天数
int Days { get; set; }

//获取对象锁定保留期限模式，Mode 取值为 Governance 或 Compliance
ObjectLockRetentionMode Mode { get; set; }

//获取 Bucket 对象锁定默认保留年数
```

```
int Years { get; set; }
```

示例

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using System.Collections.Generic;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";


        [Obsolete]
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
            delegate (
                object sender,
                X509Certificate certificate,
                X509Chain chain,
                SslPolicyErrors sslPolicyErrors)
            {
                return true;
            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480";
            config.ForcePathStyle = true;
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config);
            Zos.GeObjectLockConfigurationCallBack(client, "bucket30");
        }


        public static void
GeObjectLockConfigurationCallBack(AmazonS3Client client, string
bucketName)
        {
            GetObjectLockConfigurationRequest request = new
GetObjectLockConfigurationRequest
            {
                BucketName = bucketName,
            };
            try
            {
                ObjectLockConfiguration Configuration =
client.GetObjectLockConfiguration(request).ObjectLockConfiguration;
```

```
                Console.WriteLine("ObjectLockEnabled = " +
Configuration.ObjectLockEnabled);
                Console.WriteLine("Mode = " +
Configuration.Rule.DefaultRetention.Mode);
                Console.WriteLine("Days = " +
Configuration.Rule.DefaultRetention.Days);
                Console.WriteLine("ZOS 存储桶对象锁定配置获取成功! ");
            }
            catch (Exception ex)
            {
                Console.WriteLine("get object lock failed!" +
ex.Message);
            }
        }
    }
}
```

1.26、Put Bucket Logging

功能说明

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

方法原型

```
PutBucketLoggingResponse PutBucketLogging(PutBucketLoggingRequest
request)
```

参数说明

- request:类型 PutBucketLoggingRequest 请求接口的参数，具体定义方法如下:

```
// 设置 Bcuekt 的名称，必须设置
string BucketName{get;set;}

//设置 Bucket 日志转存功能，若关闭日志转存功能，则设置空的 LoggingConfig
S3BucketLoggingConfig LoggingConfig { get; set; }
```

类型 S3BucketLoggingConfig，设置 Bucket 日志转存配置，定义的方法如下:

```
//设置 Bucket 日志转存目标存储桶
string TargetBucketName { get; set; }

//设置 Bucket 日志转存授权
```

```
List<S3Grant> Grants { get; set; }

//设置 Bucket 日志转存目标前缀
string TargetPrefix { get; set; }
```

类型 S3Grant，设置 Bucket 日志转存授权，定义的方法如下:

```
//设置 Bucket 日志转存授权信息
S3Grantee Grantee { get; set; }

//设置 Bucket 日志转存授权访问许可，Permission 值可为 READ、WRITE 或
FULL_CONTROL
S3Permission Permission { get; set; }
```

类型 S3Grantee，设置 Bucket 日志转存授权信息，定义的方法如下:

```
//设置 Bucket 日志转存授权类型，Type 可为 Email 或 CanonicalUser
GranteeType Type { get; }

//设置 Bucket 日志转存授权显示名称
string DisplayName { get; set; }

//设置 Bucket 日志转存授权邮箱地址
string EmailAddress { get; set; }

//设置 Bucket 日志转存规范用户
string CanonicalUser { get; set; }
```

返回结果说明

- PutBucketLoggingResponse: PutBucketLogging 请求接口的返回参数，其具体定义方法如下:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义
HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }
```

示例

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using System.Collections.Generic;
```

```
namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";


        [Obsolete]
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
            delegate (
                object sender,
                X509Certificate certificate,
                X509Chain chain,
                SslPolicyErrors sslPolicyErrors)
            {
                return true;
            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480";
            config.ForcePathStyle = true;
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config);
            Zos.PutBucketLoggingCallBack(client, "bucket_21",
"bucket_21");
        }

        [Obsolete]
        public static void PutBucketLoggingCallBack(AmazonS3Client
client, string sourcebucketName, string targetbucketName)
        {
            S3BucketLoggingConfig newConfiguration = new
S3BucketLoggingConfig
            {
                TargetBucketName = targetbucketName,
                TargetPrefix = "log/",
                Grants = new List<S3Grant>
                {
                    new S3Grant
                    {
                        Grantee = new S3Grantee
                        {
                            CanonicalUser = "s3",
                            DisplayName = "Second User"
                        },
                        Permission = new S3Permission("FULL_CONTROL")
                    },
                }
            };
            PutBucketLoggingRequest request = new
PutBucketLoggingRequest
            {
                BucketName = sourcebucketName,
```

```
                LoggingConfig = newConfiguration
            };
            try
            {
                PutBucketLoggingResponse response =
client.PutBucketLogging(request);
                Console.WriteLine("ZOS 存储桶服务访问日志配置成功! ");
            }
            catch (Exception ex)
            {
                Console.WriteLine("put bucket logging failed!" +
ex.Message);
            }
        }
    }
}
```

1.27、Get Bucket Logging

功能说明

get bucket logging 请求获取存储桶的日志转存配置

方法原型

```
GetBucketLoggingResponse GetBucketLogging(GetBucketLoggingRequest
request)
```

参数说明

- request:类型 GetBucketLoggingRequest 请求接口的参数，具体定义方法如下:

```
// 设置 Bcuekt 的名称，必须设置
string BucketName{get;set;}
```

返回结果说明

- GetBucketLoggingResponse:GetBucketLogging 请求接口的返回参数，其具体定义方法如下:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义
HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }
```

```
//获取 Bucket 日志转存配置
S3BucketLoggingConfig BucketLoggingConfig { get; set; }
```

类型 S3BucketLoggingConfig，获取 Bucket 日志转存配置，定义的方法如下:

```
//获取 Bucket 日志转存目标存储桶
string TargetBucketName { get; set; }

//获取 Bucket 日志转存授权
List<S3Grant> Grants { get; set; }

//获取 Bucket 日志转存目标前缀
string TargetPrefix { get; set; }
```

类型 S3Grant，获取 Bucket 日志转存授权，定义的方法如下:

```
//获取 Bucket 日志转存授权信息
S3Grantee Grantee { get; set; }

//获取 Bucket 日志转存授权访问许可，Permission 值为 READ、WRITE 或 FULL_CONTROL
S3Permission Permission { get; set; }
```

类型 S3Grantee，获取 Bucket 日志转存授权信息，定义的方法如下:

```
//获取 Bucket 日志转存授权类型，Type 为 Email 或 CanonicalUser
GranteeType Type { get; }

//获取 Bucket 日志转存授权显示名称
string DisplayName { get; set; }

//获取 Bucket 日志转存授权邮箱地址
string EmailAddress { get; set; }

//获取 Bucket 日志转存规范用户
string CanonicalUser { get; set; }
```

示例

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using System.Collections.Generic;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
```

```
        public static string accessKeySecret = "your secret key";


        [Obsolete]
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
            delegate (
                object sender,
                X509Certificate certificate,
                X509Chain chain,
                SslPolicyErrors sslPolicyErrors)
            {
                return true;
            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480";
            config.ForcePathStyle = true;
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config);
            Zos.GetBucketLoggingCallBack(client, "bucket1");
        }



        [Obsolete]
        public static void GetBucketLoggingCallBack(AmazonS3Client
client, string sourcebucketName)
        {
            GetBucketLoggingRequest request = new
GetBucketLoggingRequest
            {
                BucketName = sourcebucketName,
            };
            try
            {
                S3BucketLoggingConfig Config =
client.GetBucketLogging(request).BucketLoggingConfig;
                Console.WriteLine(" TargetBucketName = " +
Config.TargetBucketName);
                Console.WriteLine(" TargetPrefix = " +
Config.TargetPrefix);
                Console.WriteLine("TargetGrants contains {0} grants",
Config.Grants.Count);
                foreach (S3Grant grant in Config.Grants)
                {
                    Console.WriteLine("Grantee");
                    Console.WriteLine(" CanonicalUser = " +
grant.Grantee.CanonicalUser);
                    Console.WriteLine(" DisplayName = " +
grant.Grantee.DisplayName);
                    Console.WriteLine(" Type = " + grant.Grantee.Type);
                    Console.WriteLine(" Permission = " +
grant.Permission.Value);
                }
                Console.WriteLine("ZOS 存储桶服务访问日志获取成功! ");
```

```
            }
            catch (Exception ex)
            {
                Console.WriteLine("get bucket logging failed!" +
ex.Message);
            }
        }
    }
}
```

1.28、Put CORS Configuration

功能说明

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

方法原型

```
PutCORSConfigurationResponse
PutCORSConfiguration(PutCORSConfigurationRequest request)
```

参数说明

- request:类型 PutCORSConfiguration 请求接口的参数，PutCORSConfigurationRequest 类型具体定义方法如下:

```
//设置 Bcuekt 的名称，必须设置
string BucketName{get;set;}

//描述存储桶中对象的跨域访问配置，必须设置
CORSConfiguration Configuration { get; set; }
```

CORSConfiguration 类具体方法如下：

```
//跨域规则列表，最多允许设置 100 条跨域规则
List<CORSRule> Rules { get; set; }
```

CORSRule 类描述跨域规则，其具体方法如下：

```
//允许该源执行的 HTTP 方法列表，包括 GET , PUT , HEAD , POST , and DELETE。单
//挑规则可以配置多个方法。必须设置。
List<string> AllowedMethods { get; set; }

//允许能够访问该 bucket 的一个或多个源,支持 * 通配符，表示所有域名都允许访问，
//不推荐。一条 CORSRule 可以配置多个 allowedorigins。必须设置。
List<string> AllowedOrigins { get; set; }

//跨域规则的 ID,最大长度 255
```

```
string Id { get; set; }

//允许浏览器获取的 CORS 请求响应中的头部，不区分英文大小写,单条 CORSRule 可以
//配置多个 ExposeHeader。
List<string> ExposeHeaders { get; set; }

//跨域资源共享配置的有效时间，单位为秒，对应 CORS 请求响应中的 //Access-
Control-Max-Age 头部，单条 CORSRule 只能配置一个 MaxAgeSeconds
int MaxAgeSeconds { get; set; }

//允许浏览器发送 CORS 请求时携带的自定义 HTTP 请求头部，不区分英文大小写，单
条 //CORSRule 可以配置多个 AllowedHeader。
List<string> AllowedHeaders { get; set; }
```

返回结果说明

- PutCORSConfigurationResponse :PutCORSConfiguration 请求接口的返回参数，其具体定义方法如下:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义

HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }
```

示例

```
using System;
using System.Collections.Generic;
using System.Linq;
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;

namespace sdk_test
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
        static void Main(string[] args)
        {
            Console.WriteLine("hello world");

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                delegate (
```

```
                object sender,
                X509Certificate certificate,
                X509Chain chain,
                SslPolicyErrors sslPolicyErrors) // 自签名忽略认证
                {
                    return true;
                };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480"; // 指定
IP 和 Port
            config.ForcePathStyle = true; // 使用路径模式
            config.SignatureVersion = "2"; // "4" 签名版本
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接

            PutCORSConfigurationRequest req = new
PutCORSConfigurationRequest
            {
                BucketName = "rgwuser01-testbucket03",
                Configuration = new CORSConfiguration
                {
                    Rules = new List<CORSRule>
                    {
                        new CORSRule
                        {
                            AllowedMethods = new List<string>
                            {
                                "GET","PUT"
                            },
                            AllowedOrigins = new List<string>
                            {

"https://www.ctyun.com","http://10.10.10.1"
                            },
                            Id = "rule_1",
                        },
                         new CORSRule
                        {
                            AllowedMethods = new List<string>
                            {
                                "GET","PUT"
                            },
                            AllowedOrigins = new List<string>
                            {
                                "https://www.ctyun.com"
                            },
                            Id = "rule_2",
                        }
                    }
                }
            };
            client.PutCORSConfiguration(req);

            Console.Read();
        }
```

```
    }
}
```

1.29、Get CORS Configuration

功能说明

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

方法原型

```
GetCORSConfigurationResponse
GetCORSConfiguration(GetCORSConfigurationRequest request)
```

参数说明

- request:类型 GetCORSConfiguration 请求接口的参数，GetCORSConfigurationRequest 类型具体定义方法如下:

```
// 设置 Bcuekt 的名称，必须设置
string BucketName{get;set;}
```

返回结果说明

- 返回结果为 GetCORSConfigurationResponse 类型,具体方法如下：

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义
HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }

//存储桶中对象的跨域访问配置,其具体方法参见 Put CORS Configuration 部分的参数说//明
CORSConfiguration Configuration { get; set; }
```

