对象存储服务

.NET SDK 开发指南

文档版本 01

发布日期 2025-05-16

华为技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

安全声明

漏洞处理流程

华为公司对产品漏洞管理的规定以“漏洞处理流程”为准，该流程的详细内容请参见如下网址：
https://www.huawei.com/cn/psirt/vul-response-process
如企业客户须获取漏洞信息，请参见如下网址：
https://securitybulletin.huawei.com/enterprise/cn/security-advisory

目 录

1 使用前须知

本文介绍.NET SDK的版本变更，并提供版本兼容性说明，以及其他使用前须知。

变更说明

如表1所示，本节将为您展示.NET SDK的版本变更情况和兼容性说明。

表 1-1 .NET SDK 版本变更及兼容性说明

版本	变更类型	说明	是否兼容
v3.22.11.4	-	在NuGet中提供.NET Framework版本。	是
v3.22.11.3	-	在NuGet中提供.NET Core版本。	是
v3.22.3	新特性	支持Content-Disposition标准元数据接口。	是
低于v3.22.3（停止配套，EOM）	-	版本过低，已停止维护，建议及时升级版本。	-

兼容性

- 版本修订记录信息：**ChangeLog**。
- 推荐使用的.NET版本：.NET Framework 3.5、4.0、4.5版本，.NET Core 2.0、3.1版本，.NET 6、7版本。
- 命名空间：**重新整理了旧版本（2.x.x）的命名空间，所有公共接口均调整到OBS和OBS.Model这两个命名空间下**。
- 接口函数：**重新设计了接口函数，与旧版本（2.x.x）不兼容**。

其他使用前须知

- 请确认您已经熟悉OBS的基本概念，如**桶（Bucket）**、**对象（Object）**、**访问密钥（AK和SK）**等。

- 您可以先参考**OBS客户端通用示例**，了解OBS .NET SDK接口调用的通用方式。
- 使用OBS客户端进行接口调用操作完成后，没有异常抛出，则表明返回值有效；如果抛出异常，则说明操作失败，此时应从**SDK自定义异常**实例中获取错误信息。
- 使用OBS客户端进行接口调用成功后，均会返回包含响应头信息的**SDK公共响应头**实例。

2 SDK 下载

SDK 源码和 API 文档

- OBS .NET SDK最新版本源码：**最新版本源码下载**
- OBS .NET SDK历史版本下载地址：**历史版本下载**
- OBS .NET SDK API文档：**OBS .NET SDK API参考**

3 示例程序

OBS .NET SDK提供了丰富的示例程序，方便用户参考或直接使用。您可以从OBS .NET SDK开发包的demo文件夹中获取示例程序。

示例包括以下内容：

示例代码	说明
BucketOperationsSample	展示了桶相关接口的用法
ObjectOperationsSample	展示了对象相关接口的用法
TemporarySignatureSample	展示了使用URL进行授权访问的用法

4 快速入门

4.1 OBS 服务环境搭建

步骤1 注册云服务账号

使用OBS之前必须要有一个云服务账号。

1. 打开浏览器。
2. 登录**公有云网站**。
3. 在页面右上角单击“注册”。
4. 按需填写注册信息并单击“同意协议并注册”。

步骤2 开通OBS服务

使用OBS服务之前必须先充值，才能正常使用OBS服务。

1. 登录**管理控制台**。
2. 单击页面右上角的“费用和成本”进入费用中心页面。
3. 选择“资金管理 > 充值”，系统自动跳转到充值窗口。
4. 根据界面提示信息，对账户进行充值。
5. 充值成功后，关闭充值窗口，返回管理控制台首页。
6. 在服务列表中选择“对象存储服务 OBS”，开通并进入OBS管理控制台。

步骤3 创建访问密钥

OBS通过用户账户中的AK和SK进行签名验证，确保通过授权的账户才能访问指定的OBS资源。以下是对AK和SK的解释说明：

- AK：Access Key ID，接入键标识，用户在对象存储服务系统中的接入键标识，一个接入键标识唯一对应一个用户，一个用户可以同时拥有多个接入键标识。对象存储服务系统通过接入键标识识别访问系统的用户。
- SK：Secret Access Key，安全接入键，用户在对象存储服务系统中的安全接入键，是用户访问对象存储服务系统的密钥，用户根据安全接入键和请求头域生成鉴权信息。安全接入键和接入键标识一一对应。

访问密钥分永久访问密钥（AK/SK）和临时访问密钥（AK/SK和SecurityToken）两种。每个用户最多可创建两个有效的永久访问密钥。临时访问密钥只在设置的有效期

内能够访问OBS，过期后需要重新获取。出于安全性考虑，建议您使用临时访问密钥访问OBS，或使用永久访问密钥访问OBS时，定期更新您的访问密钥（AK/SK）。两种密钥的获取方式如下所示。

- 永久访问密钥：
  a. 登录**管理控制台**。
  b. 单击页面右上角的用户名，并选择“我的凭证”。
  c. 在“我的凭证”页面，单击左侧导航栏的“访问密钥”。
  d. 在“访问密钥”页面，单击“新增访问密钥”。

说明

每个用户最多可创建两个有效的访问密钥。

  e. 在弹出的“新增访问密钥”对话框中，输入描述内容（建议），单击“确定”。

  f. （可选）在弹出的“身份验证”对话框中，选择合适的验证方式进行验证，单击“确定”。

g. 在弹出的“创建成功”提示框中，单击“立即下载”后，密钥会直接保存到浏览器默认的下载文件夹中。

h. 打开下载下来的“credentials.csv”文件即可获取到访问密钥（AK和SK）。

说明

- 在密钥文件中，Access Key ID列对应的值即AK，Secret Access Key列对应的值即SK。
- 为防止访问密钥泄露，建议您将其保存到安全的位置。如果用户在此提示框中单击“取消”，则不会下载密钥，后续也将无法重新下载。如果需要使用访问密钥，可以重新创建新的访问密钥。

- 临时访问密钥：

  临时AK/SK和SecurityToken是系统颁发给用户的临时访问令牌，通过接口设置有效期，范围为15分钟至24小时，过期后需要重新获取。临时AK/SK和SecurityToken遵循权限最小化原则。使用临时AK/SK鉴权时，临时AK/SK和SecurityToken必须同时使用。

  获取临时访问密钥的接口请参考**获取临时AK/SK和securitytoken**。

须知

OBS属于全局级服务，所以在获取临时访问密钥时，需要设置Token的使用范围取值为domain，表示获取的Token可以作用于全局服务，全局服务不区分项目或者区域。

----结束

4.2 安装 SDK

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

方式一、通过visual studio 软件的NuGet包管理器安装SDK，步骤如下：

步骤1 启动Visual Studio，选择“工具 > NuGet包管理器 > 管理解决方案的NuGet程序包”。

步骤2 搜索“esdk”关键词，选择需要集成的程序包，.NET Framework安装HuaweiCloud.ESDK.OBS包，.NET Core 安装HuaweiCloud.ESDK.OBS.Core包。

----结束

方式二、以安装OBS .NET SDK最新版本为例，步骤如下：

步骤1 下载OBS .NET SDK开发包。

步骤2 解压开发包，可以看到其中包含demo文件夹（示例代码）。在release文件夹中包含各个版本的SDK，请选择最新版本解压开发包。可以看到其中包含en文件夹（包含第三方日志库文件log4net.dll和SDK库文件esdk_obs_.net.dll文件），Log4Net.config文件（日志配置文件）。

步骤3 启动Visual Studio，选择“FILE > New > Project > Templates > Visual C# > Console Application”，新建一个Console Application工程。

步骤4 右键单击新建的工程，选择“References > Add Reference…”导入开发包中的SDK库文件esdk_obs_.net.dll。

----结束

4.3 获取服务地址

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

- 您可以从**这里**查看OBS当前开通的服务地址和区域信息。

须知

SDK支持带协议名和不带协议名两种方式传入服务地址，例如获取到的服务地址为“your-endpoint”，则初始化OBS客户端时传入的服务地址可以为“http://your-endpoint”、“https://your-endpoint”和“your-endpoint”三种形式。

4.4 初始化 OBS 客户端

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

向OBS发送任一HTTP/HTTPS请求之前，必须先创建一个ObsClient实例：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
```

```
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 使用访问OBS
```

说明

更多关于OBS客户端初始化的操作请参考“初始化”章节。

日志配置详见**配置SDK日志**。

4.5 创建桶

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

桶是OBS全局命名空间，相当于数据的容器、文件系统的根目录，可以存储若干对象。以下代码展示如何新建一个桶：

```
CreateBucketRequest request = new CreateBucketRequest();
request.BucketName = "bucketname";
client.CreateBucket(request);
```

说明

- 桶的名字是全局唯一的，所以您需要确保不与已有的桶名称重复。
- 桶命名规则如下：
  - 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。
  - 禁止使用类IP地址。
  - 禁止以“-”或“.”开头及结尾。
  - 禁止两个“.”相邻（如：“my..bucket”）。
  - 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。
- 同一用户多次创建同名桶不会报错，创建的桶属性以第一次请求为准。
- 更多创建桶的信息，请参见**创建桶**。

须知

- 创建桶时，如果使用的终端节点归属于默认区域华北-北京一（cn-north-1），则可以不指定区域；如果使用的终端节点归属于其他区域，则必须指定区域，且指定的区域必须与终端节点归属的区域一致。当前有效的区域名称可从**这里**查询。
- 您可以使用**带参数创建**方式，在创建桶时，指定桶的区域位置。

4.6 上传对象

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

本示例用于上传字符串“Hello OBS”到桶名为“bucketname”里，名称为“objectname”。

代码示例如下所示：

```
PutObjectRequest request = new PutObjectRequest
{
    BucketName = "bucketname",
    ObjectKey = "objectname",
    InputStream = new MemoryStream(Encoding.UTF8.GetBytes("Hello OBS"))
};
client.PutObject(request);
```

说明

- 更多上传对象的信息，请参见**上传对象**。

4.7 下载对象

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

本示例用于下载桶名为“bucketname”里，名称为“objectname”的对象。

代码示例如下所示：

```
GetObjectRequest request = new GetObjectRequest()
{
    BucketName = "bucketname",
    ObjectKey = "objectname",
};
using (GetObjectResponse response = client.GetObject(request))
{
    //保存到本地文件
    string dest = "savepath";
    if (!File.Exists(dest))
    {
        response.WriteResponseStreamToFile(dest);
    }
}
```

说明

- 更多下载对象的信息，请参见**下载对象**。

4.8 列举对象

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

当完成一系列上传对象操作后，可能需要查看桶中包含哪些对象。以下代码展示如何列举指定桶中的对象：

```
ListObjectsRequest request = new ListObjectsRequest();
request.BucketName = "bucketname";
ListObjectsResponse response = client.ListObjects(request);
foreach (ObsObject Object in response.ObsObjects)
{
    Console.WriteLine("ObjectKey={0}, Size={1}", Object.ObjectKey, Object.Size);
}
```

说明

- 可通过ListObjectsResponse.ObsObjects获取所有对象（Object）的描述信息。
- 上面的代码默认列举1000个对象（Object）。
- 更丰富的列举功能，请参见**列举对象**。

4.9 删除对象

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

本示例用于删除桶名为“bucketname”里，名称为“objectname”的对象。

代码示例如下所示：

```
DeleteObjectRequest request = new DeleteObjectRequest()
{
    BucketName = "bucketname",
    ObjectKey = "objectname",
};
client.DeleteObject(request);
```

说明

- 本示例仅用于删除单个对象，OBS批量删除对象，需自行遍历构建待批量删除对象的列表。
- 更丰富的删除功能，请参见**删除对象**。

4.10 OBS 客户端通用示例

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

调用ObsClient类的相关接口时，没有异常抛出，则表明返回值有效，返回**SDK公共响应头**子类实例；如果抛出异常，则说明操作失败，此时应从**SDK自定义异常**实例中获取错误信息。OBS客户端提供同步调用和异步调用两种方式，具体示例如下：

同步调用方式

以下代码展示了使用OBS客户端进行同步调用的通用方式：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);

// 同步调用接口，例如创建桶
try
{
    CreateBucketRequest request = new CreateBucketRequest
    {
        BucketName = "bucketname",
    };
    CreateBucketResponse response = client.CreateBucket(request);

    Console.WriteLine("Create bucket response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

异步调用方式

异步调用使用回调函数返回调用结果，以下代码展示了使用OBS客户端进行异步调用的通用方式：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
```

```
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);

// 异步调用接口，例如创建桶
CreateBucketRequest request = new CreateBucketRequest
{
    BucketName = "bucketname",
};
client.BeginCreateBucket(request, delegate(IAsyncResult ar){
  try
  {
      CreateBucketResponse response = client.EndCreateBucket(ar);
      Console.WriteLine("Create bucket response: {0}", response.StatusCode);
   }
   catch (ObsException ex)
   {
      Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
      Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
   }
}, null);
Console.ReadKey();
```


5 初始化

5.1 配置密钥

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

要接入OBS服务，您需要拥有一组有效的访问密钥（AK和SK）用来进行签名认证。

具体可参考**OBS服务环境搭建**。

获取AK和SK之后，您便可以按照以下步骤进行初始化。

- **创建OBS客户端**
- **配置OBS客户端**
- **配置SDK日志**

5.2 创建 OBS 客户端

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

OBS客户端（ObsClient）是访问OBS服务的.NET客户端，它为调用者提供一系列与OBS服务进行交互的接口，用于管理、操作桶（Bucket）和对象（Object）等OBS服务上的资源。使用OBS .NET SDK向OBS发起请求，您需要初始化一个ObsClient实例，并根据需要修改ObsConfig的默认配置项。

- 直接使用服务地址创建OBS客户端（ObsClient），所有配置均为默认值，且后续不支持修改。

- 永久访问密钥（accessKey/secretKey）创建OBS客户端的代码如下

```
// 初始化配置参数
// 认证用的accessKey和secretKey硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, "https://your-endpoint");
// 使用访问OBS
```

- 临时访问密钥（accessKey/secretKey/securityToken）创建OBS客户端的代码如下

```
// 初始化配置参数
// 认证用的accessKey和secretKey硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
string securityToken= "your_securityToken"
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey,securityToken, "https://your-endpoint");
// 使用访问OBS
```

- 使用配置类（ObsConfig）创建OBS客户端（ObsClient），可自定义配置各参数，后续不支持修改，具体参数配置可参见**配置OBS客户端**。

```
// 创建ObsConfig配置类实例
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的accessKey和secretKey硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 使用访问OBS
```

说明

- 您的工程中可以有多个ObsClient，也可以只有一个ObsClient。
- ObsClient是线程安全的，可在并发场景下使用。

5.3 配置 OBS 客户端

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

当使用配置类（ObsConfig）创建OBS客户端（ObsClient）时，您可通过ObsConfig配置类对ObsClient进行配置，可配置代理、连接超时、最大连接数等参数。通过ObsConfig可以设置的参数见下表：

参数	描述	建议值
Endpoint	连接OBS的服务地址。可包含协议类型、域名、端口号。示例：https://your-endpoint:443。（出于安全性考虑，建议使用https协议）	N/A
Timeout	同步调用的超时时间（单位：毫秒）。默认为-1表示不超时。	N/A
ReadWriteTimeout	Socket层传输数据的超时时间（单位：毫秒）。默认为60000毫秒。	[10000, 60000]
AsyncSocketTimeout	异步调用的超时时间（单位：毫秒）。默认为-1表示不超时。	N/A
MaxIdleTime	如果空闲时间超过此参数的设定值，则关闭连接（单位：毫秒）。默认为30000毫秒。	默认
ConnectionLimit	允许打开的最大HTTP连接数。默认为1000。	默认
MaxErrorRetry	请求失败（请求异常、服务端报500或503错误）后最大的重试次数。默认3次。	[1, 5]
ReceiveBufferSize	套接字接收缓冲区的大小。默认为8192。	[8192, 65536]
SecurityProtocolType	使用HTTPS时的加密协议类型。	N/A
ProxyHost	代理服务器的主机地址。	N/A
ProxyPort	代理服务器的端口号。	N/A
ProxyUserName	连接代理服务器时使用的用户名。	N/A
ProxyPassword	连接代理服务器时使用的用户密码。	N/A
ProxyDomain	代理服务器的域。	N/A
ValidateCertificate	是否验证服务端证书。默认为false。	N/A
BufferSize	上传对象到Socket流时的读/写缓存大小（单位：字节）。默认为8192字节。	默认
KeepAlive	是否使用长连接访问OBS服务。默认为true。	N/A

说明

- 建议值为N/A的表示需要根据实际情况进行设置。
- 如网络状况不佳，建议调整Timeout、AsyncSocketTimeout和ReadWriteTimeout的值。
- 如果设置的Endpoint不带协议类型，则默认使用HTTPS协议。
- 出于DNS解析性能和OBS服务可靠性的考虑，不允许将Endpoint设置为IP，必须使用域名访问OBS服务。

5.4 配置 SDK 日志

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

OBS .NET SDK基于Apache Log4net开源库提供了日志功能，您可以通过加入日志配置文件开启日志功能。具体步骤如下：

步骤1 在工程中添加对开发包中的日志库文件log4net.dll的引用。

步骤2 将开发包中的日志配置文件Log4Net.config拷贝至工程目录中bin目录下的Debug或者Release中，确保与工程可执行文件在同一个目录下。

步骤3 根据实际情况修改Log4Net.config中的日志级别。

----结束

说明

- 如果不引入日志库和日志配置文件，则视为关闭日志功能，不会有日志输出。
- 您可以从**日志分析**章节获取更多关于SDK日志的信息。
- 日志文件默认放在工程可执行文件路径下，可在Log4Net.config中进行修改。

6 管理桶

6.1 创建桶

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.CreateBucket创建桶。

带参数创建

创建桶时可以指定桶的访问权限、存储类型和区域位置。OBS支持的桶的存储类型有三类，参见**桶存储类型**。示例代码如下：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 创建桶
try
{
    CreateBucketRequest request = new CreateBucketRequest
    {
        BucketName = "bucketname",
        // 设置桶区域位置
        Location = "bucketLocation",
        // 设置桶的存储类型为归档存储类型
        StorageClass = StorageClassEnum.Cold,
        // 设置桶ACL为公共读，默认是私有
        CannedAcl = CannedAclEnum.Private
    };
```

```
    CreateBucketResponse response = client.CreateBucket(request);
    Console.WriteLine("Create bucket response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- 桶的名字是全局唯一的，所以您需要确保不与已有的桶名称重复。
- 桶命名规则如下：
  - 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。
  - 禁止使用类IP地址。
  - 禁止以“-”或“.”开头及结尾。
  - 禁止两个“.”相邻（如：“my..bucket”）。
  - 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。
- 同一用户在同一区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。
- 本示例创建的桶的**访问权限**默认是私有读写，存储类型默认是标准类型，区域位置是默认区域。