示例

```
using System;
using System.Collections.Generic;
using System.Linq;
using System;
using Amazon;
```

```
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;


namespace sdk_test
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
        static void Main(string[] args)
        {
            Console.WriteLine("hello world");

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                delegate (
                object sender,
                X509Certificate certificate,
                X509Chain chain,
                SslPolicyErrors sslPolicyErrors) // 自签名忽略认证
                {
                    return true;
                };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480"; // 指定
IP 和 Port
            config.ForcePathStyle = true; // 使用路径模式
            config.SignatureVersion = "2"; // "4" 签名版本
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接

            GetCORSConfigurationRequest req = new
GetCORSConfigurationRequest
            {
                BucketName = "rgwuser01-testbucket03"
            };
            GetCORSConfigurationResponse res =
client.GetCORSConfiguration(req);
            Console.WriteLine(res.Configuration.Rules[0].Id);

Console.WriteLine(res.Configuration.Rules[0].AllowedOrigins[0]);

Console.WriteLine(res.Configuration.Rules[0].AllowedOrigins[1]);

Console.WriteLine(res.Configuration.Rules[0].AllowedMethods[0]);

Console.WriteLine(res.Configuration.Rules[0].AllowedMethods[1]);
            Console.WriteLine(res.Configuration.Rules[1].Id);

Console.WriteLine(res.Configuration.Rules[1].AllowedOrigins[0]);

Console.WriteLine(res.Configuration.Rules[1].AllowedMethods[0]);
```

```
Console.WriteLine(res.Configuration.Rules[1].AllowedMethods[1]);
            Console.Read();
        }
    }
}
```

1.30、Delete CORS Configuration

功能说明

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

方法原型

```
DeleteCORSConfigurationResponse
DeleteCORSConfiguration(DeleteCORSConfigurationRequest request)
```

参数说明

- request:类型 DeleteCORSConfiguration 请求接口的参数，DeleteCORSConfigurationResponse 类型具体定义方法如下:

```
// 设置 Bcuekt 的名称，必须设置
string BucketName{get;set;}
```

返回结果说明

- DeleteCORSConfigurationResponse :DeleteCORSConfiguration 请求接口的返回参数，其具体定义方法如下:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考 HttpStatusCode 定义
HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考 ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }
```

示例

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;


namespace sdk_test
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
        static void Main(string[] args)
        {
            Console.WriteLine("hello world");

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                delegate (
                object sender,
                X509Certificate certificate,
                X509Chain chain,
                SslPolicyErrors sslPolicyErrors) // 自签名忽略认证
                {
                    return true;
                };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480"; // 指定
IP 和 Port
            config.ForcePathStyle = true; // 使用路径模式
            config.SignatureVersion = "2"; // "4" 签名版本
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接

            DeleteCORSConfigurationRequest req = new
DeleteCORSConfigurationRequest
            {
                BucketName = "rgwuser01-testbucket03"
            };
            DeleteCORSConfigurationResponse res =
client.DeleteCORSConfiguration(req);
            Console.Read();
        }
    }
}
```

1.31、Put Bucket Versioning

功能说明

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

方法原型

```
PutBucketVersioningResponse
PutBucketVersioning(PutBucketVersioningRequest request);
```

参数说明

- request:是个 class,具体定义方法如下:

```
// 设置 Bcuekt 的名称，必须设置

string BucketName{get;set;}

//设置 Bucket 的版本控制功能，必须设置，S3BucketVersioningConfig 是个 class，具体定义如下

S3BucketVersioningConfig VersioningConfig { get; set; }
```

S3BucketVersioningConfig 定义方法如下:

```
//设置 Bucket 的版本控制功能，VersionStatus 是个 class，内部有两个只读的 VersionStatus 类型的变量 Suspended 和 Enabled，分别对应暂停或者开启 Bucketde 版本控制功能，具体用法查看示例

VersionStatus Status { get; set; }
```

返回结果说明

- PutBucketVersioningResponse:是个 class 类型，具体定义方法如下:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考 HttpStatusCode 定义

HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考 ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }
```

示例

```
using System;

using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;

namespace aws_console
```

```
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                            delegate (
                            object sender,
                            X509Certificate certificate,
                            X509Chain chain,
                            SslPolicyErrors sslPolicyErrors) // 自签名忽
略认证
                            {
                                return true;
                            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480"; // 指定
IP 和 Port
            config.ForcePathStyle = true; // 使用路径模式
            config.SignatureVersion = "2"; // "4" 签名版本
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接


            S3BucketVersioningConfig cfg = new
S3BucketVersioningConfig();
            cfg.Status = VersionStatus.Suspended;
            PutBucketVersioningRequest request = new
PutBucketVersioningRequest();
            request.BucketName = "bucket-s6";
            request.VersioningConfig = cfg;
            PutBucketVersioningResponse response;

            try
            {
                response = client.PutBucketVersioning(request);
                Console.WriteLine((int)response.HttpStatusCode);
            } catch (Exception ex) {
                Console.WriteLine("requeset failed!" + ex.Message);
            }
        }
    }

}
```

1.32、Get Bucket Versioning

功能说明

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

方法原型

```
GetBucketVersioningResponse
GetBucketVersioning(GetBucketVersioningRequest request);
```

参数说明

- request:是个 class 类型，具体定义方法如下:

```
// 设置 Bcuekt 的名称，必须设置

string BucketName{get;set;}
```

返回结果说明

- GetBucketVersioningResponse:是一个 class 类型，具体定义方法如下:

```
//获取 Bucket 的版本控制配置,S3BucketVersioningConfig 是个 class，具体定义如下:

S3BucketVersioningConfig VersioningConfig { get; set; }

// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum,具体参考 HttpStatusCode 定义
HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考 ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set;}
```

S3BucketVersioningConfig 的定义如下:

```
//获取 Bucket 的版本控制配置,VersionStatus 是个 class，可以通过其方法 string ToString()将配置变为字符串输出，具体可看示例

 VersionStatus Status { get; set; }
```

示例

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;

namespace aws_console
{
    class Zos
    {
```

```
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                            delegate (
                            object sender,
                            X509Certificate certificate,
                            X509Chain chain,
                            SslPolicyErrors sslPolicyErrors) // 自签名忽
略认证
                            {
                                return true;
                            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480"; // 指定
IP 和 Port
            config.ForcePathStyle = true; // 使用路径模式
            config.SignatureVersion = "2"; // "4" 签名版本
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接

            GetBucketVersioningRequest request = new
GetBucketVersioningRequest();
            request.BucketName = "bucket-s6";
            GetBucketVersioningResponse response;

            try
            {
                response = client.GetBucketVersioning(request);

Console.WriteLine(response.VersioningConfig.Status.ToString());
                Console.WriteLine((int)response.HttpStatusCode);
            } catch (Exception ex) {
                Console.WriteLine("requeset failed!" + ex.Message);
            }
        }
    }

}
```

1.33、Put Bucket Notification Configuration

功能说明

设置桶的指定事件通知。使用此 API，您可以替换现有的通知配置。配置定义了希望 ZOS 发布的事件类型，以及当 ZOS 检测到指定类型的事件时，希望 ZOS 发布事件通知的目的地。默认情况下，桶没有配置事件通知。也就是说，通知配置将是一个空的 NotificationConfiguration。在 ZOS 接收到这个请求之后，它首先验证通知的目的地是否存在，以及 bucket 所有者是否具有发送测试通知的权限。有关更多信息，请参见为 Amazon S3 事件配置通知。

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

方法原型

```
PutBucketNotificationResponse PutBucketNotification(PutBucketNotificationRequest request)
```

参数说明

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

```
//设置桶名称
string BucketName { get; set; }

//设置桶通知信息
List<TopicConfiguration> TopicConfigurations { get; set; }
```

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

```
//设置 Notification 的 Id 值，必须设置
string Id { get; set; }

//设置通知的 TopicArn，必须先创建
string Topic { get; set; }
```

Aws::S3::Model::NotificationConfiguration 是 TopicConfiguration 的抽象类，其定义方法如下：

```
//设置通知的事件，Event 是一个枚举 class 类型，必须设置
List<EventType> Events { get; set; }
```

Aws::S3::Model::EventType 中，通知支持的事件如下：

```
EventType ObjectCreatedAll
EventType ObjectCreatedPost
EventType ObjectCreatedPut
EventType ObjectCreatedCopy
EventType ObjectRemovedDeleteMarkerCreated
EventType ObjectRemovedAll
EventType ObjectRemovedDelete
EventType ObjectCreatedCompleteMultipartUpload
```

返回结果及说明

- PutBucketNotificationResponse，其具体定义方法如下：

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义

HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }
```

示例

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using System.Collections.Generic;

namespace aws_console
{
    class PutBucketNotification
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                delegate (
                object sender,
                X509Certificate certificate,
                X509Chain chain,
                SslPolicyErrors sslPolicyErrors) // 自签名忽略认证
            {
                return true;
            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480"; // 指定
IP 和 Port
            config.ForcePathStyle = true; // 使用路径模式
            config.SignatureVersion = "4"; // 签名版本，"2"或"4"
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接

            PutBucketNotificationResponse response;

            List<TopicConfiguration> TopicConfigurations = new
List<TopicConfiguration>();
            TopicConfiguration TopicConfig = new TopicConfiguration();
            TopicConfig.Id = "12345";
            TopicConfig.Topic = "arn:aws:sns:zg_cn::Kafka_topic";
```

```
            List<EventType> events = new List<EventType>();
            events.Add(EventType.ObjectCreatedAll);
            events.Add(EventType.ObjectRemovedAll);

            TopicConfig.Events = events;
            TopicConfigurations.Add(TopicConfig);

            try
            {
                PutBucketNotificationRequest request = new
PutBucketNotificationRequest();
                request.BucketName = "java-zos";
                request.TopicConfigurations = TopicConfigurations;

                response = client.PutBucketNotification(request);

                Console.WriteLine("requeset succeed!");
            } catch (Exception ex) {
                Console.WriteLine("requeset failed!" + ex.Message);
            }
        }
    }
}
```

1.34、Get Bucket Notification Configuration

功能说明

返回桶的通知配置。如果桶上没有通知，则返回一个不包含 TopicConfigurations 元素的响应。当指定 Notification 名称时，返回该 Notification 的配置信息；当不指定 Notification 名称时，返回该桶下所有 Notification 的配置信息。

默认情况下，必须是桶的所有者才能读取桶的通知配置。但是，桶的所有者可以使用桶策略授予其他用户使用 s3:GetBucketNotification 权限读取该配置的权限。

方法原型

```
GetBucketNotificationResponse GetBucketNotification(GetBucketNotificat
ionRequest request)
```

参数说明

- request:类型 Aws::S3::Model::GetBucketNotificationRequest，获取 Bucket Notification 请求接口参数，定义的方法如下:

```
//设置桶名称，必须设置
void SetBucket(const Aws::String& value)
```

返回结果及说明

- GetBucketNotificationResponse 类型，该类定义的属性有:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义

HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }

//获取本次请求返回的数据
List<TopicConfiguration> TopicConfigurations { get; set; }
```

示例

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using System.Collections.Generic;


namespace aws_console
{
    class GetBucketNotification
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";

        public static void mMain(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                delegate (
                object sender,
                X509Certificate certificate,
                X509Chain chain,
                SslPolicyErrors sslPolicyErrors) // 自签名忽略认证
            {
                return true;
            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480"; // 指定
IP 和 Port
            config.ForcePathStyle = true; // 使用路径模式
            config.SignatureVersion = "4"; // 签名版本，"2"或"4"
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接

            GetBucketNotificationResponse response;
```

```
            try
            {
                GetBucketNotificationRequest request = new
GetBucketNotificationRequest();
                request.BucketName = "java-zos";
                response = client.GetBucketNotification(request);
                List<TopicConfiguration> TopicConfigurations =
response.TopicConfigurations;

                Console.WriteLine(TopicConfigurations.ToString());

                if (TopicConfigurations.Count == 0)
                    Console.WriteLine("result is null!");

                for (int i = 0; i < TopicConfigurations.Count; i++)
                {
                    Console.WriteLine(TopicConfigurations[i].Id);
                    Console.WriteLine(TopicConfigurations[i].Topic);
                }
            } catch (Exception ex) {
                Console.WriteLine("requeset failed!" + ex.Message);
            }
        }
    }
}
```

2、Object 操作

2.1、Get Object

功能说明

Get Object 请求可以将一个文件（Object）下载至本地。该操作需要对目标 Object 具有读权限或目标 Object 对所有人都开放了读权限（公有读）。

方法原型

```
GetObjectResponse GetObject(GetObjectRequest request);
```

参数说明

- requeset:类型 GetObjectRequest ，GetObject 请求接口的参数，具体定义方法如下:

```
// 设置 Bcuekt 的名称，必须设置
```

```
string BucketName{get;set;}

//设置文件名，必须设置
string Key { get; set; }

// 只有当对象的 Etag 和 EtagToMatch 中的值一致时返回数据，否则返回
PreconditionFailed 错误
string EtagToMatch { get; set; }

//只有当对象的 Etag 和 EtagToNotMatch 中的值不一致时返回数据，否则返回
NotModified 错误
string EtagToNotMatch { get; set; }

//只有指定时间之后有修改记录的文件才会返回对象，否则返回 NotModified 错误;
DateTime ModifiedSinceDateUtc{ get; set; }

//只有指定时间之后没有修改才会返回对象，否则返回 PreconditionFailed 错误;
DateTime UnmodifiedSinceDateUtc { get; set; }

//设置文件特定版本号，获取该版本文件的元数据，没有该版本则返回 404 错误
string VersionId { get; set; }

// 下载 object 的指定字节范围， 其中 ByteRange 构造可见示例
ByteRange ByteRange { get; set; }
```

返回结果说明

- GetObjectResponse: GetObject 请求接口的返回参数，具体定义方法如下:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义

HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }

// 将获取到的对象写入指定文件，可以指定是否追加写文件
void WriteResponseStreamToFile(string filePath);
void WriteResponseStreamToFile(string filePath, bool append);

//获取当前版本文件的 Etag
string ETag { get; set; }

//获取当前文件的版本号
string VersionId { get; set; }

// 获取 Object 创建时间
 DateTime LastModified { get; set; }
```

```
// Object 拥有的 Tag 数量
int TagCount { get; set; }

 // Object 所在 Bucket 的名称
string BucketName { get; set; }

// Object Key
string Key { get; set; }

// 获取 Object 的元数据，具体见示例
MetadataCollection Metadata { get; }

// 若请求时指定了 Range，则返回结果为 “bytes”
string AcceptRanges { get; set; }

// 指明获取的 Object 是否是一个 DeleteMarker
string DeleteMarker { get; set; }

// 获取 Object LegalHold 的设置，具体见 2.12
ObjectLockLegalHoldStatus ObjectLockLegalHoldStatus { get; set; }

// 获取 Object Retention 的模式，具体见 2.13
ObjectLockMode ObjectLockMode { get; set; }

// 获取 Retention 过期日期，具体见 2.13
DateTime ObjectLockRetainUntilDate { get; set; }

// 获取对象的存储级别
S3StorageClass StorageClass { get; set; }
```

示例

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;

namespace zos_sdk_demo
{
    class GetObject
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
        static void Main(string[] args)
        {
            {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
```

```
                    delegate (
                    object sender,
                    X509Certificate certificate,
                    X509Chain chain,
                    SslPolicyErrors sslPolicyErrors) // 自签名忽略认证
                    {
                        return true;
                    };
                AmazonS3Config config = new AmazonS3Config();
                config.ServiceURL = "http://192.168.218.130:7480"; //
指定 IP 和 Port
                config.ForcePathStyle = true; // 使用路径模式
                config.SignatureVersion = "2"; // "4" 签名版本
                var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接

                var request = new GetObjectRequest();
                request.BucketName = "void3";
                request.Key = "GetObjectLock.cxx";
                //request.ByteRange = new ByteRange(100, 200);
                //request.UnmodifiedSinceDateUtc =
DateTime.UtcNow.AddDays(-1);
                //request.ModifiedSinceDateUtc =
DateTime.UtcNow.AddDays(1);
                //request.EtagToNotMatch =
"0f07c8aa0c852701e6632249fdd96890";
                //request.EtagToMatch =
"0f07c8aa0c852701e6632249fdd96891";

                GetObjectResponse response;
                try
                {
                    response = client.GetObject(request);

                }
                catch (Exception ex)
                {
                    Console.WriteLine("requeset failed!" + ex.Message);
                    return;
                }

                var meta = response.Metadata;
                foreach (var k in meta.Keys)
                {
                    Console.Out.WriteLine("key: {0}, value: {1}", k,
meta[k]);
                }
                response.WriteResponseStreamToFile("test_file");

            }
        }
    }
}
```

2.2、Get Object Metadata

功能说明

Get Object Metadata 请求可以获取对应 Object 的元数据。

方法原型

```
GetObjectMetadataResponse GetObjectMetadata(GetObjectMetadataRequest
request)
```

参数说明

- requeset:类型 GetObjectMetadataRequest，GetObjectMetadata 请求接口的参数，具体定义方法如下:

```
// 设置 Bcuekt 的名称，必须设置
string BucketName{get;set;}

//设置文件名，必须设置
string Key { get; set; }

//设置 IfMatch 参数，只有当文件的 Etag 和 IfMatch 中的值一致时返回文件元数据，否则返回 412 错误
string EtagToMatch { get; set; }

//设置 IfNoneMatch 参数，只有文件的 Etag 不满足指定字符串时才会返回元数据，否则返回 304 错误
string EtagToNotMatch { get; set; }

//设置 IfModifiedSince 参数，只有指定时间之后有修改记录的文件才会返回元数据，否则返回 304 错误;DateTime 是个 class，可以通过时间戳来构造，具体可看示例
DateTime ModifiedSinceDateUtc{ get; set; }

//设置文件特定版本号，获取该版本文件的元数据，没有该版本则返回 404 错误
string VersionId { get; set; }
```

返回结果说明

- GetObjectMetadataResponse :GetObjectMetadata 请求接口的返回参数，具体定义方法如下:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考 HttpStatusCode 定义

HttpStatusCode HttpStatusCode{get;set;}
```

```
//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }

//获取当前版本文件的 Etag
string ETag { get; set; }

//获取当前文件的版本号
string VersionId { get; set; }
```

示例

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                            delegate (
                            object sender,
                            X509Certificate certificate,
                            X509Chain chain,
                            SslPolicyErrors sslPolicyErrors) // 自签名忽
略认证
                            {
                                return true;
                            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480"; // 指定
IP 和 Port
            config.ForcePathStyle = true; // 使用路径模式
            config.SignatureVersion = "2"; // "4" 签名版本
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接

            GetObjectMetadataRequest request = new
GetObjectMetadataRequest();
            request.BucketName = "bucket-s6";
            request.Key = "boto.py";
            request.VersionId = "6uFZqH1nU7qc2bRvXAy.NgqlgwQMV3X";
```

```
            DateTime datebegin = new DateTime(1615203736);
            request.ModifiedSinceDateUtc = datebegin;
            GetObjectMetadataResponse response;

            try
            {
                response = client.GetObjectMetadata(request);
                Console.WriteLine((int)response.HttpStatusCode);
                Console.WriteLine(response.VersionId);
            } catch (Exception ex) {
                Console.WriteLine("requeset failed!" + ex.Message);
            }
        }
    }
}
```

2.3、Put Object

功能说明

Put Object 请求可以将一个文件（Object）上传至指定 Bucket。对象权限默认为私有权限。

方法原型

```
PutObjectResponse PutObject(PutObjectRequest request);
```

参数说明

- requeset:类型 PutObjectRequest ，PutObject 请求接口的参数，具体定义方法如下:

```
// 上传的 Bucket Name， 必须设置
string BucketName { get; set; }

//设置 Object 的名字，必须设置
string Key { get; set; }

// 设置 Object 的 ACL，详见 2.6
S3CannedACL CannedACL { get; set; }

// 其他设置 Object ACL 相关的属性，详见 2.6
List<S3Grant> Grants { get; set; }

// 设置 request body 的 MD5
string MD5Digest { get; set; }
```

```
//key value 形式的 Object Metadata， 使用 Add(string, string)方法添加
MetadataCollection Metadata { get; }

// 设置 Object LegalHold 详见 2.11
ObjectLockLegalHoldStatus ObjectLockLegalHoldStatus { get; set; }

// Object Retention 相关设置，详见 2.13
ObjectLockMode ObjectLockMode { get; set; }
DateTime ObjectLockRetainUntilDate { get; set; }

// 确认由请求者付费，合法值为 RequestPayer.RequestPayer
RequestPayer RequestPayer { get; set; }

// 设置 Object 的标签，详见 2.8
List<Tag> TagSet { get; set; }

//设置用于上传的流，和 FilePath ContentBody 互斥
Stream InputStream { get; set; }

// 设置上传的文件名，和 InputStream ContentBody 互斥
string FilePath { get; set; }

// 设置用于上传的 ContentBody，与 InputStream FilePath 互斥
string ContentBody { get; set; }

// 设置追加模式上传 Object
bool IsAppend { get; set; }

// 设置追加上传的起始位置，单位 Byte
long AppendPosition { get; set; }
```

返回结果说明

- PutObjectResponse： PutObject 请求的返回结果，具体定义方法如下：

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义
HttpStatusCode HttpStatusCode{get;set;}

// 获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }

// Object 的 Etag
string ETag { get; set; }

//获取本次请求返回的文件的版本号
string VersionId { get; set; }

// 获取本次请求者是否付费，合法值为 RequestCharged.Requester
```

```
RequestCharged RequestCharged { get; set; }

// 获取下次 append 的起始位置
long NextAppendPosition { get; set; }
```

示例

```
using System;

using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;

namespace zos_sdk_demo
{
    class PutObject
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
        static void Main(string[] args)
        {
            {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                    delegate (
                    object sender,
                    X509Certificate certificate,
                    X509Chain chain,
                    SslPolicyErrors sslPolicyErrors) // 自签名忽略认证
                    {
                        return true;
                    };
                AmazonS3Config config = new AmazonS3Config();
                config.ServiceURL = "http://192.168.218.130:7480"; //
指定 IP 和 Port
                config.ForcePathStyle = true; // 使用路径模式
                config.SignatureVersion = "2"; // "4" 签名版本
                var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接

                var request = new PutObjectRequest();
                request.BucketName = "void3";
                request.Key = "C-sharp-putobject";
                request.CannedACL = S3CannedACL.PublicReadWrite;
                request.FilePath = "test_file";
                request.Metadata.Add("k1", "v1");
//System.IO.FileStream("test_file", System.IO.FileMode.Open,
System.IO.FileAccess.Read);


                PutObjectResponse response;
```

```
                try
                {
                    response = client.PutObject(request);

                }
                catch (Exception ex)
                {
                    Console.WriteLine("requeset failed!" + ex.Message);
                    return;
                }

                Console.WriteLine("PutObject, Etag: {0}, VersionId:
{1}", response.ETag, response.VersionId);
            }
        }
    }
}
```

2.4、Delete Object

功能说明

Delete Object 请求可以将一个文件（Object）删除。

方法原型

```
DeleteObjectResponse DeleteObject(DeleteObjectRequest request);
```

参数说明

- DeleteObjectRequest: DeleteObject 请求的输入参数，具体定义方法如下：

```
//设置 Object 所在 Bucket 名称，必须设置
public string BucketName { get; set; }

//设置 Object 的名字，必须设置
public string Key { get; set; }

// 执行操作时是否绕过 Governance 模式锁的限制
public bool BypassGovernanceRetention { get; set; }

// 设置要删除的 Object 的 VersionId
public string VersionId { get; set; }
```

返回结果说明

- DeleteObjectResponse 定义方法如下:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义

HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }

// 表示删除的多版本 object 是否是 delete marker
public string DeleteMarker { get; set; }

// 若创建了 DeleteMarker ，则可获取到 DeleteMarker 的 VersionId
string VersionId { get; set; }
```

示例

```
using System;

using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;

namespace zos_sdk_demo
{
    class DeleteObject
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
        static void Main(string[] args)
        {
            {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                    delegate (
                    object sender,
                    X509Certificate certificate,
                    X509Chain chain,
                    SslPolicyErrors sslPolicyErrors) // 自签名忽略认证
                    {
                        return true;
                    };
                AmazonS3Config config = new AmazonS3Config();
                config.ServiceURL = "http://192.168.218.130:7480"; //
指定 IP 和 Port
                config.ForcePathStyle = true; // 使用路径模式
                config.SignatureVersion = "2"; // "4" 签名版本
```

```
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接

            var request = new DeleteObjectRequest();
            request.BucketName = "void3";
            request.Key = "C-sharp-putobject";
            request.BypassGovernanceRetention = true;

            DeleteObjectResponse response;
            try
            {
                response = client.DeleteObject(request);

            }
            catch (Exception ex)
            {
                Console.WriteLine("requeset failed!" + ex.Message);
                return;
            }

            Console.WriteLine("DeleteObject {0}", request.Key);
        }
    }
}
```

2.5、Delete Objects

功能说明

Delete Multiple Object 请求实现批量删除文件，最大支持单次删除 1000 个文件。对于返回结果，ZOS 提供 Verbose 和 Quiet 两种结果模式。Verbose 模式将返回每个 Object 的删除结果；Quiet 模式只返回报错的 Object 信息。

方法原型

```
DeleteObjectsResponse DeleteObjects(DeleteObjectsRequest request);
```

参数说明

- DeleteObjectsRequest: DeleteObjects 请求的输入参数，具体定义方法如下：

```
//设置 Object 所在 Bucket 名称，必须设置
string BucketName { get; set; }

// 设置是否开启 quiet 模式，开启 quiet 模式后，返回结果中只返回报错的 Object 信息
```

```
bool Quiet { get; set; }

//向要删除的 object 集合中添加 key
void AddKey(string key);

//向要删除的 object 集合中添加 key 并指定 version
void AddKey(string key, string version);
```

返回结果说明

- DeleteObjectsResponse: DeleteObjects 请求的返回结果，具体定义方法如下：

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义

HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }

// 若请求时未指定 Quiet 模式，则可以获取删除成功的 Object
List<DeletedObject> DeletedObjects { get; set; }