须知

- 创建桶时，如果使用的终端节点归属于默认区域“华北-北京一”（cn-north-1），则可以不指定区域；如果使用的终端节点归属于其他区域，则必须指定区域，且指定的区域必须与终端节点归属的区域一致。当前有效的区域名称可从**这里**查询。
- 您可以使用**带参数创建**方式，在创建桶时，指定桶的区域位置。

简单创建

以下代码展示如何新建一个桶：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 创建桶
try
{
    CreateBucketRequest request = new CreateBucketRequest
    {
        BucketName = "bucketname",
    };
    CreateBucketResponse response = client.CreateBucket(request);
    Console.WriteLine("Create bucket response: {0}", response.StatusCode);
}
catch (ObsException ex)
```

```
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

6.2 列举桶

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.ListBuckets列举桶。以下代码展示如何获取桶列表：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 列举桶
try
{
    ListBucketsRequest request = new ListBucketsRequest();
    ListBucketsResponse response = client.ListBuckets(request);
    request.IsQueryLocation = true;
    foreach (ObsBucket bucket in response.Buckets)
    {
        Console.WriteLine("Bucket name is : {0}", bucket.BucketName);
        Console.WriteLine("Bucket creationDate is : {0}", bucket.CreationDate);
        Console.WriteLine("Bucket location is : {0}", bucket.Location);
        Console.WriteLine("\n");
    }
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- 获取到的桶列表将按照桶名字典顺序排列。
- 设置ListBucketsRequest.IsQueryLocation属性为true后，可在列举桶时查询桶的区域位置。

6.3 删除桶

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.DeleteBucket删除桶。以下代码展示如何删除一个桶：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
//删除桶
try
{
    DeleteBucketRequest request = new DeleteBucketRequest
    {
        BucketName = "bucketname",
    };
    DeleteBucketResponse response = client.DeleteBucket(request);
    Console.WriteLine("Delete bucket response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- 如果桶不为空（包含对象或分段上传碎片），则该桶无法删除。
- 删除桶非幂等操作，删除不存在的桶会报错。

6.4 判断桶是否存在

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.HeadBucket接口判断该桶是否已存在。

本示例用于判断桶名为“bucketname”是否存在。

代码示例如下所示:

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config
// 判断桶是否存在
try
{
    HeadBucketRequest request = new HeadBucketRequest
    {
        BucketName = "bucketname",
    };
    bool exists = client.HeadBucket(request);
    Console.WriteLine("Bucket exists: {0}", exists);
}
catch (ObsException ex)
{
    Console.WriteLine("StatusCode: {0}", ex.StatusCode);
}
```

说明

- 如果抛出异常且HTTP状态码为404，表明桶不存在。

6.5 获取桶元数据

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.GetBucketMetadata接口获取桶元数据。

本示例用于获取桶名为“bucketname”的元数据信息。

代码示例如下所示:

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 获取桶元数据
try
{
```



```
    List<string> headers = new List<string>();
    headers.Add("x-obs-header");
    GetBucketMetadataRequest request = new GetBucketMetadataRequest
    {
        BucketName = "bucketname",
        Origin = "http://www.a.com",
        AccessControlRequestHeaders = headers,
    };
    GetBucketMetadataResponse response = client.GetBucketMetadata(request);
    Console.WriteLine("StorageClass: {0}", response.StorageClass);
    Console.WriteLine("Location: {0}", response.Location);
}
catch (ObsException ex)
{
  Console.WriteLine("StatusCode: {0}", ex.StatusCode);
}
```

说明

- 获取桶元数据过程中返回的错误码含义、问题原因及处理措施可参考**OBS服务端错误码**。

6.6 管理桶 ACL

访问控制列表（Access Control List，ACL）用于资源拥有者给其他账号授予资源的访问权限。默认情况下，创建存储桶或对象时仅资源拥有者对资源的完全控制权限，即桶创建者对桶拥有完全控制权限，对象上传者对对象拥有完全控制权限，而其他账号默认无权访问资源。如果资源拥有者想授予其他账号资源的读写权限，可以使用ACL实现。OBS桶和对象的ACL是基于账号进行授权，授权后对账号和账号下的IAM用户都生效。

了解更多可参见**ACL权限控制方式介绍**。

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

桶ACL可以通过三种方式设置：

1. 创建桶时指定预定义ACL。
2. 调用ObsClient.SetBucketAcl指定预定义ACL。
3. 调用ObsClient.SetBucketAcl自定义ACL。

OBS支持的桶或对象权限包含五类，见下表：

权限	描述	OBS .NET SDK对应值
读权限	如果有桶的读权限，则可以获取该桶内对象列表和桶的元数据。 如果有对象的读权限，则可以获取该对象内容和元数据。	PermissionEnum.Read

权限	描述	OBS .NET SDK对应值
写权限	如果有桶的写权限，则可以上传、覆盖和删除该桶内任何对象。 此权限在对象上不适用。	PermissionEnum.Write
读ACP权限	如果有读ACP的权限，则可以获取对应的桶或对象的权限控制列表（ACL）。 桶或对象的所有者永远拥有读对应桶或对象ACP的权限。	PermissionEnum.ReadAcp
写ACP权限	如果有写ACP的权限，则可以更新对应桶或对象的权限控制列表（ACL）。 桶或对象的所有者永远拥有写对应桶或对象的ACP的权限。 拥有了写ACP的权限，由于可以更改权限控制策略，实际上意味着拥有了完全访问的权限。	PermissionEnum.WriteAcp
完全控制权限	如果有桶的完全控制权限意味着拥有读权限、写权限、读ACP权限和写ACP权限的权限。 如果有对象的完全控制权限意味着拥有读权限、写权限、读ACP权限和写ACP权限的权限。	PermissionEnum.FullControl

OBS预定义的访问策略包含五类，见下表：

权限	描述	OBS .NET SDK对应值
私有读写	桶或对象的所有者拥有完全控制的权限，其他任何人都没有访问权限。	CannedAclEnum.Private
公共读	设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、桶的多版本。 设在对象上，所有人可以获取该对象内容和元数据。	CannedAclEnum.PublicRead
公共读写	设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象、删除对象、初始化段任务、上传段、合并段、拷贝段、取消多段上传任务。 设在对象上，所有人可以获取该对象内容和元数据。	CannedAclEnum.PublicReadWrite

权限	描述	OBS .NET SDK对应值
桶公共读，桶内对象公共读	设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据，可以获取该桶内对象的内容和元数据。 不能应用于对象。	CannedAclEnum.PublicReadDelivered
桶公共读写，桶内对象公共读写	设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象、删除对象、初始化段任务、上传段、合并段、拷贝段、取消多段上传任务，可以获取该桶内对象的内容和元数据。 不能应用于对象。	CannedAclEnum.PublicReadWriteDelivered

创桶时指定预定义 ACL

以下代码展示如何在创建桶时指定预定义ACL：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 创建桶
try
{
    CreateBucketRequest request = new CreateBucketRequest
    {
        BucketName = "bucketname",
        // 设置桶ACL为公共读写
        CannedAcl = CannedAclEnum.PublicReadWrite,
    };
    CreateBucketResponse response = client.CreateBucket(request);
    Console.WriteLine("StatusCode: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

为桶设置预定义 ACL

以下代码展示如何为桶设置预定义ACL：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存
```



```
放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
//设置桶ACL
try
{
    SetBucketAclRequest request = new SetBucketAclRequest
    {
        BucketName = "bucketname",
        // 设置桶ACL为私有
        CannedAcl = CannedAclEnum.Private
    };
    SetBucketAclResponse response = client.SetBucketAcl(request);
    Console.WriteLine("Set bucket acl response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

自定义设置桶 ACL

以下代码展示如何直接自定义设置桶ACL：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
//设置桶ACL
try
{
    //桶的所有者信息
    Owner owner = new Owner
    {
        Id = "ownerid",//所有者的DomainId
    };
    AccessControlList acl = new AccessControlList();
    acl.Owner = owner ;

    Grant item = new Grant()
    {
        Grantee = new GroupGrantee()
        {
            GroupGranteeType = GroupGranteeEnum.AllUsers
        },
        Permission = PermissionEnum.FullControl
    };

    IList<Grant> grants = new List<Grant>();
    grants.Add(item);
    acl.Grants = grants;
```

```
    SetBucketAclRequest request = new SetBucketAclRequest()
    {
        BucketName = "bucketname",
        AccessControlList = acl
    };

    SetBucketAclResponse response = client.SetBucketAcl(request);
    Console.WriteLine("Set bucket acl response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

ACL中需要填写的所有者（Owner）或者被授权用户（Grantee）的ID，是指用户的账号ID，可通过OBS控制台“我的凭证”页面查看。

获取桶 ACL

您可以通过ObsClient.GetBucketAcl获取桶ACL。以下代码展示如何获取桶ACL：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
//获取桶ACL
try
{
    GetBucketAclRequest request = new GetBucketAclRequest
    {
        BucketName = "bucketname",
    };
    GetBucketAclResponse response = client.GetBucketAcl(request);
    Console.WriteLine("Get bucket acl response: {0}", response.StatusCode);
    foreach(Grant grant in response.AccessControlList.Grants)
    {
        if(grant.Grantee is CanonicalGrantee)
        {
            Console.WriteLine("Grantee id: {0}", (grant.Grantee as CanonicalGrantee).Id);
        }else if(grant.Grantee is GroupGrantee)
        {
            Console.WriteLine("Grantee type: {0}", (grant.Grantee as GroupGrantee).GroupGranteeType);
        }
            Console.WriteLine("Grant permission: {0}", grant.Permission);
        }
    }
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```


6.7 管理桶策略

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

除了桶ACL外，桶的拥有者还可以通过桶策略，提供对桶和桶内对象的集中访问控制。

更多关于桶策略的内容请参考**桶策略**。

设置桶策略

您可以通过ObsClient.SetBucketPolicy设置桶策略。示例代码如下：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
try
{
    SetBucketPolicyRequest request = new SetBucketPolicyRequest
    {
        BucketName = "bucketname",
        Policy = "your policy",
    };
    SetBucketPolicyResponse response = client.SetBucketPolicy(request);
    Console.WriteLine("Set bucket policy response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

桶策略内容的具体格式（JSON格式字符串）请参考《对象存储服务API参考》。

获取桶策略

您可以通过ObsClient.GetBucketPolicy获取桶策略。示例代码如下：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-
```

```
cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
try
{
    GetBucketPolicyRequest request = new GetBucketPolicyRequest
    {
        BucketName = "bucketname",
    };
    GetBucketPolicyResponse response = client.GetBucketPolicy(request);
    Console.WriteLine("Get bucket policy response: {0}", response.StatusCode);
    Console.WriteLine("Policy: {0}", response.Policy);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

删除桶策略

您可以通过ObsClient.DeleteBucketPolicy删除桶策略。示例代码如下：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
try
{
    DeleteBucketPolicyRequest request = new DeleteBucketPolicyRequest
    {
        BucketName = "bucketname",
    };
    DeleteBucketPolicyResponse response = client.DeleteBucketPolicy(request);
    Console.WriteLine("Delete bucket policy response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

6.8 获取桶区域位置

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.GetBucketLocation获取桶的区域位置。

本示例用于获取桶名为“bucketname”的区域位置信息。

代码示例如下所示:

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 获取桶区域位置
try
{
    GetBucketLocationRequest request = new GetBucketLocationRequest
    {
        BucketName = "bucketname",
    };
    GetBucketLocationResponse response = client.GetBucketLocation(request);
    Console.WriteLine("Get bucket location response: {0}", response.StatusCode);
    Console.WriteLine("Location: {0}", response.Location);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- 创建桶时可以指定桶的区域位置，请参见**创建桶**。

6.9 获取桶存量信息

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

桶存量信息包括桶已使用的空间大小以及桶包含的对象个数。

您可以通过ObsClient.GetBucketStorageInfo获取桶的存量信息。

本示例用于获取桶名为“bucketname”的存量信息。

代码示例如下所示:

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
```

```
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID", EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey", EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 获取桶存量信息
try
{
    GetBucketStorageInfoRequest request = new GetBucketStorageInfoRequest
    {
        BucketName = "bucketname",
    };
    GetBucketStorageInfoResponse response = client.GetBucketStorageInfo(request);
    Console.WriteLine("Get bucket storageinfo response: {0}", response.StatusCode);
    Console.WriteLine("ObjectNumber: {0}", response.ObjectNumber);
    Console.WriteLine("Size: {0}", response.Size);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- 获取桶存量信息过程中返回的错误码含义、问题原因及处理措施可参考**OBS服务端错误码**。

6.10 桶配额

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

设置桶配额

您可以通过ObsClient.SetBucketQuota设置桶配额。以下代码展示如何设置桶配额：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID", EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey", EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 设置桶配额
try
{
    SetBucketQuotaRequest request = new SetBucketQuotaRequest
    {
        BucketName = "bucketname",
        StorageQuota = 0L,
```



```
    };
    SetBucketQuotaResponse response = client.SetBucketQuota(request);
    Console.WriteLine("Set bucket quota response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

桶配额值类型为字符串，支持的最大值为2^{63} - 1的字符串形式。

获取桶配额

您可以通过ObsClient.GetBucketQuota获取桶配额。以下代码展示如何获取桶配额：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 获取桶配额
try
{
    GetBucketQuotaRequest request = new GetBucketQuotaRequest
    {
        BucketName = "bucketname",
    };
    GetBucketQuotaResponse response = client.GetBucketQuota(request);
    Console.WriteLine("Get bucket quota response: {0}", response.StatusCode);
    Console.WriteLine("StorageQuota: {0}", response.StorageQuota);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

6.11 桶存储类型

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

OBS允许您对桶配置不同的存储类型，桶中对象的存储类型默认将与桶的存储类型保持一致。不同的存储类型可以满足客户业务对存储性能、成本的不同诉求。桶的存储类型分为三类，见下表：

类型	说明	OBS .NET SDK对应值
标准存储	标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。	StorageClassEnum.Standard
低频访问存储	低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。	StorageClassEnum.Warm
归档存储	归档存储适用于很少访问（平均一年访问一次）数据的业务场景。	StorageClassEnum.Cold

更多关于桶存储类型的内容请参考**桶的存储类别**。

设置桶存储类型

您可以通过ObsClient.SetBucketStoragePolicy设置桶存储类型。以下代码展示如何设置桶存储类型：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 设置桶存储类型
try
{
    SetBucketStoragePolicyRequest request = new SetBucketStoragePolicyRequest
    {
        BucketName = "bucketname",
        StorageClass = StorageClassEnum.Cold,
    };
    SetBucketStoragePolicyResponse response = client.SetBucketStoragePolicy(request);
    Console.WriteLine("Set bucket storage policy response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

获取桶存储类型

您可以通过ObsClient.GetBucketStoragePolicy获取桶存储类型。以下代码展示如何获取桶存储类型：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
```

```
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID", EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey", EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 获取桶存储类型
try
{
    GetBucketStoragePolicyRequest request = new GetBucketStoragePolicyRequest()
    {
        BucketName = "bucketName",
    };
    GetBucketStoragePolicyResponse response = client.GetBucketStoragePolicy(request);
    Console.WriteLine("Get bucket storage policy response: {0}", response.StatusCode);
    Console.WriteLine("StorageClass: {0}", response.StorageClass);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```


7 上传对象

7.1 对象上传简介

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

在OBS中，用户操作的基本数据单元是对象。OBS .NET SDK提供了丰富的对象上传接口，可以通过以下方式上传对象：

- **流式上传**
- **文件上传**
- **异步上传**
- **分段上传**
- **追加上传**
- **断点续传上传**

SDK支持上传0KB~5GB的对象。流式上传、文件上传和追加上传的内容大小不能超过5GB；当上传较大文件时，请使用分段上传；分段上传每段内容大小不能超过5GB。

如果上传的对象权限设置为匿名用户读取权限，对象上传成功后，匿名用户可通过链接地址访问该对象数据。对象链接地址格式为：https://*桶名.域名/文件夹目录层级/对象名*。如果该对象存在于桶的根目录下，则链接地址将不需要有文件夹目录层级。

7.2 流式上传

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

流式上传使用System.IO.Stream作为对象的数据源。您可以通过ObsClient.PutObject上传您的数据流到OBS。以下代码展示了如何进行流式上传：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 上传流
try
{
    String str = "Hello OBS";
    Stream stream = new MemoryStream(System.Text.Encoding.Default.GetBytes(str));
    PutObjectRequest request = new PutObjectRequest()
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
        InputStream = stream,
    };
    PutObjectResponse response = client.PutObject(request);
    Console.WriteLine("put object response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- 大文件上传建议使用**分段上传**。
- 上传的内容大小不能超过5GB。

7.3 文件上传

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

文件上传使用本地文件作为对象的数据源。

本示例用于上传本地“localfile”文件到桶名为“bucketname”里，名称为“objectname”的对象。

代码示例如下所示：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
```

```
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID", EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey", EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 上传文件
try
{
    PutObjectRequest request = new PutObjectRequest()
    {
        BucketName = "bucketname",  //待传入目标桶名
        ObjectKey = "objectname",  //待传入对象名(对象名是对象在桶中的完整路径，如folder/test.txt，路径中不包含桶名)
        FilePath = "localfile",//待上传的本地文件路径，需要指定到具体的文件名
    };
    PutObjectResponse response = client.PutObject(request);
    Console.WriteLine("put object response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- 上传的内容大小不能超过5GB。

7.4 异步上传

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.BeginPutObject和ObsClient.EndPutObject，进行异步上传对象。

本示例用于异步上传本地“localfile”文件到桶名为“bucketname”里，名称为“objectname”的对象。

示例代码如下：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID", EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey", EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 异步上传文件
try
```