// 获取删除失败的 Object
List<DeleteError> DeleteErrors { get; set; }
```

DeletedObject: 具体定义方法如下：

```
// 获取 Object Key

string Key { get; set; }

// 获取删除的 Object 的 VersionId
string VersionId { get; set; }
```

DeleteError: 具体定义方法如下：

```
// 获取 Object Key
string Key { get; set; }

// 获取删除的 Object 的 VersionId
string VersionId { get; set; }


// 获取删除失败的错误码
string Code { get; set; }

// 获取删除失败的 Message
string Message { get; set; }
```

示例

```
using System;

using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using System.Collections.Generic;

namespace zos_sdk_demo
{
    class DeleteObjects
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
        static void Main(string[] args)
        {
            {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                    delegate (
                    object sender,
                    X509Certificate certificate,
                    X509Chain chain,
                    SslPolicyErrors sslPolicyErrors) // 自签名忽略认证
                    {
                        return true;
                    };
                AmazonS3Config config = new AmazonS3Config();
                config.ServiceURL = "http://192.168.218.130:7480"; //
指定 IP 和 Port
                config.ForcePathStyle = true; // 使用路径模式
                config.SignatureVersion = "2"; // "4" 签名版本
                var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接

                var request = new DeleteObjectsRequest();
                request.BucketName = "void3";
                request.Quiet = false;

                for (int i = 0; i < 5; i++)
                {
                    request.AddKey("object" + i);
                }
                request.BypassGovernanceRetention = true;

                DeleteObjectsResponse response;
                try
                {
                    response = client.DeleteObjects(request);
```

```
                }
                catch (Exception ex)
                {
                    Console.WriteLine("requeset failed!" + ex.Message);
                    return;
                }

                foreach (var item in response.DeletedObjects)
                {
                    Console.Out.WriteLine("DeleteObject {0}", item.Key);
                }

                foreach (var item in response.DeleteErrors)
                {
                    Console.Out.WriteLine("key: {0}, Error Code: {1},
Message: {2},  VersionId: {3}", item.Key, item.Code, item.Message,
item.VersionId);
                }
            }
        }
    }

}
```

2.6、Put Object ACL

功能说明

设置 Object 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。

有两种方式设置 ACL，不可同时使用，每次只能给一种参数赋值。其中，设置预定义的固定的 ACL，不能针对特定用户进行授权，且该参数实现的效果，也可以借由另一种方式实现，该参数使用请求头进行传递；AccessControlList 参数方式则可以针对特定用户进行授权，该方式通过请求体传递。该接口会覆盖原有 ACL 属性，包括对象所有者自身的权限，如需保留原有 ACL 属性，应将需要保留的原 ACL 添加到本次操作的授权中（设置预定义的固定 ACL 会默认将对象所有者权限设为 FULL_CONTROL）。

方法原型

```
PutACLResponse PutACL(PutACLRequest request)
```

参数说明

- request: Aws::S3::Model::PutACLRequest 类型，该类包含的属性有:

```
//桶名称属性
string BucketName { get; set; }

// 对象名称属性
string Key { get; set; }

// 对象版本 ID
string VersionId { get; set; }

// 指定预定 ACL，取值范围为
// Private
// PublicRead
// PublicReadWrite
// AuthenticatedRead
S3CannedACL CannedACL { get; set; }

// 指定 ACL 列表，与预定 ACL 不能同时使用
S3AccessControlList AccessControlList { get; set; }
```

S3AccessControlList 类型，该类包含的属性有:

```
// 对象所有者
Owner Owner { get; set; }

// 授权列表
List<S3Grant> Grants { get; set; }
```

Owner 类型，该类包含的属性有:

```
// 对象所有者 display name
string DisplayName { get; set; }

// 对象所有者用户 ID
string Id { get; set; }
```

S3Grant 类型，该类包含的属性有:

```
// 被授权用户
S3Grantee Grantee { get; set; }

// 被授权权限，取值范围
// READ
// WRITE
// READ_ACP
// WRITE_ACP
// FULL_CONTROL
S3Permission Permission { get; set; }
```

S3Grantee 类型，该类包含的属性有:

```
// 被授权用户类型，取值范围 CanonicalUser、Email、Group
GranteeType Type { get; set; }
```

```
// 被授权用户 ID，Type 为 CanonicalUser 必须指定该属性
string CanonicalUser { get; set; }

// 被授权用户 display name
string DisplayName { get; set; }

// 被授权用户邮箱，Type 为 Email 必须指定该属性
string EmailAddress { get; set; }

// 被授权组，Type 为 Group 必须指定该属性，取值范围
// http://acs.amazonaws.com/groups/global/AllUsers
// http://acs.amazonaws.com/groups/global/AuthenticatedUsers
string URI { get; set; }
```

返回结果说明

- PutBucketTaggingResponse，其具体定义方法如下：

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义

HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }
```

示例

```
using System;

using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using System.Collections.Generic;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";

        [Obsolete]
        public static void Main(string[] args)
        {
```

```
System.Net.ServicePointManager.ServerCertificateValidationCallback +=
           delegate (
               object sender,
               X509Certificate certificate,
               X509Chain chain,
               SslPolicyErrors sslPolicyErrors)
           {
               return true;
           };
           AmazonS3Config config = new AmazonS3Config();
           config.ServiceURL = "http://192.168.218.130:7480";
           config.ForcePathStyle = true;
           var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config);

           string bucket = "bucket-1";
           string key = "test-file";
           List<S3Grant> grants = new List<S3Grant>();
           grants.Add(new S3Grant {
               Grantee = new S3Grantee {
                   Type = GranteeType.Email,
                   EmailAddress = "abc@abc.com"
               },
               Permission = S3Permission.READ
           });
           grants.Add(new S3Grant
           {
               Grantee = new S3Grantee
               {
                   Type = GranteeType.CanonicalUser,
                   CanonicalUser = "test-2"
               },
               Permission = S3Permission.READ
           });
           grants.Add(new S3Grant
           {
               Grantee = new S3Grantee
               {
                   Type = GranteeType.Group,
                   URI =
"http://acs.amazonaws.com/groups/global/AuthenticatedUsers"
               },
               Permission = S3Permission.READ
           });
           PutACLRequest req = new PutACLRequest
           {
               BucketName = bucket,
               Key = key,
               // CannedACL = S3CannedACL.PublicReadWrite
               AccessControlList = new S3AccessControlList
               {
                   Owner = new Owner
                   {
                       Id = "test-1"
                   },
```

```
                    Grants = grants
                }
            };
            try
            {
                PutACLResponse res = client.PutACL(req);
                Console.WriteLine(res.HttpStatusCode);
            }
            catch (Exception ex)
            {
                Console.WriteLine("requeset failed!" + ex.Message);
            }
        }
    }
}
```

2.7、Get Object ACL

功能说明

Get Object ACL 接口用来获取对象的 ACL， 即对象的访问权限控制列表。该操作需要 READ_ACP 权限。该功能返回的结果与 Put Object ACL 参数一致，但是需要注意的是，如果以邮箱类型授权，返回结果中将会以对应被授权用户 ID 形式出现，即 Type 是 CanonicalUser。

方法原型

```
GetACLResponse GetACL(GetACLRequest request)
```

参数说明

- request: Aws::S3::Model::GetACLRequest 类型，该类包含的属性有:

```
// 桶名称属性
string BucketName { get; set; }

// 对象名称属性
string Key { get; set; }

// 对象版本 ID
string VersionId { get; set; }
```

返回结果说明

- GetACLResponse 类型，该类包含的属性有:

```
// ACL 列表
S3AccessControlList AccessControlList { get; set; }
```

S3AccessControlList 类型，该类包含的属性有:

```
// 对象所有者

Owner Owner { get; set; }

// 授权列表
List<S3Grant> Grants { get; set; }
```

Owner 类型，该类包含的属性有:

```
// 对象所有者 display name

string DisplayName { get; set; }

// 对象所有者用户 ID
string Id { get; set; }
```

S3Grant 类型，该类包含的属性有:

```
// 被授权用户

S3Grantee Grantee { get; set; }

// 被授权权限
S3Permission Permission { get; set; }
```

S3Grantee 类型，该类包含的属性有:

```
// 被授权用户类型

GranteeType Type { get; set; }

// 被授权用户 ID
string CanonicalUser { get; set; }

// 被授权用户 display name
string DisplayName { get; set; }

// 被授权用户邮箱
string EmailAddress { get; set; }

// 被授权组
string URI { get; set; }
```

示例

```
using System;

using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
```

```
using System.Net.Security;
using System.Collections.Generic;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";

        [Obsolete]
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
            delegate (
                object sender,
                X509Certificate certificate,
                X509Chain chain,
                SslPolicyErrors sslPolicyErrors)
            {
                return true;
            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480";
            config.ForcePathStyle = true;
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config);

            string bucket = "bucket-1";
            string key = "test-file";

            GetACLRequest get_req = new GetACLRequest
            {
                BucketName = bucket,
                Key = key
            };
            try
            {
                GetACLResponse get_res = client.GetACL(get_req);
                Console.WriteLine(get_res.AccessControlList.Owner.Id);

Console.WriteLine(get_res.AccessControlList.Owner.DisplayName);
                foreach(S3Grant grant in
get_res.AccessControlList.Grants)
                {
                    Console.WriteLine("--------------------------");
                    Console.WriteLine("Type = " + grant.Grantee.Type);
                    if (grant.Grantee.Type == GranteeType.Group)
                    {
                        Console.WriteLine("URI = " + grant.Grantee.URI);
                    }
                    else
                    {
                        Console.WriteLine("CanonicalUser = " +
grant.Grantee.CanonicalUser);
```

```
                    Console.WriteLine("DisplayName = " +
grant.Grantee.DisplayName);
                    Console.WriteLine("EmailAddress = " +
grant.Grantee.EmailAddress);
                }

                Console.WriteLine("Permission = " +
grant.Permission);
            }           }
          catch (Exception ex)
          {
              Console.WriteLine("requeset failed!" + ex.Message);
          }
      }
   }
}
```

2.8、Put Object Tagging

功能说明

将提供的标签集设置为存储桶中已存在的对象。标签是一个键值对。请注意，标签的最大数量限制为每个对象 10 个标签。要使用此操作，您必须具有执行 s3:PutObjectTagging 操作的权限。 默认情况下，Bucket 拥有者拥有此权限，并且可以将此权限授予其他人。要放置任何其他版本的标签，请使用 versionId 查询参数。 您还需要 s3:PutObjectVersionTagging 操作的权限。

方法原型

```
PutObjectTaggingResponse PutObjectTagging(PutObjectTaggingRequest
request)
```

参数说明

- request: Aws::S3::Model::PutBucketTaggingRequest 类型，该类定义的属性有:

```
// 桶名称
string BucketName { get; set; }


// 对象名称
string Key { get; set; }

// 对象版本 ID
string VersionId { get; set; }

// 标签
```

```
Tagging Tagging { get; set; }
```

Tagging 类型，该类定义的属性有:

```
// 标签集

List<Tag> TagSet { get; set; }
```

Tag 类型，该类定义的属性有:

```
// 标签 Key，最大 128 字节

string Key {get; set;}

// 标签 value， 最大 256 字节
string Value {get; set;}
```

返回结果说明

- PutObjectTaggingResponse，其具体定义方法如下:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义
HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }
```

示例

```
using System;

using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using System.Collections.Generic;


namespace aws_console
{
    class PutObjectTagging
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";

        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                delegate (
                object sender,
```

```
                X509Certificate certificate,
                X509Chain chain,
                SslPolicyErrors sslPolicyErrors) // 自签名忽略认证
            {
                return true;
            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480"; // 指定
IP 和 Port
            config.ForcePathStyle = true; // 使用路径模式
            config.SignatureVersion = "4";
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接

            PutObjectTaggingResponse response;

            Tag tag = new Tag();
            tag.Key = "TagObjectKey";
            tag.Value = "TagObjectValue";
            List<Tag> tags = new List<Tag>();
            tags.Add(tag);

            try
            {
                PutObjectTaggingRequest request = new
PutObjectTaggingRequest();
                request.BucketName = "java-zos";
                request.Key = "javatest";
                request.VersionId = "yGuRzcUi-2RxqngoTLzdt1jLyKR-p1w";
                request.Tagging.TagSet = tags;
                response = client.PutObjectTagging(request);

                Console.WriteLine(response);
            } catch (Exception ex) {
                Console.WriteLine("requeset failed!" + ex.Message);
            }
        }
    }
}
```

2.9、Get Object Tagging

功能说明

返回对象的标签集。要使用此操作，您必须具有执行 s3:GetObjectTagging 操作的权限。默认情况下，操作返回有关对象当前版本的信息。对于多版本的存储桶，您的存储桶中可以有一个对象的多个版本。此时，要检索任何其他版本的标签，请使用 versionId 查询参数。同时，您还需要 s3:GetObjectVersionTagging 操作的权限。

默认情况下，存储桶拥有者具有此权限，并且可以将此权限授予其他人。

方法原型