```
{
    PutObjectRequest request = new PutObjectRequest()
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
        FilePath = "localfile",//待上传的本地文件路径，需要指定到具体的文件名
    };
    client.BeginPutObject(request, delegate(IAsyncResult ar){
        try
        {
            PutObjectResponse response = client.EndPutObject(ar);
            Console.WriteLine("put object response: {0}", response.StatusCode);
        }
        catch (ObsException ex)
        {
            Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
            Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
        }
    }, null);
}
catch (ObsException ex)
{
    Console.WriteLine("Message: {0}", ex.Message);
}
```

说明

- 更多上传对象的信息，请参见**上传对象**。

7.5 获取上传进度

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过PutObjectRequest.UploadProgress注册System.EventHandler回调函数来获取上传的进度。

本示例用于上传本地“localfile”文件到桶名为“bucketname”里，名称为“objectname”的对象并通过System.EventHandler监控上传进度。

代码示例如下所示：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 上传文件
try
{
    PutObjectRequest request = new PutObjectRequest()
```

```
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
        FilePath = "localfile",// 上传的本地文件路径，需要指定到具体的文件名
    };
    // 以传输字节数为基准反馈上传进度
    request.ProgressType = ProgressTypeEnum.ByBytes;
    // 每上传1MB数据反馈上传进度
    request.ProgressInterval = 1024 * 1024;

    // 注册上传进度回调函数
    request.UploadProgress += delegate(object sender, TransferStatus status){
        // 获取上传平均速率
        Console.WriteLine("AverageSpeed: {0}", status.AverageSpeed / 1024  + "KB/S");
        // 获取上传进度百分比
        Console.WriteLine("TransferPercentage: {0}", status.TransferPercentage);
    };
    PutObjectResponse response = client.PutObject(request);
    Console.WriteLine("put object response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- 支持获取上传进度的接口包括：流式上传、文件上传、异步上传、上传段、追加上传和断点续传上传。

7.6 创建文件夹

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

OBS本身是没有文件夹的概念的，桶中存储的元素只有对象。创建文件夹实际上是创建了一个大小为0且对象名以“/”结尾的对象，这类对象与其他对象无任何差异，可以进行下载、删除等操作，只是OBS控制台会将这类以“/”结尾的对象以文件夹的方式展示。

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 创建文件夹
try
{
    PutObjectRequest request = new PutObjectRequest()
    {
```

```
        BucketName = "bucketname",
        ObjectKey = "dir/",
    };
    PutObjectResponse response = client.PutObject(request);
    Console.WriteLine("put object response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- 创建文件夹本质上来说是创建了一个大小为0且对象名以“/”结尾的对象。
- 多级文件夹创建最后一级即可，比如src1/src2/src3/，创建src1/src2/src3/即可，无需创建src1/、src1/src2/。

7.7 设置对象属性

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以在上传对象时设置对象属性。对象属性包含对象MIME类型、对象MD5值（用于校验）、对象存储类型、对象自定义元数据。对象属性可以在多种上传方式下（流式上传、文件上传、分段上传），或**复制对象**时进行设置。

对象属性详细说明见下表：

名称	描述	默认值
对象MIME类型（Content-Type）	对象的MIME类型，定义对象的类型及网页编码，决定浏览器将以什么形式、什么编码读取对象。	binary/octet-stream
对象MD5值（Content-MD5）	对象数据的MD5值（经过Base64编码），提供给OBS服务端，校验数据完整性。	无
对象存储类型	对象的存储类型，不同的存储类型可以满足客户业务对存储性能、成本的不同诉求。默认与桶的存储类型保持一致，可以设置为与桶的存储类型不同。	无
对象自定义元数据	用户对上传到桶中对象的自定义属性描述，以便对对象进行自定义管理。	无

设置对象 MIME 类型

您可以通过PutObjectRequest.ContentType来设置对象MIME类型。以下代码展示如何设置对象MIME类型：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 上传文件
try
{
    PutObjectRequest request = new PutObjectRequest()
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
        FilePath = "localfile",//上传的本地文件路径，需要指定到具体的文件名
        ContentType = "image/jpeg",//对象MIME类型
    };
    PutObjectResponse response = client.PutObject(request);
    Console.WriteLine("put object response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

如果未设置对象MIME类型，SDK会根据上传文件或者对象的后缀名自动判断对象MIME类型，如.xml判断为application/xml文件；.html判断为text/html文件。

设置对象 MD5 值

您可以通过PutObjectRequest.ContentMd5来设置对象MD5值。以下代码展示如何设置对象MD5值：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 上传文件
try
{
    PutObjectRequest request = new PutObjectRequest
    {
```

```
        BucketName = "bucketname",
        ObjectKey = "objectname",
        FilePath = "localfile",//上传的本地文件路径，需要指定到具体的文件名
        ContentMd5 = "your md5 which should be encoded by base64"
    };
    PutObjectResponse response = client.PutObject(request);
    Console.WriteLine("put object response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- 对象数据的MD5值必须经过Base64编码。
- OBS服务端会将该MD5值与对象数据计算出的MD5值进行对比，如果不匹配则上传失败，返回HTTP 400错误。
- 如果不设置对象的MD5值，OBS服务端会忽略对对象数据的MD5值校验。

设置对象存储类型

您可以通过PutObjectRequest.StorageClass来设置对象存储类型。以下代码展示如何设置对象存储类型：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 上传文件
try
{
    PutObjectRequest request = new PutObjectRequest
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
        FilePath ="localfile",//上传的本地文件路径，需要指定到具体的文件名
        StorageClass = StorageClassEnum.Warm,//对象存储类型
    };
    PutObjectResponse response = client.PutObject(request);
    Console.WriteLine("put object response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- 如果未设置，对象的存储类型默认与桶的存储类型保持一致。
- 对象的存储类型分为三类，其含义与**桶存储类型**一致。
- 下载归档存储类型的对象前必须将其恢复。

设置对象自定义元数据

您可以通过PutObjectRequest.Metadata来设置对象自定义元数据。以下代码展示如何设置对象自定义元数据：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 上传文件
try
{
    PutObjectRequest request = new PutObjectRequest
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
        FilePath = "localfile",//上传的本地文件路径，需要指定到具体的文件名
    };
    request.Metadata.Add("meta1", "value1");
    PutObjectResponse response = client.PutObject(request);
    Console.WriteLine("put object response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- 一个对象可以有多个元数据，总大小不能超过8KB。
- 对象的自定义元数据可以通过ObsClient.GetObjectMetadata获取，参见**获取对象元数据**。
- 使用ObsClient.GetObject下载对象时，对象的自定义元数据也会同时下载。

7.8 分段上传

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

对于较大文件上传，可以切分成段上传。用户可以在如下的应用场景内（但不仅限于此），使用分段上传的模式：

- 上传超过100MB大小的文件。
- 网络条件较差，和OBS服务端之间的链接经常断开。
- 上传前无法确定将要上传文件的大小。

分段上传分为如下3个步骤：

步骤1 初始化分段上传任务（ObsClient.InitiateMultipartUpload）。

步骤2 逐个或并行上传段（ObsClient.UploadPart）。

步骤3 合并段（ObsClient.CompleteMultipartUpload）或取消分段上传任务(ObsClient.AbortMultipartUpload)。

----结束

初始化分段上传任务

使用分段上传方式传输数据前，必须先通知OBS初始化一个分段上传任务。该操作会返回一个OBS服务端创建的全局唯一标识（UploadId），用于标识本次分段上传任务。您可以根据这个唯一标识来发起相关的操作，如取消分段上传任务、列举分段上传任务、列举已上传的段等。

您可以通过ObsClient.InitiateMultipartUpload初始化一个分段上传任务。

说明

- 用InitiateMultipartUploadRequest指定上传对象的名称和所属桶。
- 在InitiateMultipartUploadRequest中，您可以设置对象MIME类型、对象存储类型、对象自定义元数据等对象属性。
- InitiateMultipartUploadResponse.UploadId返回分段上传任务的全局唯一标识（UploadId），在后面的操作中将用到它。

上传段

初始化一个分段上传任务之后，可以根据指定的对象名和UploadId来分段上传数据。每一个上传的段都有一个标识它的号码——分段号（PartNumber，范围是1~10000）。对于同一个UploadId，该分段号不但唯一标识这一段数据，也标识了这段数据在整个对象内的相对位置。如果您用同一个分段号上传了新的数据，那么OBS上已有的这个段号的数据将被覆盖。除了最后一段以外，其他段的大小范围是100KB~5GB；最后段大小范围是0~5GB。每个段不需要按顺序上传，甚至可以在不同进程、不同机器上上传，OBS会按照分段号排序组成最终对象。

您可以通过ObsClient.UploadPart上传段。

说明

- 上传段接口要求除最后一段以外，其他的段大小都要大于100KB。但是上传段接口并不会立即校验上传段的大小（因为不知道是否为最后一块）；只有调用合并段接口时才会校验。
- OBS会将服务端收到段数据的ETag值（段数据的MD5值）返回给用户。
- 分段号的范围是1~10000。如果超出这个范围，OBS将返回400 Bad Request错误。
- OBS 3.0的桶支持最小段的大小为100KB，OBS 2.0的桶支持最小段的大小为5MB。请在OBS 3.0的桶上执行分段上传操作。

合并段

所有分段上传完成后，需要调用合并段接口来在OBS服务端生成最终对象。在执行该操作时，需要提供所有有效的分段列表（包括分段号和分段ETag值）；OBS收到提交的分段列表后，会逐一验证每个段的有效性。当所有段验证通过后，OBS将把这些分段组合成最终的对象。

您可以通过ObsClient.CompleteMultipartUpload合并段。

说明

分段可以是不连续的。

示例代码如下：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
try
{
    // 1. 初始化分段上传任务
    InitiateMultipartUploadRequest initiateRequest = new InitiateMultipartUploadRequest
    {
        BucketName = "bucketname",
        ObjectKey = "objectname"
    };

    InitiateMultipartUploadResponse initResponse = client.InitiateMultipartUpload(initiateRequest);
    Console.WriteLine("InitiateMultipartUpload status: {0}", initResponse.StatusCode);
    Console.WriteLine("InitiateMultipartUpload UploadId: {0}", initResponse.UploadId);

    // 2. 上传段
    string filePath = "localfile";//上传的本地文件路径，需要指定到具体的文件名
    long contentLength = new FileInfo(filePath).Length;
    long partSize = 15 * (long)Math.Pow(2, 20); // 每个段大小15 MB
    List<UploadPartResponse> uploadResponses = new List<UploadPartResponse>();
    long filePosition = 0;
    for (int i = 1; filePosition < contentLength; i++)
    {
        UploadPartRequest uploadRequest = new UploadPartRequest
        {
            BucketName = "bucketname",
            ObjectKey = "objectname",
            UploadId = initResponse.UploadId,
            PartNumber = i,
            PartSize = partSize,
            Offset = filePosition,
            FilePath = filePath
        };
        uploadResponses.Add(client.UploadPart(uploadRequest));
        Console.WriteLine("UploadPart count: {0}", uploadResponses.Count);
        filePosition += partSize;
    }

    // 3.合并段

    CompleteMultipartUploadRequest completeRequest = new CompleteMultipartUploadRequest()
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
        UploadId = initResponse.UploadId,
    };
    completeRequest.AddPartETags(uploadResponses);
    CompleteMultipartUploadResponse completeUploadResponse =
```



```
client.CompleteMultipartUpload(completeRequest);
    Console.WriteLine("CompleteMultipartUpload status: {0}", completeUploadResponse.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("Exception:{0}", ex.ErrorCode);
    Console.WriteLine("Exception Message:{0}", ex.ErrorMessage);
}
```

取消分段上传任务

分段上传任务可以被取消，当一个分段上传任务被取消后，就不能再使用其UploadId做任何操作，已经上传段也会被OBS服务端删除。

采用分段上传方式上传对象过程中或上传对象失败后会在桶内产生段，这些段会占用您的存储空间，您可以通过取消该分段上传任务来清理掉不需要的段，节约存储空间。

您可以通过ObsClient.AbortMultipartUpload取消分段上传任务，示例代码如下：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
//取消分段上传任务
try
{
    AbortMultipartUploadRequest request = new AbortMultipartUploadRequest
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
        UploadId = "uploadId",//待取消的分段上传任务的Id
    };
    AbortMultipartUploadResponse response = client.AbortMultipartUpload(request);
    Console.WriteLine("Abort multipart upload response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

列举已上传的段

您可使用ObsClient.ListParts列举出某一分段上传任务所有已经上传成功的段。

该接口可设置的参数如下：

参数	作用	OBS .NET SDK对应属性
BucketName	分段上传任务所属的桶名。	ListPartsRequest.BucketName

参数	作用	OBS .NET SDK对应属性
ObjectKey	分段上传任务所属的对象名。	ListPartsRequest.ObjectKey
UploadId	分段上传任务全局唯一标识，从InitiateMultipartUpload返回的结果获取。	ListPartsRequest.UploadId
MaxParts	表示列举已上传的段返回结果最大段数目，即分页时每一页中段数目。	ListPartsRequest.MaxParts
PartNumberMarker	表示待列出段的起始位置，只有Part Number大于该参数的段会被列出。	ListPartsRequest.PartNumberMarker

- 简单列举

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 简单列举
try
{
    ListPartsRequest request = new ListPartsRequest();
    request.BucketName = "bucketname";
    request.ObjectKey = "objectname";
    request.UploadId = "uploadId";
    ListPartsResponse response = client.ListParts(request);
    Console.WriteLine("List parts response: {0}", response.StatusCode);
    foreach (PartDetail part in response.Parts)
    {
        Console.WriteLine("PartNumber: " + part.PartNumber);
        Console.WriteLine("Size: " + part.Size);
        Console.WriteLine("ETag: " + part.ETag);
        Console.WriteLine("LastModified: " + part.LastModified);
    }
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

列举段至多返回1000个段信息，如果指定的Upload ID包含的段数量大于1000，则返回结果中ListPartsResult.isTruncated为true表明本次没有返回全部段，并可通过ListPartsRespon.NextPartNumberMarker获取下次列举的起始位置。

- 列举所有段

由于ObsClient.ListParts只能列举至多1000个段，如果段数量大于1000，列举所有分段请参考如下示例：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 列举所有段
try
{
    ListPartsRequest request = new ListPartsRequest();
    request.BucketName = "bucketname";
    request.ObjectKey = "objectname";
    request.UploadId = "uploadId";
    ListPartsResponse response;
    do
    {
        response = client.ListParts(request);
        Console.WriteLine("List parts response: {0}", response.StatusCode);
        foreach (PartDetail part in response.Parts)
        {
            Console.WriteLine("PartNumber: " + part.PartNumber);
            Console.WriteLine("Size: " + part.Size);
            Console.WriteLine("ETag: " + part.ETag);
            Console.WriteLine("LastModified: " + part.LastModified);
        }
        request.PartNumberMarker = response.NextPartNumberMarker;
    }
    while (response.IsTruncated);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

列举分段上传任务

您可以通过ObsClient.ListMultipartUploads列举分段上传任务。列举分段上传任务可设置的参数如下：

参数	作用	OBS .NET SDK对应属性
BucketName	桶名。	ListMultipartUploadsRequest.BucketName
Prefix	限定返回的分段上传任务中的对象名必须带有prefix前缀。	ListMultipartUploadsRequest.Prefix

参数	作用	OBS .NET SDK对应属性
Delimiter	用于对分段上传任务中的对象名进行分组的字符。对于对象名中包含Delimiter的任务，其对象名（如果请求中指定了Prefix，则此处的对象名需要去掉Prefix）中从首字符至第一个Delimiter之间的字符串将作为一个分组并作为CommonPrefix返回。	ListMultipartUploadsRequest.Delimiter
MaxUploads	列举分段上传任务的最大数目，取值范围为1~1000，当超出范围时，按照默认的1000进行处理。	ListMultipartUploadsRequest.MaxUploads
KeyMarker	表示列举时返回指定的keyMarker之后的分段上传任务。	ListMultipartUploadsRequest.KeyMarker
UploadIdMarker	只有与KeyMarker参数一起使用时才有意义，用于指定返回结果的起始位置，即列举时返回指定KeyMarker的UploadIdMarker之后的分段上传任务。	ListMultipartUploadsRequest.UploadIdMarker

- 简单列举分段上传任务

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
//简单列举分段上传任务
try
{
    ListMultipartUploadsRequest listMultipartUploadsRequest = new ListMultipartUploadsRequest();
    listMultipartUploadsRequest.BucketName = "bucketname";
    ListMultipartUploadsResponse listMultipartUploadsResponse =
client.ListMultipartUploads(listMultipartUploadsRequest);
    Console.WriteLine("ListMultipartUploadsResponse status code: " +
listMultipartUploadsResponse.StatusCode);
    foreach (MultipartUpload upload in listMultipartUploadsResponse.MultipartUploads)
    {
        Console.WriteLine("ObjectKey {0}: " , upload.ObjectKey);
        Console.WriteLine("Initiated {0}: " , upload.Initiated);
        Console.WriteLine("\n");
    }
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```


说明

- 列举分段上传任务至多返回1000个任务信息，如果指定的桶包含的分段上传任务数量大于1000，则ListMultipartUploadsResponse.isTruncated为true表明本次没有返回全部结果，并可通过ListMultipartUploadsResponse.NextKeyMarker和ListMultipartUploadsResponse.NextUploadIdMarker获取下次列举的起点。
- 如果想获取指定桶包含的所有分段上传任务，可以采用分页列举的方式。

- 分页列举全部分段上传任务

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
//列举全部分段上传任务
try
{
    ListMultipartUploadsRequest request = new ListMultipartUploadsRequest();
    request.BucketName = "bucketname";
    ListMultipartUploadsResponse response;
    do
    {
        response = client.ListMultipartUploads(request);
        Console.WriteLine("ListMultipartUploadsResponse status code: " + response.StatusCode);
        foreach (MultipartUpload upload in response.MultipartUploads)
        {
            Console.WriteLine("ObjectKey {0}: " , upload.ObjectKey);
            Console.WriteLine("Initiated {0}: " , upload.Initiated);
            Console.WriteLine("\n");
        }
        request.KeyMarker = response.NextKeyMarker;
        request.UploadIdMarker = response.NextUploadIdMarker;
    }
    while (response.IsTruncated);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

7.9 设置对象生命周期

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

上传对象或者初始化分段上传任务时，您可以直接指定对象的过期时间。示例代码如下：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
```

```
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);

try
{
    PutObjectRequest request = new PutObjectRequest()
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
        FilePath = "localfile",// 上传的本地文件路径，需要指定到具体的文件名
        Expires = 30 // 上传对象时，设置对象30天后过期
    };
    PutObjectResponse response = client.PutObject(request);
    Console.WriteLine("put object response: {0}", response.StatusCode);

    InitiateMultipartUploadRequest initiateRequest = new InitiateMultipartUploadRequest
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
        // 初始化分段上传任务时，设置合并段后生成的对象60天后过期
        Expires = 60
    };

    InitiateMultipartUploadResponse initResponse = client.InitiateMultipartUpload(initiateRequest);
    Console.WriteLine("InitiateMultipartUpload status: {0}", initResponse.StatusCode);
    Console.WriteLine("InitiateMultipartUpload UploadId: {0}", initResponse.UploadId);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- 上述方式仅支持设置以天为单位的对象过期时间，过期后的对象会被OBS服务端自动清理。
- 上述方式设置的对象过期时间，其优先级高于桶生命周期规则。

7.10 追加上传

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

追加上传可实现对同一个对象追加数据内容的功能。您可以通过ObsClient.AppendObject进行追加上传。示例代码如下：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存
```



```
放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
try
{
    // 第一次追加上传
    AppendObjectRequest request = new AppendObjectRequest();
    request.BucketName = "bucketname";
    request.ObjectKey = "objectkey";
    request.InputStream = new MemoryStream(Encoding.UTF8.GetBytes("Hello OBS"));

    AppendObjectResponse response = client.AppendObject(request);

    // 第二次追加上传
    request.Position = response.NextPosition;
    request.InputStream = new MemoryStream(Encoding.UTF8.GetBytes("Hello OBS Again"));
    response = client.AppendObject(request);
    Console.WriteLine("NextPosition:{0}", response.NextPosition);
    Console.WriteLine("ETag:{0}", response.ETag);

    // 通过获取对象属性接口获取下次追加上传的位置
    GetObjectMetadataResponse metadataResponse = client.GetObjectMetadata("bucketname", "objectkey");
    Console.WriteLine("NextPosition from metadata:{0}", metadataResponse.NextPosition);

}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- ObsClient.PutObject上传的对象可覆盖ObsClient.AppendObject上传的对象，覆盖后对象变为普通对象，不可再进行追加上传。
- 第一次调用追加上传时，如果已存在同名的普通对象，则会抛出异常（HTTP状态码为409）。
- 每次追加上传返回的ETag是当次追加数据内容的ETag，不是完整对象的ETag。
- 单次追加上传的内容不能超过5GB，且最多支持10000次追加上传。
- 追加上传成功后，可通过AppendObjectResponse.NextPosition获取下次追加上传的位置；或者通过ObsClient.GetObjectMetadata接口获取下次追加上传的位置。

7.11 分段复制

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

分段复制是分段上传的一种特殊情况，即分段上传任务中的段通过复制OBS指定桶中现有对象（或对象的一部分）来实现。您可以通过ObsClient.CopyPart来复制段。

本示例用于分段复制桶名为“sourcebucketname”里的“sourceobjectname”对象到桶名为“destbucketname”里的“destobjectname”对象。

代码示例如下所示:

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 复制段
try
{
    CopyPartRequest request = new CopyPartRequest();
    request.BucketName = "bucketname";
    request.ObjectKey = "objectname";
    request.UploadId = "uploadId";
    request.PartNumber = 1;
    request.SourceBucketName = "sourcebucketname";
    request.SourceObjectKey = "sourceobjectname";
    CopyPartResponse response = client.CopyPart(request);
    Console.WriteLine("Copy part response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

7.12 断点续传上传

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

当上传大文件时，经常出现因网络不稳定或程序崩溃导致上传失败的情况。失败后再次重新上传不仅浪费资源，而且当网络不稳定时仍然有上传失败的风险。断点续传上传接口能有效地解决此类问题引起的上传失败，其原理是将待上传的文件分成若干个分段分别上传，并实时地将每段上传结果统一记录在checkpoint文件中，仅当所有分段都上传成功时返回上传成功的结果，否则抛出异常提醒用户再次调用接口进行重新上传（重新上传时因为有checkpoint文件记录当前的上传进度，避免重新上传所有分段，从而节省资源提高效率）。

您可以通过ObsClient.UploadFile进行断点续传上传。该接口可设置的主要参数如下:

参数	作用	OBS .NET SDK对应属性
BucketName	桶名，必选参数。	UploadFileRequest.BucketName
ObjectKey	对象名，必选参数。	UploadFileRequest.ObjectKey
UploadFile	待上传的本地文件路径，必选参数。	UploadFileRequest.UploadFile
UploadPartSize	分段大小，单位字节，取值范围是5MB~5GB，默认为5MB。	UploadFileRequest.UploadPartSize
EnableCheckpoint	是否开启断点续传模式，默认为false，表示不开启。	UploadFileRequest.EnableCheckpoint
CheckpointFile	记录上传进度的文件，只在断点续传模式下有效。当该值为空时，默认与待上传的本地文件同目录。	UploadFileRequest.CheckpointFile
Metadata	对象自定义元数据。	UploadFileRequest.Metadata
EnableCheckSum	是否校验待上传文件的内容，只在断点续传模式下有效。默认为false，表示不校验。	UploadFileRequest.EnableCheckSum
TaskNum	分段上传时的最大并发数，默认为1。	UploadFileRequest.TaskNum
UploadProgress	上传进度回调函数。	UploadFileRequest.UploadProgress
ProgressType	上传进度反馈方式。	UploadFileRequest.ProgressType
ProgressInterval	上传进度反馈间隔。	UploadFileRequest.ProgressInterval
UploadEventHandler	上传事件回调函数。	UploadFileRequest.UploadEventHandler

以下代码展示了如何使用断点续传上传接口上传文件：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
```

```
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 断点续传上传文件
try
{
    UploadFileRequest request = new UploadFileRequest
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
        // 待上传的本地文件路径
        UploadFile = "localpath",
        // 上传段大小为10MB
        UploadPartSize = 10 * 1024 * 1024,
        // 开启断点续传模式
        EnableCheckpoint = true,
    };
    // 以传输字节数为基准反馈上传进度
    request.ProgressType = ProgressTypeEnum.ByBytes;
    // 每上传1MB数据反馈上传进度
    request.ProgressInterval = 1024 * 1024;

    // 注册上传进度回调函数
    request.UploadProgress += delegate(object sender, TransferStatus status){
        // 获取上传平均速率
        Console.WriteLine("AverageSpeed: {0}", status.AverageSpeed / 1024  + "KB/S");
        // 获取上传进度百分比
        Console.WriteLine("TransferPercentage: {0}", status.TransferPercentage);
    };

    // 注册上传事件回调函数
    request.UploadEventHandler += delegate(object sender, ResumableUploadEvent e){
        // 获取上传事件
        Console.WriteLine("EventType: {0}", e.EventType);
    };

    CompleteMultipartUploadResponse response = client.UploadFile(request);
    Console.WriteLine("Upload File response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
   Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
   Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- 断点续传上传接口是利用**分段上传**特性实现的，是对分段上传的封装和加强。
- 断点续传上传接口不仅能在失败重传时节省资源提高效率，还因其对分段进行并发上传的机制能加快上传速度，帮助用户快速完成上传业务；且其对用户透明，用户不用关心CheckpointFile文件的创建和删除、分段任务的切分、并发上传的实现等内部细节。
- **EnableCheckpoint参数**默认是false，代表不启用断点续传模式，此时断点续传上传接口退化成对分段上传的简单封装，不会产生CheckpointFile文件。
- **CheckpointFile参数**和**EnableCheckSum参数**仅在**EnableCheckpoint参数**为true时有效。

8 下载对象

8.1 对象下载简介

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

OBS .NET SDK提供了丰富的对象下载接口，可以通过以下方式下载对象：

- **流式下载**
- **范围下载**
- **异步下载**
- **限定条件下载**
- **断点续传下载**

您可以通过ObsClient.GetObject下载对象。

8.2 流式下载

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

以下代码展示了如何进行流式下载：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
```

```
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID", EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey", EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 下载对象
try
{
    GetObjectRequest request = new GetObjectRequest()
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
    };
    using (GetObjectResponse response = client.GetObject(request))
    {
        string dest = "savepath";
        if (!File.Exists(dest))
        {
            // 将对象的数据流写入文件中
            response.WriteResponseStreamToFile(dest);
        }
        Console.WriteLine("Get object response: {0}", response.StatusCode);
    }
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

响应结果GetObjectResponse中GetObjectResponse.OutputStream为响应流（System.IO.Stream类型），通过操作GetObjectResponse.OutputStream可将对象的内容读取到本地文件或者内存中；用户也可以调用OBS .NET SDK提供的GetObjectResponse.WriteResponseStreamToFile方法，将对象内容下载到本地文件中。

须知

GetObjectResponse.OutputStream获取的响应流一定要调用GetObjectResponse.OutputStream.Close()显式关闭，否则可能造成资源泄露。

8.3 范围下载

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

如果只需要下载对象的其中一部分数据，可以使用范围下载，下载指定范围的数据。

如果指定的下载范围是0~1000，则返回第0到第1000个字节的数据，包括第1000个，共1001字节的数据，即[0， 1000]。如果指定的范围无效，则返回整个对象的数据。

本示例用于下载桶名为“bucketname”里，名称为“objectname”对象的[10,200]范围对应的内容。

示例代码如下：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 下载对象
try
{
    ByteRange byteRange = new ByteRange(10, 200);
    GetObjectRequest request = new GetObjectRequest()
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
        ByteRange = byteRange,
    };
    using (GetObjectResponse response = client.GetObject(request))
    {
        string dest = "savepath";
        if (!File.Exists(dest))
        {
            response.WriteResponseStreamToFile(dest);
        }
        Console.WriteLine("Get object response: {0}", response.StatusCode);
    }
}
catch (ObsException ex)
{
   Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
   Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

8.4 异步下载

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.BeginGetObject和ObsClient.EndGetObject，进行异步下载对象。

本示例用于异步下载桶名为“bucketname”里，名称为“objectname”的对象。

示例代码如下：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-
```

```
cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 异步下载对象
try
{
    GetObjectRequest request = new GetObjectRequest()
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
    };
    client.BeginGetObject(request, delegate(IAsyncResult ar){
        try
        {
            using (GetObjectResponse response = client.EndGetObject(ar))
            {
                string dest = "savepath";
                if (!File.Exists(dest))
                {
                    // 将对象的数据流写入文件中
                    response.WriteResponseStreamToFile(dest);
                }
                Console.WriteLine("Get object response: {0}", response.StatusCode);
             }
        }
        catch (ObsException ex)
        {
            Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
            Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
        }
    }, null);

}
catch (ObsException ex)
{
    Console.WriteLine("Message: {0}", ex.Message);
}
```

说明

- 更多下载对象的信息，请参见**下载对象**。

8.5 获取下载进度

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过GetObjectRequest.DownloadProgress注册System.EventHandler回调函数来获取下载的进度。

本示例用于下载桶名为“bucketname”里，名称为“objectname”的对象并通过System.EventHandler监控下载进度。

代码示例如下所示：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
```



```
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 下载对象
try
{
   GetObjectRequest request = new GetObjectRequest()
   {
       BucketName = "bucketname",
       ObjectKey = "objectname",
   };
   // 以传输字节数为基准反馈下载进度
   request.ProgressType = ProgressTypeEnum.ByBytes;
   // 每下载1MB数据反馈下载进度
   request.ProgressInterval = 1024 * 1024;

   // 注册下载进度回调函数
   request.DownloadProgress += delegate(object sender, TransferStatus status){
       // 获取下载平均速率
       Console.WriteLine("AverageSpeed: {0}", status.AverageSpeed / 1024  + "KB/S");
       // 获取下载进度百分比
       Console.WriteLine("TransferPercentage: {0}", status.TransferPercentage);
   };
   using (GetObjectResponse response = client.GetObject(request))
   {
       string dest = "savepath";
       if (!File.Exists(dest))
       {
          // 将对象的数据流写入文件中
           response.WriteResponseStreamToFile(dest);
       }
       Console.WriteLine("Get object response: {0}", response.StatusCode);
   }
 }
catch (ObsException ex)
{
   Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
   Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- 支持获取下载进度的接口包括：流式下载、范围下载、异步下载和断点续传下载。

8.6 限定条件下载

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

下载对象时，可以指定一个或多个限定条件，满足限定条件时则进行下载，否则抛出异常，下载对象失败。

您可以使用的限定条件如下：

参数	作用	OBS .NET SDK对应属性
If-Modified-Since	如果对象在指定的时间后有修改，则返回对象内容，否则返回错误。	GetObjectRequest.IfModifiedSince
If-Unmodified-Since	如果对象在指定的时间后没有修改，则返回对象内容，否则返回错误。	GetObjectRequest.IfUnmodifiedSince
If-Match	如果对象的ETag值与该参数值相同，则返回对象内容，否则抛出异常。	GetObjectRequest.IfMatch
If-None-Match	如果对象的ETag值与该参数值不相同，则返回对象内容，否则抛出异常。	GetObjectRequest.IfNoneMatch

说明

- 对象的ETag值是指对象数据的MD5校验值。
- 如果包含IfUnmodifiedSince并且不符合或者包含IfMatch并且不符合，抛出异常中HTTP状态码为：412 precondition failed。
- 如果包含IfModifiedSince并且不符合或者包含IfNoneMatch并且不符合，抛出异常中HTTP状态码为：304 Not Modified。

以下代码展示了如何进行限定条件下载：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 限定条件下载对象
try
{
    DateTime datetime = new DateTime(2018, 3, 10, 12, 00, 00);
    GetObjectRequest request = new GetObjectRequest()
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
        IfModifiedSince = datetime,
    };
    using (GetObjectResponse response = client.GetObject(request))
    {
        string dest = "savepath";
        if (!File.Exists(dest))
        {
            response.WriteResponseStreamToFile(dest);
```



```
        }
        Console.WriteLine("Get object response: {0}", response.StatusCode);
    }
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

8.7 重写响应头

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

下载对象时，可以重写部分HTTP/HTTPS响应头信息。可重写的响应头信息见下表：

参数	作用	OBS .NET SDK对应属性
ContentType	重写HTTP/HTTPS响应中的Content-Type	GetObjectRequest.ResponseHeaderOverrides.ContentType
ContentLanguage	重写HTTP/HTTPS响应中的Content-Language	GetObjectRequest.ResponseHeaderOverrides.ContentLanguage
Expires	重写HTTP/HTTPS响应中的Expires	ObjectRepleaceMetadata.Expires
CacheControl	重写HTTP/HTTPS响应中的Cache-Control	GetObjectRequest.ResponseHeaderOverrides.CacheControl
ContentDisposition	重写HTTP/HTTPS响应中的Content-Disposition	GetObjectRequest.ResponseHeaderOverrides.ContentDisposition
ContentEncoding	重写HTTP/HTTPS响应中的Content-Encoding	GetObjectRequest.ResponseHeaderOverrides.ContentEncoding

以下代码展示了如何重写响应头：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
```

```
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 下载对象时重写响应头
try
{
    ResponseHeaderOverrides responseHeaderOverrides = new ResponseHeaderOverrides();
    responseHeaderOverrides.ContentType = "image/jpeg";
    GetObjectRequest request = new GetObjectRequest()
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
        ResponseHeaderOverrides = responseHeaderOverrides,
    };
    GetObjectResponse response = client.GetObject(request);
    Console.WriteLine("Get object response: {0}", response.StatusCode);
    Console.WriteLine("ContentType: {0}", response.ContentType);
    response.Dispose();
}
catch (ObsException ex)
{
   Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
   Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

8.8 获取自定义元数据

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

下载对象成功后会返回对象的自定义元数据。

本示例用于获取桶名为“bucketname”，名称为“objectname”的对象自定义元数据。

代码示例如下所示:

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 下载对象
try
{
    GetObjectRequest request = new GetObjectRequest()
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
    };
    using (GetObjectResponse response = client.GetObject(request))
```

```
    {
        //获取对象自定义元数据
        foreach (string key in response.Metadata.Keys)
        {
            Console.WriteLine("key is :" + key + " value is: " + response.Metadata[key]);
        }
        string dest = "savepath";
        if (!File.Exists(dest))
        {
            response.WriteResponseStreamToFile(dest);
        }
        Console.WriteLine("Get object response: {0}", response.StatusCode);
    }
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

8.9 下载归档存储对象

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

如果要下载归档存储对象，需要先将归档存储对象恢复。恢复归档存储对象的恢复选项可支持两类，见下表：

选项	说明	OBS .NET SDK对应值
快速恢复	恢复耗时1~5分钟。	RestoreTierEnum.Expedited
标准恢复	恢复耗时3~5小时。默认值。	RestoreTierEnum.Standard

⚠ 注意

重复恢复归档存储数据时在延长恢复有效期的同时，也将会对恢复时产生的恢复费用进行重复收取。产生的标准存储类别的对象副本有效期将会延长，并且收取延长时间段产生的标准存储副本费用。

您可以通过ObsClient.RestoreObject恢复归档存储对象。以下代码展示了如何下载归档存储对象：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
```

```
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);

try
{
    RestoreObjectRequest request = new RestoreObjectRequest();
    request.BucketName = "bucketname";
    request.ObjectKey = "objectname";
    request.Days = 5;
    request.Tier = RestoreTierEnum.Expedited;
    // 可选参数，默认情况下，恢复的是最新版本的对象。如果要恢复指定版本的对象，则可携带versionId参数
    // request.VersionId = "versionId";
    RestoreObjectResponse response = client.RestoreObject(request);
    Console.WriteLine("Restore object response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
   Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
   Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- ObsClient.RestoreObject中指定的对象必须是归档存储类型，否则调用该接口会抛出异常。
- RestoreObjectRequest.Days指定恢复对象保存的时间，取值范围是1~30。
- RestoreObjectRequest.Tier指定恢复选项，表示恢复对象所耗的时间。

8.10 断点续传下载

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

当下载大对象到本地文件时，经常出现因网络不稳定或程序崩溃导致下载失败的情况。失败后再次重新下载不仅浪费资源，而且当网络不稳定时仍然有下载失败的风险。断点续传下载接口能有效地解决此类问题引起的下载失败，其原理是将待下载的对象分成若干个分段分别下载，并实时地将每段下载结果统一记录在Checkpoint文件中，仅当所有分段都下载成功时返回下载成功的结果，否则抛出异常提醒用户再次调用接口进行重新下载（重新下载时因为有Checkpoint文件记录当前的下载进度，避免重新下载所有分段，从而节省资源提高效率）。

您可以通过ObsClient.DownloadFile进行断点续传下载。该接口可设置的主要参数如下：

参数	作用	OBS .NET SDK对应属性
BucketName	桶名，必选参数。	DownloadFileRequest.BucketName
ObjectKey	对象名，必选参数。	DownloadFileRequest.ObjectKey

参数	作用	OBS .NET SDK对应属性
DownloadFile	下载对象的本地文件全路径。当该值为空时，默认为当前程序的运行目录。	DownloadFileRequest.DownloadFile
DownloadPartSize	分段大小，单位字节，取值范围是5MB~5GB，默认为5MB。	DownloadFileRequest.DownloadPartSize
EnableCheckpoint	是否开启断点续传模式，默认为false，表示不开启。	DownloadFileRequest.EnableCheckpoint
CheckpointFile	记录下载进度的文件，只在断点续传模式下有效。当该值为空时，默认与下载对象的本地文件路径同目录。	DownloadFileRequest.CheckpointFile
VersionId	对象的版本号。	DownloadFileRequest.VersionId
TaskNum	分段下载时的最大并发数，默认为1。	DownloadFileRequest.TaskNum
DownloadProgress	下载进度回调函数。	DownloadFileRequest.DownloadProgress
ProgressType	下载进度反馈方式。	DownloadFileRequest.ProgressType
ProgressInterval	下载进度反馈间隔。	DownloadFileRequest.ProgressInterval
DownloadEventHandler	下载事件回调函数。	DownloadFileRequest.DownloadEventHandler

以下代码展示了如何使用断点续传下载接口下载对象到本地文件：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 断点续传下载文件
try
{
    DownloadFileRequest request = new DownloadFileRequest
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
        // 待下载的本地文件路径
```

```
        DownloadFile = "savepath",
        // 下载段大小为10MB
        DownloadPartSize = 1024 * 1024 * 10,
        // 开启断点续传模式
        EnableCheckpoint = true,
    };

    // 以传输字节数为基准反馈下载进度
    request.ProgressType = ProgressTypeEnum.ByBytes;
    // 每下载1MB数据反馈下载进度
    request.ProgressInterval = 1024 * 1024;

    // 注册下载进度回调函数
    request.DownloadProgress += delegate(object sender, TransferStatus status){
        // 获取下载平均速率
        Console.WriteLine("AverageSpeed: {0}", status.AverageSpeed / 1024  + "KB/S");
        // 获取下载进度百分比
        Console.WriteLine("TransferPercentage: {0}", status.TransferPercentage);
    };

    // 注册下载事件回调函数
    request.DownloadEventHandler += delegate(object sender, ResumableDownloadEvent e){
        // 获取下载事件
        Console.WriteLine("EventType: {0}", e.EventType);
    };

    GetObjectMetadataResponse response = client.DownloadFile(request);
    Console.WriteLine("Download File response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- 断点续传下载接口是利用**范围下载**特性实现的，是对范围下载的封装和加强。
- 断点续传下载接口不仅能在失败重下时节省资源提高效率，还因其对分段进行并发下载的机制能加快下载速度，帮助用户快速完成下载业务；且其对用户透明，用户不用关心CheckpointFile文件的创建和删除、分段任务的切分、并发下载的实现等内部细节。
- **EnableCheckpoint参数**默认是false，代表不启用断点续传模式，此时断点续传下载接口退化成对范围下载的简单封装，不会产生CheckpointFile文件。
- **CheckpointFile参数**仅在**EnableCheckpoint参数**为true时有效。

8.11 图片处理

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

OBS为用户提供了稳定、安全、高效、易用、低成本的图片处理服务。当要下载的对象是图片文件时，您可以通过传入图片处理参数对图片文件进行图片剪切、图片缩放、图片水印、格式转换等处理。

更多关于图片处理的内容，参见**图片处理特性指南**。

以下代码展示了如何使用下载对象接口实现图片处理：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);

try
{
    GetObjectRequest request = new GetObjectRequest()
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
        // 对图片依次进行缩放、旋转
        ImageProcess = "image/resize,m_fixed,w_100,h_100/rotate,90",
    };
    GetObjectResponse response = client.GetObject(request);
    Console.WriteLine("Get object response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
   Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
   Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- 使用GetObjectRequest.ImageProcess指定图片处理参数。
- 图片处理参数支持级联处理，可对图片文件依次实施多条命令。

9 管理对象

9.1 获取对象属性

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.GetObjectMetadata来获取对象属性，包括对象最后修改时间、版本号、对象自定义元数据等信息。以下代码展示了如何获取对象属性：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);

try
{
    GetObjectMetadataRequest request = new GetObjectMetadataRequest();
    // 指定存储桶名称
    request.BucketName = "bucketname";
    // 指定对象，此处以 example/objectname 为例
    request.ObjectKey = "example/objectname";
    // 获取对象元数据
    using (GetObjectMetadataResponse response = client.GetObjectMetadata(request)) {
        Console.WriteLine("Get object metadata response: {0}", response.StatusCode);
        // 获取对象的ETag值
        Console.WriteLine("Object etag {0}:  ", response.ETag);
        // 获取对象的版本号
        Console.WriteLine("Object versionId {0}:  ", response.VersionId);
        // 获取对象数据的长度，单位是字节
        Console.WriteLine("Object contentLength {0}:  ", response.ContentLength);
    }
}
```



```
catch (ObsException ex)
{
   Console.WriteLine("Message: {0}", ex.Message);
}
```