```
GetObjectTaggingResponse GetObjectTagging(GetObjectTaggingRequest
request)
```

参数说明

- request: Aws::S3::Model::GetObjectTaggingRequest 类型，该类定义的属性有:

```
// 桶名称
string BucketName { get; set; }


// 对象名称
string Key { get; set; }

// 对象版本 ID
string VersionId { get; set; }
```

返回结果说明

- GetObjectTaggingResponse 类型，该类定义的属性有:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义

HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }

// 标签集
List<Tag> Tagging { get; set; }
```

Tag 类型，该类定义的属性有:

```
// 标签 Key

string Key {get; set;}


// 标签 value
string Value {get; set;}
```

示例

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
```

```
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using System.Collections.Generic;


namespace aws_console
{
    class GetObjectTagging
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                delegate (
                object sender,
                X509Certificate certificate,
                X509Chain chain,
                SslPolicyErrors sslPolicyErrors) // 自签名忽略认证
            {
                return true;
            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480"; // 指定
IP 和 Port
            config.ForcePathStyle = true; // 使用路径模式
            config.SignatureVersion = "4"; // 签名版本, "2"或"4"
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接

            GetObjectTaggingResponse response;

            try
            {
                GetObjectTaggingRequest request = new
GetObjectTaggingRequest();
                request.BucketName = "java-zos";
                request.Key = "javatest";
                request.VersionId = "yGuRzcUi-2RxqngoTLzdt1jLyKR-p1w";
                response = client.GetObjectTagging(request);

                List<Tag> tags = response.Tagging;

                for (int i = 0; i < tags.Count; i++)
                {
                    Console.WriteLine(tags[i].Key);
                    Console.WriteLine(tags[i].Value);
                }
            } catch (Exception ex) {
                Console.WriteLine("requeset failed!" + ex.Message);
            }
        }
    }
```

}

2.10、Delete Object Tagging

功能说明

从指定的对象中删除整个标记集。要使用此操作，您必须具有执行 s3:DeleteObjectTagging 操作的权限。要删除特定对象版本的标签，请在请求中添加 versionId 查询参数。您将需要 s3:DeleteObjectVersionTagging 操作的权限。

方法原型

```
DeleteObjectTaggingResponse DeleteObjectTagging(DeleteObjectTaggingReq
uest request)
```

参数说明

- request:Aws::S3::Model::DeleteBucketTaggingRequest 类型，该类定义的属性有:

```
// 桶名称
string BucketName { get; set; }

// 对象名称
string Key { get; set; }

// 对象版本 ID
string VersionId { get; set; }
```

返回结果说明

- DeleteObjectTaggingResponse，其具体定义方法如下:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义

HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }
```

示例

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using System.Collections.Generic;


namespace aws_console
{
    class DelObjectTagging
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
        public static void Main(string[] args)
        {
                System.Net.ServicePointManager.ServerCertificateValidat
ionCallback +=
                delegate (
                object sender,
                X509Certificate certificate,
                X509Chain chain,
                SslPolicyErrors sslPolicyErrors) // 自签名忽略认证
            {
                return true;
            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480"; // 指定 I
P 和 Port
            config.ForcePathStyle = true; // 使用路径模式
            config.SignatureVersion = "4"; // 签名版本，"2"或"4"
            var client = new AmazonS3Client(accessKeyId, accessKeySecre
t, config); // 建立连接

            DeleteObjectTaggingResponse response;

            try
            {
                DeleteObjectTaggingRequest request = new DeleteObjectTag
gingRequest();
                request.BucketName = "java-zos";
                request.Key = "javatest";
                request.VersionId = "yGuRzcUi-2RxqngoTLzdt1jLyKR-p1w";
                response = client.DeleteObjectTagging(request);

                Console.WriteLine("requeset succeed!" + response.Versio
nId);
            } catch (Exception ex) {
                Console.WriteLine("requeset failed!" + ex.Message);
            }
        }
    }
}
```

2.11、Put Object Legal Hold

功能说明

put object legal hold 请求在指定对象上使用依法保留配置

方法原型

```
  PutObjectLegalHoldResponse
PutObjectLegalHold(PutObjectLegalHoldRequest request)
```

参数说明

- request:类型 PutObjectLegalHoldRequest 请求接口的参数，具体定义方法如下:

```
// 设置 Bcuekt 的名称，必须设置
string BucketName{ get; set; }

//设置对象名称，必须设置
string Key { get; set; }

//设置对象版本 Id
string VersionId { get; set; }

//设置对象依法保留配置
ObjectLockLegalHold LegalHold { get; set; }
```

类型 ObjectLockLegalHold，设置对象依法保留配置，定义的方法如下:

```
//设置对象锁定依法保留状态，Status 值可为 On 或 Off
ObjectLockLegalHoldStatus Status { get; set; }
```

返回结果说明

- PutObjectLegalHoldResponse:PutObjectLegalHold 请求接口的返回参数，其具体定义方法如下:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义
HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }
```

示例

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using System.Collections.Generic;
using System.IO;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";

        [Obsolete]
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
            delegate (
                object sender,
                X509Certificate certificate,
                X509Chain chain,
                SslPolicyErrors sslPolicyErrors)
            {
                return true;
            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480";
            config.ForcePathStyle = true;
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config);
            Zos.PutObjectLegalHoldCallBack(client, "bucket30", "time",
"kkvovO4D75RlX0soc1u1PeCXE22uuEM");
        }

        public static void PutObjectLegalHoldCallBack(AmazonS3Client
client, string bucketName, string objectName, string versionId)
        {
            ObjectLockLegalHold LH = new ObjectLockLegalHold
            {
                Status = new ObjectLockLegalHoldStatus("On")
            };
            PutObjectLegalHoldRequest request = new
PutObjectLegalHoldRequest
            {
                BucketName = bucketName,
                Key = objectName,
                VersionId = versionId,
                LegalHold = LH
            };
            try
            {
```

```
                PutObjectLegalHoldResponse response =
client.PutObjectLegalHold(request);
                Console.WriteLine("ZOS 存储桶对象依法保留设置成功! ");
            }
            catch (Exception ex)
            {
                Console.WriteLine("put object legal hold failed!" +
ex.Message);
            }
        }
    }
}
```

2.12、Get Object Legal Hold

功能说明

get object legal hold 请求获取指定对象的当前依法保留状态

方法原型

```
GetObjectLegalHoldResponse
GetObjectLegalHold(GetObjectLegalHoldRequest request)
```

参数说明

- request:类型 GetObjectLegalHoldRequest 请求接口的参数，具体定义方法如下:

```
// 设置 Bcuekt 的名称，必须设置
string BucketName{ get; set; }

//设置对象名称，必须设置
string Key { get; set; }

//设置对象版本 Id
string VersionId { get; set; }
```

返回结果说明

- GetObjectLegalHoldResponse:GetObjectLegalHold 请求接口的返回参数，其具体定义方法如下:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义
HttpStatusCode HttpStatusCode{get;set;}
```

```
//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }

//获取对象依法保留配置
ObjectLockLegalHold LegalHold { get; set; }
```

类型 ObjectLockLegalHold，获取对象依法保留配置，定义的方法如下:

```
//设置对象锁定依法保留状态，Status 值为 On 或 Off
ObjectLockLegalHoldStatus Status { get; set; }
```

示例

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using System.Collections.Generic;
using System.IO;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";

        [Obsolete]
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
            delegate (
                object sender,
                X509Certificate certificate,
                X509Chain chain,
                SslPolicyErrors sslPolicyErrors)
            {
                return true;
            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480";
            config.ForcePathStyle = true;
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config);
            Zos.GetObjectLegalHoldCallBack(client, "bucket30", "time",
"kkvovO4D75RlX0soc1u1PeCXE22uuEM");
        }

        public static void GetObjectLegalHoldCallBack(AmazonS3Client
client, string bucketName, string objectName, string versionId)
```

```
        {
            GetObjectLegalHoldRequest request = new
GetObjectLegalHoldRequest
            {
                BucketName = bucketName,
                Key = objectName,
                VersionId = versionId
            };
            try
            {
                ObjectLockLegalHold LH =
client.GetObjectLegalHold(request).LegalHold;
                Console.WriteLine("ObjectLockLegalHoldValue = " +
LH.Status.Value);
                Console.WriteLine("ZOS 存储桶对象依法保留配置获取成功! ");
            }
            catch (Exception ex)
            {
                Console.WriteLine("get object legal hold failed!" +
ex.Message);
            }
        }
    }
}
```

2.13、Put Object Retention

功能说明

put object retention 请求设置对象保留期限配置。

方法原型

```
PutObjectRetentionResponse
PutObjectRetention(PutObjectRetentionRequest request)
```

参数说明

- request:类型 PutObjectRetentionRequest 请求接口的参数，具体定义方法如下:

```
// 设置 Bcuekt 的名称，必须设置
string BucketName{ get; set; }

//设置对象名称，必须设置
string Key { get; set; }

//设置对象版本 Id
```

```
string VersionId { get; set; }

//设置绕过保留期限限制
bool BypassGovernanceRetention { get; set; }

//设置对象保留期限配置
ObjectLockRetention Retention { get; set; }
```

类型 ObjectLockRetention，设置对象保留期限配置，定义的方法如下：

```
//设置对象保留期限模式，Mode 值可为 Governance 或 Compliance
ObjectLockRetentionMode Mode { get; set; }

//设置对象保留到期日期
DateTime RetainUntilDate { get; set; }
```

返回结果说明

- PutObjectRetentionResponse:PutObjectRetention 请求接口的返回参数，其具体定义方法如下：

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义
HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }
```

示例

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using System.Collections.Generic;
using System.IO;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";

        [Obsolete]
        public static void Main(string[] args)
        {
```

```
System.Net.ServicePointManager.ServerCertificateValidationCallback +=
           delegate (
               object sender,
               X509Certificate certificate,
               X509Chain chain,
               SslPolicyErrors sslPolicyErrors)
           {
               return true;
           };
           AmazonS3Config config = new AmazonS3Config();
           config.ServiceURL = "http://192.168.218.130:7480";
           config.ForcePathStyle = true;
           var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config);
           Zos.PutObjectRetentionCallBack(client, "bucket30", "time",
"kkvovO4D75RlX0soc1u1PeCXE22uuEM");
       }

       public static void PutObjectRetentionCallBack(AmazonS3Client
client, string bucketName, string objectName, string versionId)
       {
           ObjectLockRetention retention = new ObjectLockRetention
           {
               Mode = new ObjectLockRetentionMode("GOVERNANCE"),
               RetainUntilDate = new DateTime(2021, 04, 22, 19, 35,
00)
           };
           PutObjectRetentionRequest request = new
PutObjectRetentionRequest
           {
               BucketName = bucketName,
               Key = objectName,
               VersionId = versionId,
               BypassGovernanceRetention = true,
               Retention = retention
           };
           try
           {
               PutObjectRetentionResponse response =
client.PutObjectRetention(request);
               Console.WriteLine("ZOS 存储桶对象保留期限设置成功! ");
           }
           catch (Exception ex)
           {
               Console.WriteLine("put object retention failed!" +
ex.Message);
           }
       }
   }
}
```

2.14、Get Object Retention

功能说明

get object retention 获取对象的保留期限设置

方法原型

```
GetObjectRetentionResponse
GetObjectRetention(GetObjectRetentionRequest request)
```

参数说明

- request:类型 GetObjectRetentionRequest 请求接口的参数，具体定义方法如下:

```
// 设置 Bcuekt 的名称，必须设置
string BucketName{ get; set; }

//设置对象名称，必须设置
string Key { get; set; }

//设置对象版本 Id
string VersionId { get; set; }
```

返回结果说明

- GetObjectRetentionResponse:GetObjectRetention 请求接口的返回参数，其具体定义方法如下:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义
HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }

//获取对象保留期限配置
ObjectLockRetention Retention { get; set; }
```

类型 ObjectLockRetention，获取对象保留期限配置，定义的方法如下:

```
//获取对象保留期限模式，Mode 值为 Governance 或 Compliance
ObjectLockRetentionMode Mode { get; set; }

//获取对象保留到期日期
DateTime RetainUntilDate { get; set; }
```

示例

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using System.Collections.Generic;
using System.IO;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";

        [Obsolete]
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
            delegate (
                object sender,
                X509Certificate certificate,
                X509Chain chain,
                SslPolicyErrors sslPolicyErrors)
            {
                return true;
            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480";
            config.ForcePathStyle = true;
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config);
            Zos.GetObjectRetentionCallBack(client, "bucket30", "time",
"kkvovO4D75RlX0soc1u1PeCXE22uuEM");
        }

        public static void GetObjectRetentionCallBack(AmazonS3Client
client, string bucketName, string objectName, string versionId)
        {
            GetObjectRetentionRequest request = new
GetObjectRetentionRequest
            {
                BucketName = bucketName,
                Key = objectName,
                VersionId = versionId
            };
            try
            {
                ObjectLockRetention retention =
client.GetObjectRetention(request).Retention;
```