说明

- 由于 HTTP 编码规范限制，无法发送非 ASCII 码字符，SDK 会在接收响应时使用 url 解码规则解码响应头中的信息。如您的元数据存储的 content-disposition 为“attachment; filename="%E4%B8%AD%E6%96%87.txt"”，则 SDK 获取结果为“attachment; filename="中文.txt"”。

9.2 管理对象 ACL

访问控制列表（Access Control List，ACL）用于资源拥有者给其他账号授予资源的访问权限。默认情况下，创建存储桶或对象时仅资源拥有者对资源的完全控制权限，即桶创建者对桶拥有完全控制权限，对象上传者对对象拥有完全控制权限，而其他账号默认无权访问资源。如果资源拥有者想授予其他账号资源的读写权限，可以使用ACL实现。OBS桶和对象的ACL是基于账号进行授权，授权后对账号和账号下的IAM用户都生效。

了解更多可参见**ACL权限控制方式介绍**。

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

对象ACL可以通过三种方式设置：

1. 上传对象时指定预定义ACL。
2. 调用ObsClient.SetObjectAcl指定预定义ACL。
3. 调用ObsClient.SetObjectAcl自定义设置ACL。

上传对象时指定预定义 ACL

以下代码展示如何在上传对象时指定预定义ACL：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 上传对象设置预定义ACL
try
{
   PutObjectRequest request = new PutObjectRequest
   {
      BucketName = "bucketname",
```

```
        ObjectKey = "objectname",
        // 设置对象ACL为公共读写
        CannedAcl = CannedAclEnum.PublicReadWrite,
    };
    PutObjectResponse response = client.PutObject(request);
    Console.WriteLine("Set object ac response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

为对象设置预定义 ACL

以下代码展示如何为对象设置预定义ACL：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 为对象设置预定义ACL
try
{
    SetObjectAclRequest request = new SetObjectAclRequest();
    request.BucketName = "bucketname";
    request.ObjectKey = "objectname";
    request.CannedAcl = CannedAclEnum.PublicRead;
    SetObjectAclResponse response = client.SetObjectAcl(request);
    Console.WriteLine("Set object acl response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

自定义设置对象 ACL

以下代码展示如何直接设置对象ACL：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 自定义设置对象ACL
try
{
```



```
    SetObjectAclRequest request = new SetObjectAclRequest();
    request.BucketName = "bucketname";
    request.ObjectKey = "objectname";
    request.AccessControlList = new AccessControlList();
    Owner owner = new Owner();
    owner.Id = "owerid";
    request.AccessControlList.Owner = owner;
    Grant item = new Grant();
    item.Permission = PermissionEnum.FullControl;
    item.Grantee = new GroupGrantee(GroupGranteeEnum.AllUsers);
    request.AccessControlList.Grants.Add(item);
    SetObjectAclResponse response = client.SetObjectAcl(request);
    Console.WriteLine("Set object acl response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

ACL中需要填写的所有者（Owner）或者被授权用户（Grantee）的ID，是指用户的账号ID，可通过OBS控制台“我的凭证”页面查看。

获取对象 ACL

您可以通过ObsClient.GetObjectAcl获取对象ACL。以下代码展示如何获取对象ACL：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 获取对象ACL
try
{
    GetObjectAclRequest request = new GetObjectAclRequest();
    request.BucketName = "bucketname";
    request.ObjectKey = "objectname";
    GetObjectAclResponse response = client.GetObjectAcl(request);
    Console.WriteLine("Get bucket acl response: {0}", response.StatusCode);
    foreach(Grant grant in response.AccessControlList.Grants)
    {
        if(grant.Grantee is CanonicalGrantee)
        {
            Console.WriteLine("Grantee id: {0}", (grant.Grantee as CanonicalGrantee).Id);
        }else if(grant.Grantee is GroupGrantee)
        {
            Console.WriteLine("Grantee type: {0}", (grant.Grantee as GroupGrantee).GroupGranteeType);
        }
            Console.WriteLine("Grant permission: {0}", grant.Permission);
        }
    }
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```


9.3 列举对象

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.ListObjects列举出桶里的对象。

该接口可设置的参数如下：

参数	作用	OBS .NET SDK对应属性
BucketName	桶名。	ListObjectsRequest.BucketName
Prefix	限定返回的对象名必须带有prefix前缀。	ListObjectsRequest.Prefix
Marker	列举对象的起始位置，返回的对象列表将是对象名按照字典序排序后该参数以后的所有对象。	ListObjectsRequest.Marker
MaxKeys	列举对象的最大数目，取值范围为1~1000，当超出范围时，按照默认的1000进行处理。	ListObjectsRequest.MaxKeys
Delimiter	用于对对象名进行分组的字符。对于对象名中包含Delimiter的对象，其对象名（如果请求中指定了Prefix，则此处的对象名需要去掉Prefix）中从首字符至第一个Delimiter之间的字符串将作为一个分组并作为CommonPrefix返回。 对于并行文件系统，不携带此参数时默认列举是递归列举此目录下所有内容，会列举子目录。在大数据场景下（目录层级深、目录下文件多）的列举，建议设置[delimiter="/"]，只列举当前目录下的内容，不列举子目录，提高列举效率。	ListObjectsRequest.Delimiter

简单列举

以下代码展示如何简单列举对象，最多返回1000个对象：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
```

```
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 简单列举
try
{
    ListObjectsRequest request = new ListObjectsRequest();
    request.BucketName = "bucketname";
    ListObjectsResponse response = client.ListObjects(request);
    foreach (ObsObject entry in response.ObsObjects)
    {
        Console.WriteLine("key = {0} size = {1}", entry.ObjectKey, entry.Size);
    }
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

每次至多返回1000个对象，如果指定桶包含的对象数量大于1000，则返回结果中ListObjectsResponse.IsTruncated为true表明本次没有返回全部对象，并可通过ListObjectsResponse.NextMarker获取下次列举的起始位置。

指定数目列举

以下代码展示如何指定数目列举对象：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 指定数目列举
try
{
    ListObjectsRequest request = new ListObjectsRequest();
    request.BucketName = "bucketname";
    // 只列举100个对象
    request.MaxKeys = 100;
    ListObjectsResponse response = client.ListObjects(request);
    foreach (ObsObject entry in response.ObsObjects)
    {
        Console.WriteLine("key = {0} size = {1}", entry.ObjectKey, entry.Size);
    }
}
catch (ObsException ex)
{
```

```
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

指定前缀列举

以下代码展示如何指定前缀列举对象：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 指定前缀列举
try
{
    ListObjectsRequest request = new ListObjectsRequest();
    request.BucketName = "bucketname";
    //指定前缀
    request.Prefix = "prefix"; ;
    ListObjectsResponse response = client.ListObjects(request);
    foreach (ObsObject entry in response.ObsObjects)
    {
        Console.WriteLine("key = {0} size = {1}", entry.ObjectKey, entry.Size);
    }
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

指定起始位置列举

以下代码展示如何指定起始位置列举对象：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 指定起始位置列举
try
{
    ListObjectsRequest request = new ListObjectsRequest();
    request.BucketName = "bucketname";
    //指定列举的起始位置
    request.Marker = "marker";
    ListObjectsResponse response = client.ListObjects(request);
    foreach (ObsObject entry in response.ObsObjects)
```

```
    {
        Console.WriteLine("key = {0} size = {1}", entry.ObjectKey, entry.Size);
    }
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

分页列举全部对象

以下代码展示分页列举全部对象：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
//分页列举全部对象
try
{
    ListObjectsRequest request = new ListObjectsRequest();
    ListObjectsResponse response;
    request.BucketName = "bucketname";
    request.MaxKeys = 100;
    do
    {
        response = client.ListObjects(request);
        foreach (ObsObject entry in response.ObsObjects)
        {
            Console.WriteLine("key = {0} size = {1}", entry.ObjectKey, entry.Size);
        }
        request.Marker = response.NextMarker;
    }
    while (response.IsTruncated);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

列举文件夹中的所有对象

OBS本身是没有文件夹的概念的，桶中存储的元素只有对象。文件夹对象实际上是一个大小为0且对象名以“/”结尾的对象，将这个文件夹对象名作为前缀，即可模拟列举文件夹中对象的功能。以下代码展示如何列举文件夹中的对象：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
```

```
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
//列举文件夹中的所有对象
try
{
    ListObjectsRequest request = new ListObjectsRequest();
    ListObjectsResponse response;
    request.BucketName = "bucketname";
    request.MaxKeys = 1000;
    // 设置文件夹对象名"dir/"为前缀
    request.Prefix = "dir/";
    do
    {
        response = client.ListObjects(request);
        foreach (ObsObject entry in response.ObsObjects)
        {
            Console.WriteLine("key = {0} size = {1}", entry.ObjectKey, entry.Size);
        }
        request.Marker = response.NextMarker;
    }
    while (response.IsTruncated);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

9.4 删除对象

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

说明

请您谨慎使用删除操作，如果对象所在的桶未开启多版本控制功能，该对象一旦删除将无法恢复。

删除单个对象

您可以通过ObsClient.DeleteObject删除单个对象。以下代码展示如何删除单个对象：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 删除对象
```

```
try
{
    DeleteObjectRequest request = new DeleteObjectRequest()
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
    };
    DeleteObjectResponse response = client.DeleteObject(request);
    Console.WriteLine("Delete object response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

批量删除对象

您可以通过ObsClient.DeleteObjects批量删除对象。

每次最多删除1000个对象，并支持两种响应模式：详细（verbose）模式和简单（quiet）模式。

- 详细模式：返回的删除成功和删除失败的所有结果，默认模式。
- 简单模式：只返回的删除过程中出错的结果。

以下代码展示了如何进行批量删除对象：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 批量删除对象
try
{
    DeleteObjectsRequest request = new DeleteObjectsRequest();
    request.BucketName = "bucketname";
    request.Quiet = true;
    request.AddKey("objectname1");
    request.AddKey("objectname2");
    DeleteObjectsResponse response = client.DeleteObjects(request);
    Console.WriteLine("Delete objects response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```


9.5 复制对象

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

复制对象特性用来为OBS上已经存在的对象创建一个副本。

您可以通过ObsClient.CopyObject来复制对象。复制对象时，可重新指定新对象的属性和设置对象权限，且支持条件复制。

说明

- 如果待复制的源对象是归档存储类型，则必须先恢复源对象才能进行复制。

简单复制

以下代码展示了如何进行简单复制：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);

// 复制对象
try
{
    CopyObjectRequest request = new CopyObjectRequest();
    request.SourceBucketName = "sourcebucketname";
    request.SourceObjectKey = "sourceobjectname";
    request.BucketName = "destbucketname";
    request.ObjectKey = "destobjectName";
    CopyObjectResponse response = client.CopyObject(request);
    Console.WriteLine("Copy object response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

重写对象属性

以下代码展示了如何在复制对象时重写对象属性：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
```

```
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 重写对象属性
try
{
    CopyObjectRequest request = new CopyObjectRequest();
    request.SourceBucketName = "sourcebucketname";
    request.SourceObjectKey = "sourceobjectname";
    request.BucketName = "destbucketname";
    request.ObjectKey = "destobjectName";
    request.StorageClass = StorageClassEnum.Warm;
    request.ContentType = "image/jpeg";
    request.MetadataDirective = MetadataDirectiveEnum.Replace;
    CopyObjectResponse response = client.CopyObject(request);
    Console.WriteLine("Copy object response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

限定条件复制

复制对象时，可以指定一个或多个限定条件，满足限定条件时则进行复制，否则抛出异常，复制对象失败。

您可以使用的限定条件如下：

参数	作用	OBS .NET SDK对应属性
Copy-Source-If-Modified-Since	如果源对象在指定的时间后有修改，则进行复制，否则抛出异常。	CopyObjectRequest.IfModifiedSince
Copy-Source-If-Unmodified-Since	如果源对象在指定的时间后没有修改，则进行复制，否则抛出异常。	CopyObjectRequest.IfUnmodifiedSince
Copy-Source-If-Match	如果源对象的ETag值与该参数值相同，则进行复制，否则抛出异常。	CopyObjectRequest.IfMatch
Copy-Source-If-None-Match	如果源对象的ETag值与该参数值不相同，则进行复制，否则抛出异常。	CopyObjectRequest.IfNoneMatch

说明

- 源对象的ETag值是指源对象数据的MD5校验值。
- 如果包含IfUnmodifiedSince并且不符合，或者包含IfMatch并且不符合，或者包含IfModifiedSince并且不符合，或者包含IfNoneMatch并且不符合，则复制失败，抛出异常中HTTP状态码为：412 precondition failed。
- IfModifiedSince和IfNoneMatch可以一起使用；IfUnmodifiedSince和IfMatch可以一起使用。

以下代码展示了如何进行限定条件复制：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);

// 限定条件复制
try
{
    CopyObjectRequest request = new CopyObjectRequest();
    request.SourceBucketName = "sourcebucketname";
    request.SourceObjectKey = "sourceobjectname";
    request.BucketName = "destbucketname";
    request.ObjectKey = "destobjectName";

    request.IfModifiedSince = new DateTime(2018, 3, 10, 12, 00, 00);
    CopyObjectResponse response = client.CopyObject(request);
    Console.WriteLine("Copy object response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

重写对象 ACL

以下代码展示了如何在复制对象时重写对象ACL：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 重写对象ACL
try
{
```



```
    CopyObjectRequest request = new CopyObjectRequest();
    request.SourceBucketName = "sourcebucketname";
    request.SourceObjectKey = "sourceobjectname";
    request.BucketName = "destbucketname";
    request.ObjectKey = "destobjectName";
    request.CannedAcl = CannedAclEnum.PublicRead;
    CopyObjectResponse response = client.CopyObject(request);
    Console.WriteLine("Copy object response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