```
                Console.WriteLine("Mode = " + retention.Mode);
                Console.WriteLine("Date = " +
retention.RetainUntilDate.Date);
                Console.WriteLine("Hour = " +
retention.RetainUntilDate.Hour);
                Console.WriteLine("Minute = " +
retention.RetainUntilDate.Minute);
                Console.WriteLine("Second = " +
retention.RetainUntilDate.Second);
                Console.WriteLine("ZOS 存储桶对象保留期限配置获取成功! ");
            }
            catch (Exception ex)
            {
                Console.WriteLine("get object retention failed!" +
ex.Message);
            }
        }
    }
}
```

2.15、Genertae Presigned Url

功能说明

Generate Presigned Url 请求生成一个临时的预签名的 Url，没有权限访问集群的用户可以通过该 Url 访问集群，包括上传、下载文件等等。预签名上传的对象，默认为私有权限。

方法原型

```
string GetPreSignedURL(GetPreSignedUrlRequest request)
```

参数说明

- request:类型是个 class，具体定义方法如下:

```
//设置 Bucket 的名称,必须设置
string BucketName { get; set; }

//设置 Key 的名称，必须设置
string Key { get; set; }

//设置 http 方法，必须设置，HttpVerb 是个 enum 类型，0 到 3 分别对应 GET、HEAD、PUT 和 DELETE，具体用法看示例
HttpVerb Verb { get; set; }
```

```
//设置 URL 的超时时间，超过该时间的 URL 不再有效,DateTime 是 C#内置类型，具体用法可看示例
DateTime Expires { get; set; }
```

返回结果说明

- string:详细的 URL 字符串

示例

```
using System;
using Amazon.S3;
using Amazon.S3.Model;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
        public static void Main(string[] args)
        {
            System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                            delegate (
                            object sender,
                            X509Certificate certificate,
                            X509Chain chain,
                            SslPolicyErrors sslPolicyErrors) // 自签名忽略认证
                            {
                                return true;
                            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480"; // 指定 IP 和 Port
            config.ForcePathStyle = true; // 使用路径模式
            config.SignatureVersion = "2"; // "4" 签名版本
            var client = new AmazonS3Client(accessKeyId, accessKeySecret, config); // 建立连接

            GetPreSignedUrlRequest requeset
                = new GetPreSignedUrlRequest();

            requeset.BucketName = "bucket-s6";
            requeset.Key = "boto.py";
            requeset.Verb = HttpVerb.GET;
            requeset.Expires = DateTime.Now.AddHours(2);

            String response;
            try
```

```
            {
                response = client.GetPreSignedURL(requeset);
                Console.WriteLine(response);
            } catch (Exception ex) {
                Console.WriteLine("requeset failed!" + ex.Message);
            }
        }
    }
}
```

2.16、CopyObject

功能说明

Put Object Copy 请求实现将一个文件从源路径复制到目标路径。

方法原型

```
CopyObjectResponse CopyObject(CopyObjectRequest request);
```

参数说明

- CopyObjectRequest ：CopyObject 请求的输入参数，具体定义方法如下：

```
//设置 Object 所在 Bucket 名称，必须设置
string DestinationBucket { get; set; }

//设置 Object 的名字，必须设置
void SetKey(const Aws::String& value)

// 设置 copy 的源 Bucket， 必须设置
 string SourceBucket { get; set; }

// 设置 copy 的源 Key， 必须设置
string DestinationKey { get; set; }

// 仅当指定的 Etag 和 Copy Source 指定的 object 的 Etag 匹配时才复制
string ETagToMatch { get; set; }

// 仅当指定的 Etag 和 Copy Source 指定的 object 的 Etag 不匹配时才复制
string ETagToNotMatch { get; set; }

// 仅当 CopySource 在指定时间后更新过才复制
DateTime ModifiedSinceDateUtc { get; set; }

// 仅当 CopySource 在指定时间后未更新过才复制
```

```
DateTime UnmodifiedSinceDateUtc { get; set; }

// ACL 相关设置，详见 2.6
List<S3Grant> Grants { get; set; }
S3CannedACL CannedACL { get; set; }

// 复制目的对象的 Metadata，使用 Add(string, string)方法添加
MetadataCollection Metadata { get; }

// Metadata 复制选项，'COPY' 复制源的 Metadata 或 'REPLACE'使用新指定的 Metadata 覆盖
S3MetadataDirective MetadataDirective { get; set; }

// 设置复制对象的 Tag，TagSet 用法参考 2.8
 List<Tag> TagSet { get; set; }

// 设置复制的 Object 的 LegalHold 开关， 详见 2.11
 ObjectLockLegalHoldStatus ObjectLockLegalHoldStatus { get; set; }

// 设置复制的 Object 的 Retention 模式 详见 2.13
ObjectLockMode ObjectLockMode { get; set; }

// 设置复制的 Object 的 Retention 保留过期时间 详见 2.13
 DateTime ObjectLockRetainUntilDate { get; set; }
```

返回结果说明

- CopyObjectResponse：CopyObject 请求的返回结果，具体定义方法如下：

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考 HttpStatusCode 定义
HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考 ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }

// 复制的 Object 的 ETag
 string ETag { get; set; }

// 复制的 Object 的创建时间
string LastModified { get; set; }
```

示例

```
using System;

using Amazon;
```

```
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;

namespace zos_sdk_demo
{
    class CopyObject
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
        static void Main(string[] args)
        {
            {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                    delegate (
                    object sender,
                    X509Certificate certificate,
                    X509Chain chain,
                    SslPolicyErrors sslPolicyErrors) // 自签名忽略认证
                    {
                        return true;
                    };
                AmazonS3Config config = new AmazonS3Config();
                config.ServiceURL = "http://192.168.218.130:7480"; //
指定 IP 和 Port
                config.ForcePathStyle = true; // 使用路径模式
                config.SignatureVersion = "2"; // "4" 签名版本
                var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接

                var request = new CopyObjectRequest();
                request.SourceBucket = "void3";
                request.SourceKey = "txt";
                request.DestinationBucket = "void1";
                request.DestinationKey = "txt_copy";

                CopyObjectResponse response;
                try
                {
                    response = client.CopyObject(request);

                }
                catch (Exception ex)
                {
                    Console.WriteLine("requeset failed!" + ex.Message);
                    return;
                }

                Console.WriteLine("CopyObject, new object Created on:
{0},Etag: {1}", response.LastModified, response.ETag);
            }
        }
```

```
    }
}
```

2.17、CopyPart

功能说明

Copy 请求实现将一个文件从源路径复制到目标路径，支持拷贝大于 5GB 的文件。

方法原型

```
CopyPartResponse CopyPart(CopyPartRequest request)
```

参数说明

- request: 类型 CopyPartRequest,具体定义属性方法如下：

```
// 设置 Copy 目的 Bucket 名称
string DestinationBucket { get; set; }

// 设置 Copy 目的 Object 名称
string DestinationKey { get; set; }

// 设置 Copy 源 Bucket 名称
string SourceBucket { get; set; }

// 设置 Copy 源 Object 名称
string SourceKey { get; set; }

// 设置当前 Part 的起始字节
long FirstByte { get; set; }

// 设置当前 Part 的结束字节
long LastByte { get; set; }

// 设置当前 Part 的编号
int PartNumber { get; set; }

// 设置 MultiUpload ID，由 3.1 initMultiPartUpload 返回
string UploadId { get; set; }

// 设置 Copy 源的 VersionId
string SourceVersionId { get; set; }

// 当对象的 Etag 和 List 中的值一致时才复制对象
```

```
List<string> ETagToMatch { get; set; }

// 只有指定时间之后有修改记录的对象才复制对象
DateTime ModifiedSinceDate { get; set; }

// 当对象的 Etag 和 eTagList 中的值不一致时才复制对象
List<string> ETagsToNotMatch { get; set; }

// 只有指定时间之后没有修改记录的对象才复制对象
DateTime UnmodifiedSinceDate { get; set; }
```

返回结果说明

- CopyPartResponse 是一个 class，具体定义属性方法如下

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义

HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }

// 获取复制对象的 etag
string ETag { get; set; }

// 获取本次复制的 PartNumber
int PartNumber { get; set; }

// 获取复制对象的 VersionId
string CopySourceVersionId { get; set; }

// 获取复制对象的修改时间
DateTime LastModified { get; set; }
```

示例

```
using System;

using Amazon.S3;
using Amazon.S3.Model;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using System.Collections.Generic;

namespace zos_sdk_demo
{
    class CopyPartObject
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
```

```
       static void Main(string[] args)
       {
           {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                    delegate (
                    object sender,
                    X509Certificate certificate,
                    X509Chain chain,
                    SslPolicyErrors sslPolicyErrors) // 自签名忽略认证
                    {
                        return true;
                    };
               AmazonS3Config config = new AmazonS3Config();
               config.ServiceURL = "http://192.168.218.130:7480"; //
指定 IP 和 Port
               config.ForcePathStyle = true; // 使用路径模式
               config.SignatureVersion = "2"; // "4" 签名版本
               var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接

               var request = new CopyPartRequest();
               string SourceBucket = "java-sdk-bucket";
               string SourceKey = "zos-sdk-cpp.tar.gz";
               string DestinationBucket = "void1";
               string DestinationKey = "C-Sharp-copy-part";

               InitiateMultipartUploadRequest initRequest = new
InitiateMultipartUploadRequest();
               initRequest.BucketName = DestinationBucket;
               initRequest.Key = DestinationKey;

               GetObjectMetadataRequest metaRequest = new
GetObjectMetadataRequest();
               metaRequest.BucketName = SourceBucket;
               metaRequest.Key = SourceKey;

               try
               {
                   GetObjectMetadataResponse metaResult =
client.GetObjectMetadata(metaRequest);
                   InitiateMultipartUploadResponse initResult =
client.InitiateMultipartUpload(initRequest);


                   long size = metaResult.ContentLength;


                   request.SourceBucket = SourceBucket;
                   request.SourceKey = SourceKey;
                   request.DestinationBucket = DestinationBucket;
                   request.DestinationKey = DestinationKey;
                   request.UploadId = initResult.UploadId;

                   // 以 5M 为一段复制 Object
```

```
                long partSize = 5 * 1024 * 1024;
                long bytePosition = 0;
                int partNum = 1;

                var listCopyResponse = new List<CopyPartResponse>();
                while (bytePosition < size)
                {
                    long lastByte = Math.Min(bytePosition + partSize
- 1, size - 1);

                    // 复制一段
                    CopyPartRequest copyRequest = new
CopyPartRequest();

                    copyRequest.SourceBucket = SourceBucket;
                    copyRequest.SourceKey = SourceKey;
                    copyRequest.DestinationBucket =
initRequest.BucketName;
                    copyRequest.DestinationKey = initRequest.Key;
                    copyRequest.UploadId = initResult.UploadId;
                    copyRequest.FirstByte = bytePosition;
                    copyRequest.LastByte = lastByte;
                    copyRequest.PartNumber = partNum++;


listCopyResponse.Add(client.CopyPart(copyRequest));
                    bytePosition += partSize;
                }
                // 完成 multiPartUpload
                CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest();
                completeRequest.BucketName = DestinationBucket;
                completeRequest.Key = DestinationKey;
                completeRequest.UploadId = initResult.UploadId;
                completeRequest.PartETags =
GetETags(listCopyResponse);

                client.CompleteMultipartUpload(completeRequest);
                Console.Out.WriteLine("Multipart copy complete.");
            }
            catch (Exception ex)
            {
                Console.WriteLine("requeset failed!" + ex.Message);
                return;
            }
        }
    }

    private static List<PartETag> GetETags(List<CopyPartResponse>
responses)
    {
        List<PartETag> etags = new List<PartETag>();
        foreach (CopyPartResponse response in responses)
        {
            etags.Add(new PartETag()
            {
```

```
                    PartNumber = response.PartNumber,
                    ETag = response.ETag
                });
            }
            return etags;
        }
    }
}
```

2.18、RestoreObject

功能说明

Restore Object 提交一个解冻请求，请求解冻一个归档类型文件，生成一份标准类型的临时副本数据。

方法原型

```
RestoreObjectResponse RestoreObject(RestoreObjectRequest request);
```

参数说明

- RestoreObjectRequest ： RestoreObject 请求的输入参数，具体定义方法如下：

```
//设置 Bucket 名称，必须设置
string BucketName { get; set; }

//设置 Object 名字，必须设置
string Key { get; set; }

// 设置解冻出的副本的有效期，单位为天（支持 1~31 天）， 必须设置
int Days { get; set; }

// 指定对象的版本号
string VersionId{ get; set; }
```

返回结果说明

- RestoreObjectResponse: RestoreObject 请求的返回结果，具体定义方法如下：

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义
HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }
```