9.6 判断对象是否存在

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.HeadObject来判断指定的对象是否存在。

代码示例如下所示：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 判断指定的对象是否存在
try
{
    HeadObjectRequest request = new HeadObjectRequest()
    {
        BucketName = "bucketName",
        ObjectKey = "objectKey"
    };
    bool response = client.HeadObject(request);
    Console.WriteLine("Head object response: {0}", response);
}
catch (ObsException ex)
{
    Console.WriteLine("Exception errorcode: {0}, when head object.", ex.ErrorCode);
    Console.WriteLine("Exception errormessage: {0}", ex.ErrorMessage);
}
```

说明

- 如果抛出异常且HTTP状态码为404，表明桶不存在。

10 临时授权访问

10.1 使用临时 URL 进行授权访问

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

OBS客户端支持通过访问密钥、请求方法类型、请求参数等信息生成一个在Query参数中携带鉴权信息的URL，可将该URL提供给其他用户进行临时访问。在生成URL时，您需要指定URL的有效期来限制访客用户的访问时长。

如果您想授予其他用户对桶或对象临时进行其他操作的权限（例如上传或下载对象），则需要生成带对应请求的URL后（例如使用生成PUT请求的URL上传对象），将该URL提供给其他用户。

通过该方式可支持的操作以及相关信息见下表：

操作名	HTTP请求方法（OBS .NET SDK对应值）	子资源（OBS .NET SDK对应值）	是否需要桶名	是否需要对象名
创建桶	HttpVerb.PUT	N/A	是	否
获取桶列表	HttpVerb.GET	N/A	否	否
删除桶	HttpVerb.DELETE	N/A	是	否
列举桶内对象	HttpVerb.GET	N/A	是	否
列举桶内多版本对象	HttpVerb.GET	SubResourceEnum.Versions	是	否

操作名	HTTP请求方法（OBS .NET SDK对应值）	子资源（OBS .NET SDK对应值）	是否需要桶名	是否需要对象名
列举分段上传任务	HttpVerb.GET	SubResourceEnum.Uploads	是	否
获取桶元数据	HttpVerb.HEAD	N/A	是	否
获取桶区域位置	HttpVerb.GET	SubResourceEnum.Location	是	否
获取桶存量信息	HttpVerb.GET	SubResourceEnum.StorageInfo	是	否
设置桶配额	HttpVerb.PUT	SubResourceEnum.Quota	是	否
获取桶配额	HttpVerb.GET	SubResourceEnum.Quota	是	否
设置桶存储类型	HttpVerb.PUT	SubResourceEnum.StoragePolicy	是	否
获取桶存储类型	HttpVerb.GET	SubResourceEnum.StoragePolicy	是	否
桶ACL	HttpVerb.PUT	SubResourceEnum.Acl	是	否
获取桶ACL	HttpVerb.GET	SubResourceEnum.Acl	是	否
开启/关闭桶日志	HttpVerb.PUT	SubResourceEnum.Logging	是	否
查看桶日志	HttpVerb.GET	SubResourceEnum.Logging	是	否
设置桶策略	HttpVerb.PUT	SubResourceEnum.Policy	是	否
查看桶策略	HttpVerb.GET	SubResourceEnum.Policy	是	否
删除桶策略	HttpVerb.DELETE	SubResourceEnum.Policy	是	否

操作名	HTTP请求方法（OBS .NET SDK对应值）	子资源（OBS .NET SDK对应值）	是否需要桶名	是否需要对象名
设置生命周期规则	HttpVerb.PUT	SubResourceEnum.Lifecycle	是	否
查看生命周期规则	HttpVerb.GET	SubResourceEnum.Lifecycle	是	否
删除生命周期规则	HttpVerb.DELETE	SubResourceEnum.Lifecycle	是	否
设置托管配置	HttpVerb.PUT	SubResourceEnum.Website	是	否
查看托管配置	HttpVerb.GET	SubResourceEnum.Website	是	否
清除托管配置	HttpVerb.DELETE	SubResourceEnum.Website	是	否
设置桶多版本状态	HttpVerb.PUT	SubResourceEnum.Versioning	是	否
查看桶多版本状态	HttpVerb.GET	SubResourceEnum.Versioning	是	否
设置跨域规则	HttpVerb.PUT	SubResourceEnum.Cors	是	否
查看跨域规则	HttpVerb.GET	SubResourceEnum.Cors	是	否
删除跨域规则	HttpVerb.DELETE	SubResourceEnum.Cors	是	否
设置桶标签	HttpVerb.PUT	SubResourceEnum.Tagging	是	否
查看桶标签	HttpVerb.GET	SubResourceEnum.Tagging	是	否
删除桶标签	HttpVerb.DELETE	SubResourceEnum.Tagging	是	否
上传对象	HttpVerb.PUT	N/A	是	是
追加上传	HttpVerb.POST	SubResourceEnum.Append	是	是

操作名	HTTP请求方法（OBS .NET SDK对应值）	子资源（OBS .NET SDK对应值）	是否需要桶名	是否需要对象名
下载对象	HttpVerb.GET	N/A	是	是
复制对象	HttpVerb.PUT	N/A	是	是
删除对象	HttpVerb.DELETE	N/A	是	是
批量删除对象	HttpVerb.POST	SubResourceEnum.Delete	是	是
获取对象属性	HttpVerb.HEAD	N/A	是	是
设置对象ACL	HttpVerb.PUT	SubResourceEnum.Acl	是	是
查看对象ACL	HttpVerb.GET	SubResourceEnum.Acl	是	是
初始化分段上传任务	HttpVerb.POST	SubResourceEnum.Uploads	是	是
上传段	HttpVerb.PUT	N/A	是	是
复制段	HttpVerb.PUT	N/A	是	是
列举已上传的段	HttpVerb.GET	N/A	是	是
合并段	HttpVerb.POST	N/A	是	是
取消分段上传任务	HttpVerb.DELETE	N/A	是	是
恢复归档存储对象	HttpVerb.POST	SubResourceEnum.Restore	是	是

通过OBS .NET SDK生成临时URL访问OBS的步骤如下：

步骤1 通过ObsClient.CreateTemporarySignature生成带签名信息的URL。

步骤2 使用任意HTTP库发送HTTP/HTTPS请求，访问OBS服务。

----结束

⚠ 注意

如果遇到跨域报错、签名不匹配问题，请参考以下步骤排查问题：

1. 未配置跨域，需要在控制台配置CORS规则，请参考**配置桶允许跨域请求**。
2. 签名计算问题，请参考**URL中携带签名**排查签名参数是否正确；比如上传对象功能，后端将Content-Type参与计算签名生成授权URL，但是前端使用授权URL时没有设置Content-Type字段或者传入错误的值，此时会出现跨域错误。解决方案为：Content-Type字段前后端保持一致。

以下代码展示了如何使用临时URL进行授权访问，包括：创建桶、上传对象、下载对象、列举对象、删除对象。

创建桶

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);

// URL有效期，3600秒
long exipreSeconds = 3600;

CreateTemporarySignatureRequest request = new CreateTemporarySignatureRequest();
request.BucketName = "bucketname";
request.Method = HttpVerb.PUT;
request.Expires = exipreSeconds;

CreateTemporarySignatureResponse response = client.CreateTemporarySignature(request);
Console.WriteLine("Creating bucket using temporary signature url:");
Console.WriteLine("\t" + response.SignUrl);

// 使用PUT请求创建桶
HttpWebRequest webRequest = WebRequest.Create(response.SignUrl) as HttpWebRequest;
webRequest.Method = "PUT";


foreach (KeyValuePair<string, string> header in response.ActualSignedRequestHeaders)
{
    if (!header.Key.Equals("host", StringComparison.OrdinalIgnoreCase))
    {
        webRequest.Headers.Add(header.Key, header.Value);
    }
}

string location = "your bucket location";

webRequest.SendChunked = true;
webRequest.AllowWriteStreamBuffering = false;
using (Stream requestStream = webRequest.GetRequestStream())
{
    byte[] buffer = Encoding.UTF8.GetBytes("<CreateBucketConfiguration><LocationConstraint>" + location
+ "</LocationConstraint></CreateBucketConfiguration>");
    requestStream.Write(buffer, 0, buffer.Length);
}
```

```
HttpWebResponse webResponse = null;
try
{
    webResponse = webRequest.GetResponse() as HttpWebResponse;
}
catch (WebException ex)
{
    webResponse = ex.Response as HttpWebResponse;
}
Console.WriteLine("Response Status:" + Convert.ToInt32(webResponse.StatusCode));
using (MemoryStream dest = new MemoryStream())
{
    using (Stream stream = webResponse.GetResponseStream())
    {
        byte[] buffer = new byte[8192];
        int bytesRead;
        while ((bytesRead = stream.Read(buffer, 0, buffer.Length)) > 0)
        {
            dest.Write(buffer, 0, bytesRead);
        }

    }
    Console.WriteLine("Response Content:");
    Console.WriteLine(Encoding.UTF8.GetString(dest.ToArray()));
}
```

上传对象

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);

// URL有效期，3600秒
long exipreSeconds = 3600;

CreateTemporarySignatureRequest request = new CreateTemporarySignatureRequest();
request.BucketName = "bucketname";
request.ObjectKey = "objectkey";
request.Method = HttpVerb.PUT;
request.Expires = exipreSeconds;

CreateTemporarySignatureResponse response = client.CreateTemporarySignature(request);
Console.WriteLine("Creating object using temporary signature url:");
Console.WriteLine("\t" + response.SignUrl);

// 使用PUT请求上传对象
HttpWebRequest webRequest = WebRequest.Create(response.SignUrl) as HttpWebRequest;
webRequest.Method = "PUT";


foreach (KeyValuePair<string, string> header in response.ActualSignedRequestHeaders)
{
    if (!header.Key.Equals("host", StringComparison.OrdinalIgnoreCase))
    {
        webRequest.Headers.Add(header.Key, header.Value);
    }
}
```



```
webRequest.SendChunked = true;
webRequest.AllowWriteStreamBuffering = false;
using (Stream requestStream = webRequest.GetRequestStream())
{
    byte[] buffer = Encoding.UTF8.GetBytes("Hello OBS");
    requestStream.Write(buffer, 0, buffer.Length);
}

HttpWebResponse webResponse = null;
try
{
    webResponse = webRequest.GetResponse() as HttpWebResponse;
}
catch (WebException ex)
{
    webResponse = ex.Response as HttpWebResponse;
}
Console.WriteLine("Response Status:" + Convert.ToInt32(webResponse.StatusCode));
using (MemoryStream dest = new MemoryStream())
{
    using (Stream stream = webResponse.GetResponseStream())
    {
        byte[] buffer = new byte[8192];
        int bytesRead;
        while ((bytesRead = stream.Read(buffer, 0, buffer.Length)) > 0)
        {
            dest.Write(buffer, 0, bytesRead);
        }

    }
    Console.WriteLine("Response Content:");
    Console.WriteLine(Encoding.UTF8.GetString(dest.ToArray()));
}
```

下载对象

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);

// URL有效期，3600秒
long exipreSeconds = 3600;

CreateTemporarySignatureRequest request = new CreateTemporarySignatureRequest();
request.BucketName = "bucketname";
request.ObjectKey = "objectkey";
request.Method = HttpVerb.GET;
request.Expires = exipreSeconds;

CreateTemporarySignatureResponse response = client.CreateTemporarySignature(request);
Console.WriteLine("Getting object using temporary signature url:");
Console.WriteLine("\t" + response.SignUrl);

// 使用GET请求下载对象
HttpWebRequest webRequest = WebRequest.Create(response.SignUrl) as HttpWebRequest;
webRequest.Method = "GET";
```



```
foreach (KeyValuePair<string, string> header in response.ActualSignedRequestHeaders)
{
    if (!header.Key.Equals("host", StringComparison.OrdinalIgnoreCase))
    {
        webRequest.Headers.Add(header.Key, header.Value);
    }
}

HttpWebResponse webResponse = null;
try
{
    webResponse = webRequest.GetResponse() as HttpWebResponse;
}
catch (WebException ex)
{
    webResponse = ex.Response as HttpWebResponse;
}
Console.WriteLine("Response Status:" + Convert.ToInt32(webResponse.StatusCode));
using (MemoryStream dest = new MemoryStream())
{
    using (Stream stream = webResponse.GetResponseStream())
    {
        byte[] buffer = new byte[8192];
        int bytesRead;
        while ((bytesRead = stream.Read(buffer, 0, buffer.Length)) > 0)
        {
            dest.Write(buffer, 0, bytesRead);
        }

    }
    Console.WriteLine("Response Content:");
    Console.WriteLine(Encoding.UTF8.GetString(dest.ToArray()));
}
```

列举对象

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);

// URL有效期，3600秒
long exipreSeconds = 3600;

CreateTemporarySignatureRequest request = new CreateTemporarySignatureRequest();
request.BucketName = "bucketname";
request.Method = HttpVerb.GET;
request.Expires = exipreSeconds;

CreateTemporarySignatureResponse response = client.CreateTemporarySignature(request);
Console.WriteLine("Getting object list using temporary signature url:");
Console.WriteLine("\t" + response.SignUrl);

// 使用GET请求获取对象列表
HttpWebRequest webRequest = WebRequest.Create(response.SignUrl) as HttpWebRequest;
webRequest.Method = "GET";
```



```
foreach (KeyValuePair<string, string> header in response.ActualSignedRequestHeaders)
{
    if (!header.Key.Equals("host", StringComparison.OrdinalIgnoreCase))
    {
        webRequest.Headers.Add(header.Key, header.Value);
    }
}

HttpWebResponse webResponse = null;
try
{
    webResponse = webRequest.GetResponse() as HttpWebResponse;
}
catch (WebException ex)
{
    webResponse = ex.Response as HttpWebResponse;
}
Console.WriteLine("Response Status:" + Convert.ToInt32(webResponse.StatusCode));
using (MemoryStream dest = new MemoryStream())
{
    using (Stream stream = webResponse.GetResponseStream())
    {
        byte[] buffer = new byte[8192];
        int bytesRead;
        while ((bytesRead = stream.Read(buffer, 0, buffer.Length)) > 0)
        {
            dest.Write(buffer, 0, bytesRead);
        }

    }
    Console.WriteLine("Response Content:");
    Console.WriteLine(Encoding.UTF8.GetString(dest.ToArray()));
}
```

删除对象

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);

// URL有效期，3600秒
long exipreSeconds = 3600;

CreateTemporarySignatureRequest request = new CreateTemporarySignatureRequest();
request.BucketName = "bucketname";
request.ObjectKey = "objectkey";
request.Method = HttpVerb.DELETE;
request.Expires = exipreSeconds;

CreateTemporarySignatureResponse response = client.CreateTemporarySignature(request);
Console.WriteLine("Deleting object using temporary signature url:");
Console.WriteLine("\t" + response.SignUrl);

// 使用DELETE请求删除对象
HttpWebRequest webRequest = WebRequest.Create(response.SignUrl) as HttpWebRequest;
webRequest.Method = "DELETE";
```



```
foreach (KeyValuePair<string, string> header in response.ActualSignedRequestHeaders)
{
    if (!header.Key.Equals("host", StringComparison.OrdinalIgnoreCase))
    {
        webRequest.Headers.Add(header.Key, header.Value);
    }
}

HttpWebResponse webResponse = null;
try
{
    webResponse = webRequest.GetResponse() as HttpWebResponse;
}
catch (WebException ex)
{
    webResponse = ex.Response as HttpWebResponse;
}
Console.WriteLine("Response Status:" + Convert.ToInt32(webResponse.StatusCode));
using (MemoryStream dest = new MemoryStream())
{
    using (Stream stream = webResponse.GetResponseStream())
    {
        byte[] buffer = new byte[8192];
        int bytesRead;
        while ((bytesRead = stream.Read(buffer, 0, buffer.Length)) > 0)
        {
            dest.Write(buffer, 0, bytesRead);
        }

    }
    Console.WriteLine("Response Content:");
    Console.WriteLine(Encoding.UTF8.GetString(dest.ToArray()));
}
```

说明

HttpVerb是OBS .NET SDK定义的枚举类型，代表请求方法类型。

11 多版本控制

11.1 多版本控制简介

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

OBS支持保存一个对象的多个版本，您可以利用多版本控制，在一个桶中保留多个版本的对象。

多版本功能可以方便地检索和还原各个版本，在意外操作或应用程序故障时快速恢复数据。

在默认情况下，OBS中新创建的桶不会开启多版本功能，向同一个桶上传同名的对象时，新上传的对象将覆盖原有的对象。

更多关于多版本控制的内容请参见**多版本控制**。

11.2 设置桶多版本状态

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.SetBucketVersioning设置桶的多版本状态。OBS中的桶支持两种多版本状态：

多版本状态	说明	OBS .NET SDK对应值
启用状态	1. 上传对象时，系统为每一个对象创建一个唯一版本号，上传同名的对象将不再覆盖旧的对象，而是创建新的不同版本号的同名对象。 2. 可以指定版本号下载对象，不指定版本号默认下载最新对象。 3. 删除对象时可以指定版本号删除，不带版本号删除对象仅产生一个带唯一版本号的删除标记，并不删除对象。 4. 列出桶内对象列表（ObsClient.ListObjects）时默认列出最新对象列表，可以指定列出桶内所有版本对象列表（ObsClient.ListVersions）。 5. 除了删除标记外，每个版本的对象存储均需计费。	VersionStatusEnum.Enabled
暂停状态	1. 旧的版本数据继续保留。 2. 上传对象时创建对象的版本号为null，上传同名的对象将覆盖原有同名的版本号为null的对象。 3. 可以指定版本号下载对象，不指定版本号默认下载最新对象。 4. 删除对象时可以指定版本号删除，不带版本号删除对象将产生一个版本号为null的删除标记，并删除版本号为null的对象。 5. 除了删除标记外，每个版本的对象存储均需计费。	VersionStatusEnum.Suspended

以下代码展示了如何设置桶的多版本状态：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 设置桶的多版本状态
try
{
    SetBucketVersioningRequest request = new SetBucketVersioningRequest();
    request.BucketName = "bucketname";
    request.Configuration = new VersioningConfiguration();
```



```
    //开启桶的多版本
    request.Configuration.Status = VersionStatusEnum.Enabled;
    SetBucketVersioningResponse response = client.SetBucketVersioning(request);
    Console.WriteLine("Set bucket version response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

11.3 查看桶多版本状态

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.GetBucketVersioning查看桶的多版本状态。

本示例用于获取桶名为“bucketname”里的多版本状态。

代码示例如下所示:

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 查看桶的多版本状态
try
{
    GetBucketVersioningRequest request = new GetBucketVersioningRequest();
    request.BucketName = "bucketname";
    GetBucketVersioningResponse response = client.GetBucketVersioning(request);
    Console.WriteLine("Get bucket version response: {0}", response.StatusCode);
    Console.WriteLine("Status: {0}", response.Configuration.Status);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- 获取桶多版本状态过程中返回的错误码含义、问题原因及处理措施可参考**OBS服务端错误码**。

11.4 获取多版本对象

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.GetObject接口传入版本号（VersionId）来获取多版本对象。

本示例用于下载桶名为“bucketname”里，名称为“objectname”，指定VersionId的对象。

代码示例如下所示：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 获取多版本对象
try
{
    GetObjectRequest getObjectRequest = new GetObjectRequest()
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
        VersionId = "versionId",
    };
    using (GetObjectResponse response = client.GetObject(getObjectRequest))
    {
        Console.WriteLine("Get object response: {0}", response.StatusCode);
        //将文件保存到本地
        if (!File.Exists("savePath"))
        {
            response.WriteResponseStreamToFile("savePath");
        }
    }
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- 如果版本号为空则默认下载最新版本的对象。

11.5 复制多版本对象

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.CopyObject接口传入版本号（ SourceVersionId ）来复制多版本对象。

本示例用于通过设置SourceVersionId 复制sourcebucketname桶中sourceobjectname的多版本对象，复制到destbucketname桶中destobjectname。

代码示例如下所示：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 复制多版本对象
try
{
    CopyObjectRequest request = new CopyObjectRequest();
    request.SourceBucketName = "sourcebucketname";
    request.SourceObjectKey = "sourceobjectname";
    request.BucketName = "destbucketname";
    request.ObjectKey = "destobjectName";
    request.SourceVersionId = "sourceversionId";
    CopyObjectResponse response = client.CopyObject(request);
    Console.WriteLine("copy object response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- 复制多版本对象过程中返回的错误码含义、问题原因及处理措施可参考**OBS服务端错误码**。

11.6 恢复多版本归档存储对象

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.RestoreObject接口传入版本号（VersionId）来恢复多版本归档存储对象。

本示例用于通过设置versionId值将destbucketname桶中destobjectname的多版本归档存储对象，快速恢复为标准存储对象。

代码示例如下所示：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 恢复多版本归档存储对象
try
{
    RestoreObjectRequest request = new RestoreObjectRequest();
    request.BucketName = "bucketname";
    request.ObjectKey = "objectname";
    request.Days = 5;
    // 使用快速恢复方式，恢复多版本对象
    request.Tier = RestoreTierEnum.Expedited;
    request.VersionId = "versionId";
    RestoreObjectResponse response = client.RestoreObject(request);
    Console.WriteLine("Restore object response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

⚠ 注意

重复恢复归档存储数据时在延长恢复有效期的同时，也将会对恢复时产生的恢复费用进行重复收取。产生的标准存储类别的对象副本有效期将会延长，并且收取延长时间段产生的标准存储副本费用。

11.7 列举多版本对象

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.ListVersions列举多版本对象。

该接口可设置的参数如下：

参数	作用	OBS .NET SDK对应属性
BucketName	桶名。	ListVersionsRequest.BucketName
Prefix	限定返回的对象名必须带有Prefix前缀。	ListVersionsRequest.Prefix
KeyMarker	列举多版本对象的起始位置，返回的对象列表将是对象名按照字典序排序后该参数以后的所有对象。	ListVersionsRequest.KeyMarker
MaxKeys	列举多版本对象的最大数目，取值范围为1~1000，当超出范围时，按照默认的1000进行处理。	ListVersionsRequest.MaxKeys
Delimiter	用于对对象名进行分组的字符。对于对象名中包含Delimiter的对象，其对象名（如果请求中指定了Prefix，则此处的对象名需要去掉Prefix）中从首字符至第一个Delimiter之间的字符串将作为一个分组并作为CommonPrefix返回。	ListVersionsRequest.Delimiter
VersionIdMarker	与KeyMarker配合使用，返回的对象列表将是对象名和版本号按照字典序排序后该参数以后的所有对象。	ListVersionsRequest.VersionIdMarker

说明

- 如果VersionIdMarker不是KeyMarker的一个版本号，则该参数无效。
- ObsClient.ListVersions返回结果包含多版本对象和对象删除标记。

简单列举

以下代码展示如何简单列举多版本对象，最多返回1000个对象：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
//列举多版本对象
try
{
    ListVersionsRequest request = new ListVersionsRequest();
    request.BucketName = "bucketname";
```

```
    ListVersionsResponse response = client.ListVersions(request);
    foreach (ObsObjectVersion objectVersion in response.Versions)
    {
        Console.WriteLine("Key: {0}", objectVersion.ObjectKey);
        Console.WriteLine("VersionId: {0}", objectVersion.VersionId);
    }
    Console.WriteLine("List versions response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

每次至多返回1000个多版本对象，如果指定桶包含的对象数量大于1000，则返回结果中ListVersionsResponse.IsTruncated为true表明本次没有返回全部对象，并可通过ListVersionsResponse.NextKeyMarker和ListVersionsResponse.NextVersionIdMarker获取下次列举的起始位置。

指定数目列举

以下代码展示如何指定数目列举多版本对象：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
//指定数目列举
try
{
    ListVersionsRequest request = new ListVersionsRequest();
    request.BucketName = "bucketname";
    request.MaxKeys = 100;
    ListVersionsResponse response = client.ListVersions(request);
    foreach (ObsObjectVersion objectVersion in response.Versions)
    {
        Console.WriteLine("Key: {0}", objectVersion.ObjectKey);
        Console.WriteLine("VersionId: {0}", objectVersion.VersionId);
    }
    Console.WriteLine("List versions response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

指定前缀列举

以下代码展示如何指定前缀列举多版本对象：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存
```



```
放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
//指定前缀列举
try
{
    ListVersionsRequest request = new ListVersionsRequest();
    request.BucketName = "bucketname";
    request.MaxKeys = 100;
    request.Prefix = "prefix";
    ListVersionsResponse response = client.ListVersions(request);
    foreach (ObsObjectVersion objectVersion in response.Versions)
    {
        Console.WriteLine("Key: {0}", objectVersion.ObjectKey);
        Console.WriteLine("VersionId: {0}", objectVersion.VersionId);
    }
    Console.WriteLine("List versions response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

指定起始位置列举

以下代码展示如何指定起始位置列举多版本对象：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
//指定起始位置列举
try
{
    ListVersionsRequest request = new ListVersionsRequest();
    request.BucketName = "bucketname";
    request.MaxKeys = 100;
    request.Prefix = "prefix";
    request.KeyMarker = "keyMarker";
    ListVersionsResponse response = client.ListVersions(request);
    foreach (ObsObjectVersion objectVersion in response.Versions)
    {
        Console.WriteLine("Key: {0}", objectVersion.ObjectKey);
        Console.WriteLine("VersionId: {0}", objectVersion.VersionId);
    }
    Console.WriteLine("List versions response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
```

```
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

分页列举全部多版本对象

以下代码展示分页列举全部多版本对象：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
//分页列举全部多版本对象
try
{
    ListVersionsRequest request = new ListVersionsRequest();
    request.BucketName = "bucketname";
    request.MaxKeys = 100;
    ListVersionsResponse response;
    do
    {
        response = client.ListVersions(request);
        Console.WriteLine("List versions response: {0}", response.StatusCode);
        foreach (ObsObjectVersion objectVersion in response.Versions)
        {
            Console.WriteLine("Key: {0}", objectVersion.ObjectKey);
            Console.WriteLine("VersionId: {0}", objectVersion.VersionId);
        }
        request.KeyMarker = response.NextKeyMarker;
        request.VersionIdMarker = response.NextVersionIdMarker;
    }
    while (response.IsTruncated);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

列举文件夹中的所有多版本对象

OBS本身是没有文件夹的概念的，桶中存储的元素只有对象。文件夹对象实际上是一个大小为0且对象名以“/”结尾的对象，将这个文件夹对象名作为前缀，即可模拟列举文件夹中对象的功能。以下代码展示如何列举文件夹中的多版本对象：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
```



```
ObsClient client = new ObsClient(accessKey, secretKey, config);
//分页列举全部多版本对象
try
{
    ListVersionsRequest request = new ListVersionsRequest();
    request.BucketName = "bucketname";
    request.MaxKeys = 1000;
    // 设置文件夹对象名"dir/"为前缀
    request.Prefix = "dir/";
    ListVersionsResponse response;
    do
    {
        response = client.ListVersions(request);
        Console.WriteLine("List versions response: {0}", response.StatusCode);
        foreach (ObsObjectVersion objectVersion in response.Versions)
        {
            Console.WriteLine("Key: {0}", objectVersion.ObjectKey);
            Console.WriteLine("VersionId: {0}", objectVersion.VersionId);
        }
        request.KeyMarker = response.NextKeyMarker;
        request.VersionIdMarker = response.NextVersionIdMarker;
    }
    while (response.IsTruncated);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

11.8 多版本对象权限

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

设置多版本对象 ACL

您可以通过ObsClient.SetObjectAcl接口传入版本号（VersionId）设置多版本对象ACL，示例代码如下：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 设置多版本对象ACL
try
{
    SetObjectAclRequest request = new SetObjectAclRequest();
    request.BucketName = "bucketname";
    request.ObjectKey = "objectname";
```

```
    request.VersionId = "versionId";
    request.AccessControlList = new AccessControlList();
    Owner owner = new Owner();
    owner.Id = "ownerid";
    request.AccessControlList.Owner = owner;
    Grant item = new Grant();
    item.Permission = PermissionEnum.FullControl;
    item.Grantee = new GroupGrantee(GroupGranteeEnum.AllUsers);
    request.AccessControlList.Grants.Add(item);
    SetObjectAclResponse response = client.SetObjectAcl(request);
    Console.WriteLine("Set object acl response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

ACL中需要填写的所有者（Owner）或者被授权用户（Grantee）的ID，是指用户的账号ID，可通过OBS控制台“我的凭证”页面查看。

获取多版本对象 ACL

您可以通过ObsClient.GetObjectAcl接口传入版本号（VersionId）获取多版本对象ACL，示例代码如下：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 获取多版本访问权限
try
{
    GetObjectAclRequest request = new GetObjectAclRequest();
    request.BucketName = "bucketname";
    request.ObjectKey = "objectname";
    request.VersionId = "versionId";
    GetObjectAclResponse response = client.GetObjectAcl(request);
    Console.WriteLine("GetObjectAcl grant account: {0}", response.AccessControlList.Grants.Count);
    Console.WriteLine("GetObjectAcl owner id: {0}", response.AccessControlList.Owner.Id);
    foreach (Grant grant in response.AccessControlList.Grants)
    {
        if(grant.Grantee is CanonicalGrantee)
        {
            Console.WriteLine("Grantee id: {0}", (grant.Grantee as CanonicalGrantee).Id);
        }else if(grant.Grantee is GroupGrantee)
        {
            Console.WriteLine("Grantee type: {0}", (grant.Grantee as GroupGrantee).GroupGranteeType);
        }
            Console.WriteLine("Grant permission: {0}", grant.Permission);
        }
    }
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
```



```
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

11.9 删除多版本对象

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

删除单个多版本对象

您可以通过ObsClient.DeleteObject接口传入版本号（VersionId）删除多版本对象。

本示例用于通过设置多版本对象的VersionId值，删除桶名为“bucketname”里，名称为“objectname”的对象。

代码示例如下所示：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 删除单个多版本对象
try
{
    DeleteObjectRequest request = new DeleteObjectRequest()
    {
        BucketName = "buckername",
        ObjectKey = "objectname",
        VersionId = "versionId"
    };
    DeleteObjectResponse response = client.DeleteObject(request);
    Console.WriteLine("Delete object response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

批量删除多版本对象

您可以通过ObsClient.DeleteObjects接口传入每个待删除对象的版本号（VersionId）批量删除多版本对象。

本示例用于通过设置多版本对象的VersionId值，批量删除桶名为“bucketname”里，名称为“objectname1”和“objectname2”的对象。