示例

```
using System;

using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;

namespace zos_sdk_demo
{
    class RestoreObject
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
        static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                    delegate (
                    object sender,
                    X509Certificate certificate,
                    X509Chain chain,
                    SslPolicyErrors sslPolicyErrors) // 自签名忽略认证
                    {
                        return true;
                    };
                AmazonS3Config config = new AmazonS3Config();
                config.ServiceURL = "http://192.168.218.130:7480"; //
指定 IP 和 Port
                config.ForcePathStyle = true; // 使用路径模式
                config.SignatureVersion = "2"; // "4" 签名版本
                var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接

                var request = new RestoreObjectRequest();
                request.BucketName = "your_bucket_name";
                request.Key = "your_object_name";
                request.Days = 1;// 1-31 当前解冻时间限制是 1-31 天

                RestoreObjectResponse response;
                try
```

```
                {
                    response = client.RestoreObject(request);
                    Console.WriteLine((int)response.HttpStatusCode);

                }
                catch (Exception ex)
                {
                    Console.WriteLine("requeset failed!" + ex.Message);
                }
            }
        }
    }
```

3、分块上传操作

3.1、Init Multipart Upload

功能说明

Init Multipart Upload 请求实现初始化分片上传，成功执行此请求以后会返回 Upload ID 用于后续的 Upload Part 请求。如您不主动设置对象的权限，则对象权限默认为私有。

方法原型

```
InitiateMultipartUploadResponse
InitiateMultipartUpload(InitiateMultipartUploadRequest request);
```

参数说明

- request:是个 class 类型，具体定义方法如下:

```
// 设置 Bcuekt 的名称，必须设置
string BucketName{get;set;}

//设置文件在集群中保存的 Key 名称，一般和文件名一致，必须设置
string Key { get; set; }

//设置 Bucket 的 ACL 规则，S3CannedACL 是个 class，内部包含几个 static readonly
类型为 S3CannedACL 的成员变量 NoACL、Private、PublicReadWrite、
PublicReadWrite、AuthenticatedRead 具体用法参考示例
S3CannedACL CannedACL { get; set;}
```

```
//设置文件标签，文件标签是个 Key-Value 的键值对，Tag 是个 class，具体定义如下
List<Tag> TagSet { get; set; }
```

Tag 的定义如下：

```
// 设置标签的 Key
string string Key { get; set; }

//设置标签 Key 的 Value
string Key { get; set; }
```

返回结果说明

- InitiateMultipartUploadResponse ：类型是个 class，具体方法定义如下：

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum,具体参考 HttpStatusCode 定义
HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考 ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set;}

//获取分段上传的 UploadId
string UploadId { get; set; }

//获取 Bucket 的名称
string BucketName { get; set; }

//获取 Key 的名称
string Key { get; set; }
```

示例

```
using System;
using Amazon.S3;
using Amazon.S3.Model;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
```

```
                            delegate (
                            object sender,
                            X509Certificate certificate,
                            X509Chain chain,
                            SslPolicyErrors sslPolicyErrors) // 自签名忽
略认证
                            {
                                return true;
                            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480"; // 指定
IP 和 Port
            config.ForcePathStyle = true; // 使用路径模式
            config.SignatureVersion = "2"; // "4" 签名版本
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接

            InitiateMultipartUploadRequest requeset
                = new InitiateMultipartUploadRequest();
            requeset.BucketName = "bucket-s6";
            requeset.Key = "50M";
            requeset.CannedACL = S3CannedACL.NoACL;
            InitiateMultipartUploadResponse response;

            try
            {
                response = client.InitiateMultipartUpload(requeset);
                Console.WriteLine(response.UploadId);
                Console.WriteLine(response.BucketName);
                Console.WriteLine(response.Key);
                Console.WriteLine((int)response.HttpStatusCode);
            } catch (Exception ex) {
                Console.WriteLine("requeset failed!" + ex.Message);
            }
        }
    }
}
```

3.2、Upload Part

功能说明

Upload Part 请求实现在初始化以后的分块上传，支持的块的数量为 1 到 10000，除了最后一块，其他每块大小都必须大于或等于 5M。在每次请求 Upload Part 时，需要携带 partNumber 和 uploadID，partNumber 为块的编号，支持乱序上传。partNumber 的取值需要从 1 开始。

方法原型

```
```

```
UploadPartResponse UploadPart(UploadPartRequest request);
```

参数说明

- requeset:类型是个 class，具体定义方法如下:

```
//设置在 3.1 中返回的 UploadId，必须设置
string UploadId { get; set; }

// 设置 Bcuekt 的名称，必须设置
string BucketName{get;set;}

//设置文件在集群中保存的 Key 名称，一般和文件名一致，必须设置
string Key { get; set; }

//设置分段号，必须设置
int PartNumber { get; set; }

//设置文件的输入流，必须设置
Stream InputStream { get; set; }

//设置分段大小
long PartSize { get; set; }
```

返回结果说明

- UploadPartResponse :类型是个 class，具体定义方法如下:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义

HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }

//获取该分段的 Etag
string ETag { get; set; }

//获取分段号
 int PartNumber { get; set; }
```

示例

```
using System;
using System.IO;
using Amazon.S3;
using Amazon.S3.Model;
```

```
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                            delegate (
                            object sender,
                            X509Certificate certificate,
                            X509Chain chain,
                            SslPolicyErrors sslPolicyErrors) // 自签名忽
略认证
                            {
                                return true;
                            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480"; // 指定
IP 和 Port
            config.ForcePathStyle = true; // 使用路径模式
            config.SignatureVersion = "2"; // "4" 签名版本
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接

            UploadPartRequest requeset
                = new UploadPartRequest();
            requeset.BucketName = "bucket-s6";
            requeset.Key = "50M";
            requeset.UploadId = "2~tjdeCEd0rSap5bpwuVVq1Ylx9haUa-G";
            requeset.PartNumber = 2;

            FileStream fileStream =
                new FileStream("error.txt", FileMode.Open,
FileAccess.Read, FileShare.Read);
            byte[] bytes = new byte[fileStream.Length];
            Console.WriteLine(fileStream.Length);
            fileStream.Read(bytes, 0, bytes.Length);
            // 把 byte[] 转换成 Stream
            Stream stream = new MemoryStream(bytes);
            requeset.PartSize = fileStream.Length;
            fileStream.Close();
            requeset.InputStream = stream;
            UploadPartResponse response;

            try
            {
                response = client.UploadPart(requeset);
                Console.WriteLine(response.ETag);
                Console.WriteLine(response.PartNumber);
```

```
                Console.WriteLine((int)response.HttpStatusCode);
            } catch (Exception ex) {
                Console.WriteLine("requeset failed!" + ex.Message);
            }
        }
    }

}
```

3.3、Complete Multipart Upload

功能说明

Complete Multipart Upload 用来实现完成整个分块上传。当您已经使用 Upload Parts 上传所有块以后，你可以用该 API 完成上传。在使用该 API 时，您必须在 Body 中给出每一个块的 PartNumber 和 ETag，用来校验块的准确性。

方法原型

```
CompleteMultipartUploadResponse
CompleteMultipartUpload(CompleteMultipartUploadRequest request);
```

参数说明

- requeset:类型是个 class，具体定义方法如下:

```
//设置在 3.1 中返回的 UploadId，必须设置
string UploadId { get; set; }

// 设置 Bcuekt 的名称，必须设置
string BucketName{get;set;}

//设置文件在集群中保存的 Key 名称，一般和文件名一致，必须设置
string Key { get; set; }

//设置分段信息，必须按照分段号顺序设置,PartETag 是个 class，具体定义如下:
List<PartETag> PartETags { get; set; }
```

PartETag 具体定义如下:

```
//设置分段号
int PartNumber { get; set; }

//设置分段 Etag
```

```
string BucketName{get;set;}
```

返回结果说明

- CompleteMultipartUploadResponse:类型是个 class，具体定义方法如下:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义

HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }

//获取 Bucket 的名称
string BucketName { get; set; }

//获取 Key 的名称
string Key { get; set; }

//获取文件整体的 Etag
string Key { get; set; }

//获取文件具体位置
string Location { get; set; }
```

示例

```
using System;
using Amazon.S3;
using System.Collections.Generic;
using Amazon.S3.Model;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                            delegate (
                            object sender,
                            X509Certificate certificate,
                            X509Chain chain,
                            SslPolicyErrors sslPolicyErrors) // 自签名忽
略认证
```

```
                        {
                            return true;
                        };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480"; // 指定
IP 和 Port
            config.ForcePathStyle = true; // 使用路径模式
            config.SignatureVersion = "2"; // "4" 签名版本
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接

            CompleteMultipartUploadRequest requeset
                = new CompleteMultipartUploadRequest();

            requeset.BucketName = "bucket-s6";
            requeset.UploadId = "2~tjdeCEd0rSap5bpwuVVq1Ylx9haUa-G";
            requeset.Key = "50M";


            List<PartETag> list = new List<PartETag>();
            PartETag part1 = new PartETag();
            part1.PartNumber = 1;
            part1.ETag = "0be97adbf15a72a54aeffbbebf34344d";
            list.Add(part1);
            PartETag part2 = new PartETag();
            part2.PartNumber = 2;
            part2.ETag = "d15c1563f1790d1d9e9ae689193a7633";
            list.Add(part2);
            PartETag part3 = new PartETag();
            part3.PartNumber = 3;
            part3.ETag = "375a4e535cdb488d9b43a66e28b43e5a";
            list.Add(part3);
            PartETag part4 = new PartETag();
            part4.PartNumber = 4;
            part4.ETag = "765bd038f35f2d745c773758c4a5d4c5";
            list.Add(part4);
            requeset.PartETags = list;
            CompleteMultipartUploadResponse response;

            try
            {
                response = client.CompleteMultipartUpload(requeset);
                Console.WriteLine(response.BucketName);
                Console.WriteLine(response.Key);
                Console.WriteLine(response.ETag);
                Console.WriteLine(response.Location);
            } catch (Exception ex) {
                Console.WriteLine("requeset failed!" + ex.Message);
            }
        }
    }

}
```

3.4、List Parts

功能说明

List Parts 用来查询特定分段上传中的已上传的分段的信息。

方法原型

```
ListPartsResponse ListParts(ListPartsRequest request)
```

参数说明

- request:类型是个 class，具体定义方法如下:

```
//设置在 3.1 中返回的 UploadId，必须设置
string UploadId { get; set; }

// 设置 Bcuekt 的名称，必须设置
string BucketName{get;set;}

//设置 Key 名称，必须设置
string Key { get; set; }

//设置最多返回的分段数目
int MaxParts { get; set; }

//设置返回的分段起始号，只有分段号大于该值得分段信息才会返回
string PartNumberMarker { get; set; }
```

返回结果说明

- ListPartsResponse :是个 class 类型，具体定义方法如下:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义

HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }

//获取 UploadId
string UploadId { get; set; }

//获取本次请求可以返回的最大分段信息数目
```

```
int MaxParts { get; set; }

//获取本次请求的 PartNumberMarker
int PartNumberMarker { get; set; }

//获取下次请求的 PartNumberMarker，主要用于多次顺序获取分段信息
int MaxParts { get; set; }

//设置返回的分段起始号，只有分段号大于该值得分段信息才会返回
int NextPartNumberMarker { get; set; }

//本次请求分段信息是否是截断返回
bool IsTruncated { get; set; }

//获取分段信息，PartDetail 是个 class，具体方法定义如下
List<PartDetail> Parts { get; set; }
```

PartDetail 具体定义方法如下:

```
//获取分段号
int PartNumber { get; set; }

//获取分段 Etag
string ETag { get; set; }

//获取分段大小
long Size { get; set; }

//获取分段上次修改时间
DateTime LastModified { get; set; }
```

示例

```
using System;
using Amazon.S3;
using System.Collections.Generic;
using Amazon.S3.Model;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                            delegate (
                            object sender,
```

```
                        X509Certificate certificate,
                        X509Chain chain,
                        SslPolicyErrors sslPolicyErrors) // 自签名忽
略认证
                        {
                            return true;
                        };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480"; // 指定
IP 和 Port
            config.ForcePathStyle = true; // 使用路径模式
            config.SignatureVersion = "2"; // "4" 签名版本
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接

            ListPartsRequest requeset
                = new ListPartsRequest();
            requeset.BucketName = "bucket-s6";
            requeset.Key = "50M";
            requeset.UploadId = "2~tjdeCEd0rSap5bpwuVVq1Ylx9haUa-G";
            requeset.MaxParts = 3;
            requeset.PartNumberMarker = "0";
            ListPartsResponse response;

            try
            {
                response = client.ListParts(requeset);
                Console.WriteLine(response.UploadId);
                Console.WriteLine(response.MaxParts);
                Console.WriteLine(response.NextPartNumberMarker);
                Console.WriteLine(response.PartNumberMarker);
                Console.WriteLine(response.IsTruncated);
                List<PartDetail> parts = response.Parts;
                for(int i = 0; i < parts.Count; i++)
                {
                    Console.WriteLine(parts[i].PartNumber);
                    Console.WriteLine(parts[i].ETag);
                    Console.WriteLine(parts[i].Size);
                    Console.WriteLine(parts[i].LastModified.ToString());
                }

            } catch (Exception ex) {
                Console.WriteLine("requeset failed!" + ex.Message);
            }
        }
    }
}
```

3.5、Abort Multipart Upload

功能说明

Abort Multipart Upload 用来实现舍弃一个分块上传并删除已上传的块。当您调用 Abort Multipart Upload 时，如果有正在使用这个 Upload Parts 上传块的请求，则 Upload Parts 会返回失败。

方法原型

```
AbortMultipartUploadResponse
AbortMultipartUpload(AbortMultipartUploadRequest request);
```

参数说明

- requeset:类型是个 class，具体定义方法如下:

```
// 设置 Bcuekt 的名称，必须设置
string BucketName{get;set;}

//设置 Key 的名称，必须设置
string Key { get; set; }

//设置 UploadId，必须设置
string UploadId { get; set; }
```

返回结果说明

- AbortMultipartUploadResponse:类型是个 class，具体定义方法如下:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义

HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }
```

示例

```
using System;
using Amazon.S3;
using Amazon.S3.Model;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
```

```
        public static string accessKeySecret = "your secret key";
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                            delegate (
                            object sender,
                            X509Certificate certificate,
                            X509Chain chain,
                            SslPolicyErrors sslPolicyErrors) // 自签名忽
略认证
                            {
                                return true;
                            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480"; // 指定
IP 和 Port
            config.ForcePathStyle = true; // 使用路径模式
            config.SignatureVersion = "2"; // "4" 签名版本
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接

            AbortMultipartUploadRequest requeset
                = new AbortMultipartUploadRequest();

            requeset.BucketName = "bucket-s6";
            requeset.UploadId = "2~-23KagCv9Tv-_34XIZJpInonBiJPVd2";
            requeset.Key = "30M";

            AbortMultipartUploadResponse response;

            try
            {
                response = client.AbortMultipartUpload(requeset);
                Console.WriteLine("requeset success!");
            } catch (Exception ex) {
                Console.WriteLine("requeset failed!" + ex.Message);
            }
        }
    }

}
```

3.6、List Multipart Uploads

功能说明

List Multiparts Uploads 用来查询正在进行中的分段上传，也就是已经 Created 但是还没有 Aaborted 或者 Completed 的分段上传数据，单次最多列出 1000 个正在进行中的分段上传。

方法原型

```
ListMultipartUploadsResponse
ListMultipartUploads(ListMultipartUploadsRequest request);
```

参数说明

- requeset:类型是个 class，具体定义方法如下:

```
// 设置 Bcuekt 的名称，必须设置
string BucketName{get;set;}

//设置 KeyMarker，只有 Key 大于 KeyMarker 的分段上传信息才会返回
string KeyMarker { get; set; }

//设置返回的分段上传数目
int MaxUploads { get; set; }

//设置 Prefix，只有以 Prefix 作为开头的 Key 的分段上传信息才会返回
string Prefix { get; set; }
```

返回结果说明

- ListMultipartUploadsResponse:是个 class 类型，具体定义函数如下:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义

HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }

// 获取 Bcuekt 的名称
string BucketName{get;set;}

//获取最大返回的分段上传信息数目
int MaxUploads { get; set; }

//是否是截断返回
bool IsTruncated { get; set; }

//获取这次请求的 KeyMarker
string KeyMarker { get; set; }

//获取下次请求的 KeyMarker，只要用于顺序获取分段上传信息数据
string NextKeyMarker { get; set; }
```

```
//获取本次请求的 Prefix
string Prefix { get; set; }

//获取分段上传信息,MultipartUpload 是个 class,具体定义如下:
List<MultipartUpload> MultipartUploads
```

MultipartUpload 具体定义方法如下:

```
//获取 UploadId
string UploadId { get; set; }

//获取 Key
string Key { get; set; }

//获取分段上传初始化时间
DateTime Initiated { get; set; }
```

示例

```
using System;
using Amazon.S3;
using System.Collections.Generic;
using Amazon.S3.Model;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
                            delegate (
                            object sender,
                            X509Certificate certificate,
                            X509Chain chain,
                            SslPolicyErrors sslPolicyErrors) // 自签名忽
略认证
                            {
                                return true;
                            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480"; // 指定
IP 和 Port
            config.ForcePathStyle = true; // 使用路径模式
            config.SignatureVersion = "2"; // "4" 签名版本
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config); // 建立连接
```

```
            ListMultipartUploadsRequest requeset
                = new ListMultipartUploadsRequest();
            requeset.BucketName = "bucket-s6";
            requeset.KeyMarker = "30M";
            requeset.MaxUploads = 2;
            requeset.Prefix = "5";
            ListMultipartUploadsResponse response;

            try
            {
                response = client.ListMultipartUploads(requeset);
                Console.WriteLine(response.BucketName);
                Console.WriteLine(response.MaxUploads);
                Console.WriteLine(response.IsTruncated);
                Console.WriteLine(response.KeyMarker);
                Console.WriteLine(response.NextKeyMarker);
                Console.WriteLine(response.Prefix);

                List<MultipartUpload> multipartuploads =
response.MultipartUploads;
                for(int i = 0; i < multipartuploads.Count; i++)
                {
                    Console.WriteLine(multipartuploads[i].Initiated);
                    Console.WriteLine(multipartuploads[i].Key);
                    Console.WriteLine(multipartuploads[i].UploadId);
                }

            } catch (Exception ex) {
                Console.WriteLine("requeset failed!" + ex.Message);
            }
        }
    }
}
```

4、图片处理

4.1、Post 请求处理图片

功能说明

该功能是对存储桶中的图片进行处理，并将处理后的图片持久化到指定的存储桶中,支持处理图片格式 JPG、PNG、WEBP、BMP、TIFF。

方法原型

```
ProcessObjectResponse ProcessObject(ProcessObjectRequest request)
```

参数说明

- request:类型 ProcessObjectRequest 请求接口的参数，具体定义方法如下:

```
// 设置处理后的图片要存放的存储桶，必须设置
string BucketName{ get; set; }

//设置处理后的图片要保存的名称，必须设置
string Key { get; set; }

//设置图片处理参数，必须设置
string ZosProcess { get; set; }

//设置源图片处理存储桶，源对象以及版本 id，格式为 Bucket/Object?Versionid=id，
//未开启多版本功能可以不填 Versionid，必须设置
string ProcessSource { get; set; }
```

ZosProcess 的定义方法为：

- 图片缩放(e.g. image/resize,w_300,h_200,m_fixed)

参数名称	参数描述	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	参数描述	取值	是否必须
w	指定裁剪宽度。	[0,图片宽度] 默认为最大值。	否
h	指定裁剪高度。	[0,图片高度] 默认为最大值。	否
x	指定裁剪起点横坐标（默认左上角为原	[0,图片边界]	否

	点）。		
y	指定裁剪起点纵坐标（默认左上角为原点）。	[0,图片边界]	否
g	设置裁剪的原点位置。原点按照九宫格的形式分布，一共有九个位置可以设置，为每个九宫格的左上角顶点。	nw：左上(默认) north：中上 ne：右上 west：左中 center：中部 east：右中 sw：左下 south：中下 se：右下	否

- 图片旋转(e.g. image/rotate,45)

参数名称	参数描述	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_aHVkaWUuanBnP3gtem9zLXByb2N1c3M9aW1hZ2UvcmVzaXplLHBfMzAvcm90YXRlLDE4MA==,g_north,t_40)
  - 文字水印(e.g. image/watermark,text_Q2hpbmFOZWxlY29t,type_heiti,color_FF0000,size_40,g_se,t_80)

参数名称	参数用途	取值	是否必须
t	图片水印或文字水印的透明度	[0, 100]	否
x	文字水印距离图片边界的水平距离	[0, 4096] 默认值：10	否
y	文字水印距离图片边界的垂直距离	[0, 4096] 默认值：10	否
text	指定文字水印内容	Base64 编码后的字符串，编码结果字符串中'/'要替换为'_'	否
color	指定文字水印的颜色	RGB 颜色值。 默认：FFFFFF（白色）	否
size	指定文字水印的字体大小	默认值：40	否
type	指定文字水印的字体类型	如 Airal, Helvetica, 支持中文字体包括 yahei(微软雅黑)，heiti(黑体)，kaishu（楷书）,youyuan(幼圆)	否
rotate	指定文字水印顺时针旋转角度	[0, 360] 默认值：0	否
image	指定图片水印名称，水印图片需要和原图存放在相同存储空间	水印图片可以进行预处理（e.g. 水印图片缩放为 30%并旋转 180 度，hudie.jpg?x-zos-process=image/resize,p_30/rotate,180 ），需要转换成 base64 编码，编码结果字符串中'/'要替换为'_'	否
g	指定水印在图片中的位置	nw：左上(默认) north：中上 ne：右上 west：左中 center：中部 east：右中 sw：左下 south：中下	否

		se：右下	

- 格式转化(e.g. image/format,png)

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

- 获取图片信息(e.g. image/info)

返回结果说明

- ProcessObjectResponse:ProcessObject 请求接口的返回参数，其具体定义方法如下:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考 HttpStatusCode 定义
HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考 ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }
```

示例

```
using System;

using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using System.Collections.Generic;
using System.IO;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
        public static string accessKeySecret = "your secret key";

        [Obsolete]
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
            delegate (
                object sender,
                X509Certificate certificate,
                X509Chain chain,
                SslPolicyErrors sslPolicyErrors)
            {
```

```
                return true;
            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "https://192.168.218.130";
            config.ForcePathStyle = true;
            config.SignatureVersion = "4";
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config);
            Zos.ProcessObjectCallBack(client, "bucket_sdk_target",
"new_celibrate.jpg");
        }

        public static void ProcessObjectCallBack(AmazonS3Client
client, string bucketName, string objectName)
        {
            ProcessObjectRequest request = new ProcessObjectRequest
            {
                BucketName = bucketName,
                Key = objectName,
                ZosProcess = "image/rotate,90",
                ProcessSource = "bucket_sdk/celibrate.jpg"
            };
            try
            {
                // Issue request and remember to dispose of the
response
                ProcessObjectResponse response =
client.ProcessObject(request);
                Console.WriteLine("ETag = " + response.ETag);
                Console.WriteLine("VersionId = " + response.VersionId);
                Console.WriteLine("ZOS 存储桶持久化处理对象成功! ");
            }
            catch (Exception ex)
            {
                Console.WriteLine("process object failed!" +
ex.Message);
            }
        }
    }
}
```

4.2、Get 请求获取图片

功能说明

图片处理是在 get_object 基础进行的扩展，用于对存储桶中的图片文件在线处理，支持处理图片格式 JPG、PNG、WEBP、BMP、TIFF。

方法原型

```

GetObjectResponse GetObject(GetObjectRequest request)
```

参数说明

- request:类型 GetObjectRequest 请求接口的参数，具体定义方法如下:

```
// 设置待处理图片所在的存储桶，必须设置
string BucketName{ get; set; }

//设置待处理后的图片的名称，必须设置
string Key { get; set; }

//设置图片处理参数，定义方法同 5.1 参数说明，必须设置
string ZosProcess { get; set; }

//设置对象版本 Id
string VersionId { get; set; }
```

返回结果说明

- GetObjectResponse:GetObject 请求接口的返回参数，其具体定义方法如下:

```
// 获取本次 http 请求的状态码，HttpStatusCode 是个 enum，具体参考
HttpStatusCode 定义
HttpStatusCode HttpStatusCode{get;set;}

//获取本次请求的一些元数据，包括请求 ID 等等，ResponseMetadata 定义参考
ResponseMetadata 定义
ResponseMetadata ResponseMetadata { get; set; }

//下载处理后的图片，filePath 参数指定存放路径
void WriteResponseStreamToFile(string filePath);
```

示例

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System.Threading.Tasks;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using System.Collections.Generic;
using System.IO;

namespace aws_console
{
    class Zos
    {
        public static string accessKeyId = "your access key";
```

```
        public static string accessKeySecret = "your secret key";

        [Obsolete]
        public static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback +=
            delegate (
                object sender,
                X509Certificate certificate,
                X509Chain chain,
                SslPolicyErrors sslPolicyErrors)
            {
                return true;
            };
            AmazonS3Config config = new AmazonS3Config();
            config.ServiceURL = "http://192.168.218.130:7480";
            config.ForcePathStyle = true;
            config.SignatureVersion = "4";
            var client = new AmazonS3Client(accessKeyId,
accessKeySecret, config);
            Zos.GetObjectCallBack(client, "bucket_sdk",
"celibrate.jpg");
        }

        public static void GetObjectCallBack(AmazonS3Client client,
string bucketName, string objectName)
        {
            GetObjectRequest request = new GetObjectRequest
            {
                BucketName = bucketName,
                Key = objectName,
                ZosProcess = "image/rotate,90"
            };
            try
            {
                // Issue request and remember to dispose of the
response
                using (GetObjectResponse response =
client.GetObject(request))
                {
                    // Save object to local file

response.WriteResponseStreamToFile("d:\\File\\compile\\Item1.jpg");
                }

                Console.WriteLine("ZOS 存储桶下载对象成功! ");
            }
            catch (Exception ex)
            {
                Console.WriteLine("get object failed!" + ex.Message);
            }
        }
    }
}
```