代码示例如下所示：


```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 删除两个多版本对象
try
{
    DeleteObjectsRequest request = new DeleteObjectsRequest();
    request.BucketName = "bucketname";
    request.Quiet = true;
    request.AddKey("objectName1", "versionId1");
    request.AddKey("objectName2", "versionId2");
    DeleteObjectsResponse response = client.DeleteObjects(request);
    Console.WriteLine("Delete objects response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```


12 生命周期管理

12.1 生命周期管理简介

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

OBS允许您对桶设置生命周期规则，实现自动转换对象的存储类型、自动淘汰过期的对象，以有效利用存储特性，优化存储空间。针对不同前缀的对象，您可以同时设置多条规则。一条规则包含：

- 规则ID，用于标识一条规则，不能重复。
- 受影响的对象前缀，此规则只作用于符合前缀的对象。
- 最新版本对象的转换策略，指定方式为：
  a. 指定满足前缀的对象创建后第几天时转换为指定的存储类型。
  b. 直接指定满足前缀的对象转换为指定的存储类型的日期。
- 最新版本对象过期时间，指定方式为：
  a. 指定满足前缀的对象创建后第几天时过期。
  b. 直接指定满足前缀的对象过期日期。
- 历史版本对象转换策略，指定方式为：
  - 指定满足前缀的对象成为历史版本后第几天时转换为指定的存储类型。
- 历史版本对象过期时间，指定方式为：
  - 指定满足前缀的对象成为历史版本后第几天时过期。
- 是否生效标识。

更多关于生命周期的内容请参考**生命周期管理**。

说明

- 对象过期后会被OBS服务端自动删除。
- 对象转换策略中的时间必须早于对象过期时间；历史版本对象转换策略中的时间也必须早于历史版本对象的过期时间。
- 桶的多版本状态必须处于Enabled或者Suspended，历史版本对象转换策略和历史版本对象过期时间配置才能生效。

12.2 设置生命周期规则

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.SetBucketLifecycle设置桶的生命周期规则。

设置对象转换策略

以下代码展示了如何设置最新版本对象和历史版本对象的转换策略：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
//设置对象转换策略
try
{
    SetBucketLifecycleRequest request = new SetBucketLifecycleRequest();
    request.BucketName = "bucketname";
    request.Configuration = new LifecycleConfiguration();
    LifecycleRule rule1 = new LifecycleRule();
    rule1.Id = "rule1";
    rule1.Prefix = "prefix";
    rule1.Status = RuleStatusEnum.Enabled;
    Transition transition = new Transition();
    rule1.Transitions.Add(transition);
    // 指定满足前缀的对象创建30天后转换
    transition.Days = 30;
    // 指定对象转换后的存储类型
    transition.StorageClass = StorageClassEnum.Warm;
    NoncurrentVersionTransition noncurrentVersionTransition  = new NoncurrentVersionTransition();
    rule1.NoncurrentVersionTransitions.Add(noncurrentVersionTransition);
    // 指定满足前缀的对象成为历史版本60天后转换
    noncurrentVersionTransition.NoncurrentDays = 60;
    // 指定历史版本对象转换后的存储类型
    noncurrentVersionTransition.StorageClass = StorageClassEnum.Warm;
    request.Configuration.Rules.Add(rule1);
    SetBucketLifecycleResponse response = client.SetBucketLifecycle(request);
    Console.WriteLine("Set bucket lifecycle response: {0}", response.StatusCode);
}
```

```
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

设置对象过期时间

以下代码展示了如何设置最新版本对象和历史版本对象的过期时间：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 设置对象过期时间
try
{
    SetBucketLifecycleRequest request = new SetBucketLifecycleRequest();
    request.BucketName = "bucketname";
    request.Configuration = new LifecycleConfiguration();
    LifecycleRule rule1 = new LifecycleRule();
    rule1.Id = "rule1";
    rule1.Prefix = "prefix";
    rule1.Status = RuleStatusEnum.Enabled;
    rule1.Expiration = new Expiration();
    //指定满足前缀的对象创建60天后过期
    rule1.Expiration.Days = 60;
    // 指定满足前缀的对象成为历史版本60天后过期
    rule1.NoncurrentVersionExpiration = new NoncurrentVersionExpiration();
    rule1.NoncurrentVersionExpiration.NoncurrentDays = 60;
    request.Configuration.Rules.Add(rule1);
    SetBucketLifecycleResponse response = client.SetBucketLifecycle(request);
    Console.WriteLine("Set bucket lifecycle response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- 调用设置生命周期规则后会将原有规则覆盖。

12.3 查看生命周期规则

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.GetBucketLifecycle查看桶的生命周期规则。

本示例用于查看桶名为“bucketname”的生命周期规则。

代码示例如下所示:

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 查看桶的生命周期
try
{
    GetBucketLifecycleRequest request = new GetBucketLifecycleRequest();
    request.BucketName = "bucketname";
    GetBucketLifecycleResponse response = client.GetBucketLifecycle(request);
    foreach (LifecycleRule lifecycleRule in response.Configuration.Rules)
    {
        Console.WriteLine("Lifecycle rule id: {0}", lifecycleRule.Id);
        Console.WriteLine("Lifecycle rule prefix: {0}", lifecycleRule.Prefix);
        Console.WriteLine("Lifecycle rule status: {0}", lifecycleRule.Status);
        if (null != lifecycleRule.Expiration)
        {
            Console.WriteLine("expiration days: {0}", lifecycleRule.Expiration.Days);
        }
        if (null != lifecycleRule.NoncurrentVersionExpiration)
        {
            Console.WriteLine("NoncurrentVersionExpiration NoncurrentDays: {0}",
lifecycleRule.NoncurrentVersionExpiration.NoncurrentDays);
        }
        foreach (Transition transition in lifecycleRule.Transitions)
        {
            Console.WriteLine("Transition Days : {0}", transition.Days);
            Console.WriteLine("Transition StorageClass : {0}", transition.StorageClass);
        }
        foreach (NoncurrentVersionTransition noncurrentVersionTransition in
lifecycleRule.NoncurrentVersionTransitions)
        {
            Console.WriteLine("NoncurrentVersionTransition NoncurrentDays: {0}",
noncurrentVersionTransition.NoncurrentDays);
            Console.WriteLine("NoncurrentVersionTransition StorageClass : {0}",
noncurrentVersionTransition.StorageClass);
        }
        Console.WriteLine("Get bucket lifecycle response: {0}", response.StatusCode);
    }
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- 查看生命周期规则过程中返回的错误码含义、问题原因及处理措施可参考**OBS服务端错误码**。

12.4 删除生命周期规则

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.DeleteBucketLifecycle删除桶的生命周期规则。

本示例用于删除桶名为“bucketname”的生命周期规则。

代码示例如下所示:

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 删除桶生命规则
try
{
    DeleteBucketLifecycleRequest request = new DeleteBucketLifecycleRequest();
    request.BucketName = "bucketname";
    DeleteBucketLifecycleResponse response = client.DeleteBucketLifecycle(request);
    Console.WriteLine("Delete bucket lifecycle response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- 删除生命周期规则过程中返回的错误码含义、问题原因及处理措施可参考**OBS服务端错误码**。

13 跨域资源共享

13.1 跨域资源共享简介

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

跨域是指不同域名之间相互访问。跨域访问是浏览器出于安全考虑而设置的一个限制，即同源策略。

由于JavaScript同源策略的限制，A域名下的JavaScript无法操作B域名或C域名下的对象。

同协议、同域名（或IP）、以及同端口视为同一个域。两个页面的协议、域名和端口（如果指定了端口）相同，则视为同源。

跨域资源共享（CORS）允许Web端的应用程序访问不属于本域的资源。OBS提供接口方便开发者控制跨域访问的权限。

更多关于跨域资源共享的内容请参考**跨域资源访问**。

13.2 设置跨域规则

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.SetBucketCors设置桶的跨域规则，如果原规则存在则覆盖原规则。

本示例用于设置桶名为“bucketname”的跨域规则。

代码示例如下所示:

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 设置桶跨域规则
try
{
    SetBucketCorsRequest request = new SetBucketCorsRequest();
    request.BucketName = "bucketname";
    request.Configuration = new CorsConfiguration();
    CorsRule rule = new CorsRule();
    rule.Id = "20170820";
    // 指定允许跨域请求的来源
    rule.AllowedOrigins.Add("http://www.a.com");
    rule.AllowedOrigins.Add("http://www.b.com");
    // 控制在OPTIONS预取指令中Access-Control-Request-Headers头中指定的header是否被允许使用
    rule.AllowedHeaders.Add("x-obs-header");
    // 指定允许的跨域请求方法(GET/PUT/DELETE/POST/HEAD)
    rule.AllowedMethods.Add(HttpVerb.HEAD);
    rule.AllowedMethods.Add(HttpVerb.PUT);
    rule.AllowedMethods.Add(HttpVerb.GET);
    rule.AllowedMethods.Add(HttpVerb.POST);
    rule.AllowedMethods.Add(HttpVerb.DELETE);
    // 指定允许用户从应用程序中访问的header
    rule.ExposeHeaders.Add("x-obs-test1");
    rule.ExposeHeaders.Add("x-obs-test2");
    rule.MaxAgeSeconds = 100;
    request.Configuration.Rules.Add(rule);
    SetBucketCorsResponse response = client.SetBucketCors(request);
    Console.WriteLine("Set bucket cors response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- AllowedOrigins、AllowedMethods、AllowedHeaders都能够最多支持一个“*”通配符。“*”表示对于所有的域来源、操作或者头域都满足。

13.3 查看跨域规则

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.GetBucketCors查看桶的跨域规则。

本示例用于查看桶名为“bucketname”的跨域规则。

代码示例如下所示:

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 查看跨域规则
try
{

    GetBucketCorsRequest request = new GetBucketCorsRequest();
    request.BucketName = "bucketname";
    GetBucketCorsResponse response = client.GetBucketCors(request);
    foreach (CorsRule rule in response.Configuration.Rules)
    {
        Console.WriteLine("rule id is: {0}\n", rule.Id);
        foreach (string allowOrigin in rule.AllowedOrigins)
        {
            Console.WriteLine("allowOrigin is: {0}\n", allowOrigin);
        }
        foreach (string allowHeader in rule.AllowedHeaders)
        {
            Console.WriteLine("allowHeader is: {0}\n", allowHeader);
        }
        foreach (HttpVerb allowMethod in rule.AllowedMethods)
        {
            Console.WriteLine("allowMethod is: {0}\n", allowMethod);
        }
        foreach (string exposeHeader in rule.ExposeHeaders)
        {
            Console.WriteLine("exposeHeader is: {0}\n", exposeHeader);
        }
        Console.WriteLine("rule maxAgeSeconds is: {0}\n", rule.MaxAgeSeconds);

    }
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- 查看跨域规则过程中返回的错误码含义、问题原因及处理措施可参考**OBS服务端错误码**。

13.4 删除跨域规则

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.DeleteBucketCors删除桶的跨域规则。

本示例用于删除桶名为“bucketname”的跨域规则。

代码示例如下所示:

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 删除桶跨域规则
try
{
    DeleteBucketCorsRequest request = new DeleteBucketCorsRequest();
    request.BucketName = "bucketname";
    DeleteBucketCorsResponse response = client.DeleteBucketCors(request);
    Console.WriteLine("Delete bucket cors response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- 删除跨域规则过程中返回的错误码含义、问题原因及处理措施可参考**OBS服务端错误码**。

14 设置访问日志

14.1 日志简介

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

OBS允许您对桶设置访问日志记录，设置之后对于桶的访问会被记录成日志，日志存储在OBS上您指定的目标桶中。

出于分析或审计等目的，用户可以开启日志记录功能。通过访问日志记录，桶的拥有者可以深入分析访问该桶的用户请求性质、类型或趋势。

当用户开启一个桶的日志记录功能后，OBS会自动对这个桶的访问请求记录日志，并生成日志文件写入用户指定的桶（即目标桶）中。

日志文件存放位置需要在开启桶日志功能时指定，可以存放到您拥有的，且与开启日志功能的桶位于同一区域的任一存储桶，当然也包括开启日志功能的桶本身。

更多关于访问日志的内容请参考**日志记录**。

14.2 开启桶日志

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.SetBucketLogging开启桶日志功能。

须知

日志目标桶与源桶必须在同一个区域（region）。

说明

如果桶的存储类型为低频访问存储或归档存储，则不能作为日志目标桶。

Agency字段为目标桶owner通过统一身份认证服务创建的对OBS服务的委托的名称，创建委托可参考统一身份认证服务委托相关章节。

开启桶日志

以下代码展示了如何开启桶日志：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
try
{
    // 设置源桶logging
    SetBucketLoggingRequest putrequest = new SetBucketLoggingRequest();
    putrequest.BucketName = "bucketname";
    putrequest.Configuration = new LoggingConfiguration();
    putrequest.Configuration.TargetBucketName = "targetbucketname";
    putrequest.Configuration.TargetPrefix = "access-log.";
    putrequest.Configuration.Agency= "your agency";
    SetBucketLoggingResponse putresponse = client.SetBucketLogging(putrequest);
    Console.WriteLine("Set bucket logging response: {0}", putresponse.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

14.3 查看桶日志配置

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.GetBucketLogging查看桶日志配置。

本示例用于查看桶名为“bucketname”的日志配置。

代码示例如下所示:

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 查看桶日志
try
{
    GetBucketLoggingRequest request = new GetBucketLoggingRequest
    {
        BucketName = "bucketname",
    };
    GetBucketLoggingResponse response = client.GetBucketLogging(request);
    Console.WriteLine("TargetBucketName is : " + response.Configuration.TargetBucketName);
    Console.WriteLine("TargetPrefix is : " + response.Configuration.TargetPrefix);
    Console.WriteLine("Get bucket logging response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- 查看桶日志过程中返回的错误码含义、问题原因及处理措施可参考**OBS服务端错误码**。

14.4 关闭桶日志

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

关闭桶日志功能实际上就是调用ObsClient.SetBucketLogging将日志配置清空。

本示例用于关闭桶名为“bucketname”的日志配置。

代码示例如下所示:

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
```

```
try
{
    SetBucketLoggingRequest putrequest = new SetBucketLoggingRequest();
    putrequest.BucketName = "bucketname";//源桶
    putrequest.Configuration = new LoggingConfiguration();
    SetBucketLoggingResponse putresponse = client.SetBucketLogging(putrequest);
    Console.WriteLine("Delete bucket logging response: {0}", putresponse.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- 关闭桶日志过程中返回的错误码含义、问题原因及处理措施可参考**OBS服务端错误码**。

15 静态网站托管

15.1 静态网站托管简介

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

静态网站通常仅包含静态网页，以及可能包含部分可在客户端运行的脚本，如JavaScript、Flash等。相比之下，动态网站则依赖于服务器端处理脚本，包括PHP、JSP或ASP.Net等。

您可以将静态网站文件上传至OBS的桶中作为对象，并对这些对象赋予公共读权限，然后将该桶配置成静态网站托管模式，以实现在OBS上托管静态网站的目的。

第三方用户在访问您网站的时候，实际上是在访问OBS的桶中的对象。

在使用静态网站托管功能时，OBS还支持配置请求重定向，通过重定向配置您可以将特定的请求或所有请求实施重定向。

更多关于静态网站托管的内容请参考**静态网站托管**。

15.2 网站文件托管

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可通过以下步骤实现网站文件托管：

步骤1 将网站文件上传至OBS的桶中，并设置对象MIME类型。

步骤2 设置对象ACL为公共读。

步骤3 通过浏览器访问对象。

----结束

以下代码展示了如何实现网站文件托管：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
try
{
    //设置对象MIME类型
    PutObjectRequest request = new PutObjectRequest
    {
        BucketName = "bucketname",
        ObjectKey = "test.html",
        FilePath = "localfile.html",//上传的本地文件路径，需要指定到具体的文件名
        CannedAcl = CannedAclEnum.PublicRead,//设置对象ACL为公共读
        ContentType = "text/html",
    };
    //上传对象
    PutObjectResponse response = client.PutObject(request);
    Console.WriteLine("put object response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

上例中您可以使用https://bucketname.your-endpoint/test.html在浏览器直接访问托管的文件。

15.3 设置托管配置

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.SetBucketWebsite设置桶的托管配置。

配置默认主页和错误页面

以下代码展示了如何配置默认主页和错误页面：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
```



```
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
//配置默认主页和错误页面
try
{
    SetBucketWebsiteRequest request = new SetBucketWebsiteRequest();
    request.BucketName = "bucketname";
    request.Configuration = new WebsiteConfiguration();
    //配置默认主页
    request.Configuration.IndexDocument = "index.html";
    //配置错误页面
    request.Configuration.ErrorDocument = "error.html";
    SetBucketWebsiteResponse response = client.SetBucketWebsiteConfiguration(request);
    Console.WriteLine("Set bucket website response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

配置重定向规则

以下代码展示了如何配置重定向规则：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 配置重定向规则
try
{
    SetBucketWebsiteRequest request = new SetBucketWebsiteRequest();
    request.BucketName = "bucketname";
    request.Configuration = new WebsiteConfiguration(); //配置默认主页
    request.Configuration.IndexDocument= "index.html";
    //配置错误页面
    request.Configuration.ErrorDocument = "error.html";
    RoutingRule routingRule = new RoutingRule();
    routingRule.Redirect = new Redirect();
    routingRule.Redirect.HostName = "www.example.com";
    routingRule.Redirect.HttpRedirectCode = "305";
    routingRule.Redirect.Protocol = ProtocolEnum.Http;
    routingRule.Redirect.ReplaceKeyPrefixWith = "replacekeyprefix";
    routingRule.Condition = new Condition();
    routingRule.Condition.HttpErrorCodeReturnedEquals = "404";
    routingRule.Condition.KeyPrefixEquals = "keyprefix";
    request.Configuration.RoutingRules.Add(routingRule);
    SetBucketWebsiteResponse response = client.SetBucketWebsiteConfiguration(request);
```



```
    Console.WriteLine("Set bucket website response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

配置所有请求重定向

以下代码展示了如何配置所有请求重定向：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 配置所有请求重定向
try
{
    SetBucketWebsiteRequest request = new SetBucketWebsiteRequest();
    request.BucketName = "bucketname";
    request.Configuration = new WebsiteConfiguration();
    request.Configuration.RedirectAllRequestsTo = new RedirectBasic();
    request.Configuration.RedirectAllRequestsTo.HostName = "www.example.com";
    request.Configuration.RedirectAllRequestsTo.Protocol = ProtocolEnum.Https;
    SetBucketWebsiteResponse response = client.SetBucketWebsiteConfiguration(request);
    Console.WriteLine("Set bucket website response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

15.4 查看托管配置

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.GetBucketWebsite查看桶的托管配置。

本示例用于查看桶名为“bucketname”的托管配置。

代码示例如下所示:

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境
```



```
变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID", EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey", EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 查看托管配置
try
{
    GetBucketWebsiteRequest request = new GetBucketWebsiteRequest();
    request.BucketName = "bucketname";
    GetBucketWebsiteResponse response = client.GetBucketWebsite(request);
    Console.WriteLine("GetBucketWebsite website configuration error document: {0}", response.Configuration.ErrorDocument);
    Console.WriteLine("GetBucketWebsite website configuration index document: {0}", response.Configuration.IndexDocument);
    Console.WriteLine("Get bucket website response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- 查看桶托管配置过程中返回的错误码含义、问题原因及处理措施可参考**OBS服务端错误码**。

15.5 清除托管配置

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.DeleteBucketWebsite清除桶的托管配置。

本示例用于清除桶名为“bucketname”的托管配置。

代码示例如下所示:

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID", EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey", EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 清除托管配置
try
{
    DeleteBucketWebsiteRequest request = new DeleteBucketWebsiteRequest();
    request.BucketName = "bucketname";
```

```
    DeleteBucketWebsiteResponse response = client.DeleteBucketWebsite(request);
    Console.WriteLine("Delete bucket website response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- 清除桶托管配置过程中返回的错误码含义、问题原因及处理措施可参考**OBS服务端错误码**。

16 标签管理

16.1 标签简介

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

标签用于标识OBS中的桶，以此来达到对OBS中的桶进行分类的目的。

16.2 设置桶标签

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.SetBucketTagging设置桶标签。以下代码展示了如何设置桶标签：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 设置桶标签
```

```
try
{
    SetBucketTaggingRequest request = new SetBucketTaggingRequest();
    request.BucketName = "bucketname";
    Tag tag1 = new Tag();
    tag1.Key = "tag1";
    tag1.Value = "value1";
    Tag tag2 = new Tag();
    tag2.Key = "tag2";
    tag2.Value = "value2";
    request.Tags.Add(tag2);
    request.Tags.Add(tag1);
    SetBucketTaggingResponse response = client.SetBucketTagging(request);
    Console.WriteLine("Set bucket tag response: {0}", response.StatusCode);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

说明

- 每个桶支持最多10个标签。
- 标签的Key和Value支持Unicode。

16.3 查看桶标签

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.GetBucketTagging查看桶标签。以下代码展示了如何查看桶标签：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 获取桶标签
try
{
    GetBucketTaggingRequest request = new GetBucketTaggingRequest
    {
        BucketName = "bucketname",
    };
    GetBucketTaggingResponse response = client.GetBucketTagging(request);
    foreach (Tag tag in response.Tags)
    {
        Console.WriteLine("Get bucket Tagging response Key: {0}" + tag.Key);
        Console.WriteLine("Get bucket Tagging response Value:{0} " + tag.Value);
```



```
    }
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

16.4 删除桶标签

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.DeleteBucketTagging删除桶标签。以下代码展示了如何删除桶标签：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 删除桶标签
try
{
    DeleteBucketTaggingRequest request = new DeleteBucketTaggingRequest
    {
        BucketName = "bucketname",
    };
    DeleteBucketTaggingResponse response = client.DeleteBucketTagging(request);
    Console.WriteLine("Delete bucket tag response: {0}", response.StatusCode);
}

catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

17 服务端加密

17.1 服务端加密简介

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

OBS支持服务端加密功能，使对象加密的行为在OBS服务端进行。

更多关于服务端加密的内容请参考**服务端加密**。

17.2 加密说明

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

OBS .NET SDK支持服务端加密的接口见下表：

OBS .NET SDK接口方法	描述	支持加密类型
ObsClient.PutObject	上传对象时设置加密算法、密钥，对对象启用服务端加密。	SSE-KMS SSE-C
ObsClient.GetObject	下载对象时设置解密算法、密钥，用于解密对象。	SSE-C

OBS .NET SDK接口方法	描述	支持加密类型
ObsClient.CopyObject	1. 复制对象时设置源对象的解密算法、密钥，用于解密源对象。 2. 复制对象时设置目标对象的加密算法、密钥，对目标对象启用加密算法。	SSE-KMS SSE-C
ObsClient.GetObjectMetadata	获取对象元数据时设置解密算法、密钥，用于解密对象。	SSE-C
ObsClient.InitiateMultipartUpload	初始化分段上传任务时设置加密算法、密钥，对分段上传任务最终生成的对象启用服务端加密。	SSE-KMS SSE-C
ObsClient.UploadPart	上传段时设置加密算法、密钥，对分段数据启用服务端加密。	SSE-C
ObsClient.CopyPart	1. 复制段时设置源对象的解密算法、密钥，用于解密源对象。 2. 复制段时设置目标段的加密算法、密钥，对目标段启用加密算法。	SSE-C

17.3 加密示例

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

上传对象加密

以下代码展示了在上传对象时使用服务端加密功能：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 生成一个加密密钥
System.Security.Cryptography.Aes aesEncryption = System.Security.Cryptography.Aes.Create();
aesEncryption.KeySize = 256;
aesEncryption.GenerateKey();
string customerkey = Convert.ToBase64String(aesEncryption.Key);
```

```
// 请根据实际情况配置本地待加密上传的文件
string filePathKms = "D:\\test\\testSseC.zip";
string filePathSseC = "D:\\test\\testSseC.zip";
// 上传对象
try
{
    // 上传时以SSE-KMS算法加密对象
    SseKmsHeader kms = new SseKmsHeader();
    kms.Algorithm = SseKmsAlgorithmEnum.Kms;
    PutObjectRequest request1 = new PutObjectRequest
    {
        BucketName = "bucketname",
        ObjectKey = "objectname1",
        FilePath = filePathKms,
        SseHeader = kms,
    };
    client.PutObject(request1);
    // 上传时以SSE-C算法加密对象
    PutObjectRequest request2 = new PutObjectRequest
    {
        BucketName = "bucketname",
        ObjectKey = "objectname2",
        FilePath = filePathSseC,
        SseHeader = new SseCHeader()
        {
            Algorithm = SseCAlgorithmEnum.Aes256,
            KeyBase64 = customerkey
        }
    };
    client.PutObject(request2);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

下载对象解密

以下代码展示了在下载对象时使用服务端解密功能：

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 下载对象
try
{
    // 下载时以SSE-C算法解密对象
    GetObjectRequest request = new GetObjectRequest
    {
        BucketName = "bucketname",
        ObjectKey = "objectname2",
        // 此处的密钥必须和上传对象加密时使用的密钥一致
        SseCHeader = new SseCHeader()
        {
            Algorithm = SseCAlgorithmEnum.Aes256,
            KeyBase64 = "customerkey"
        }
```



```
    };
    client.GetObject(request);
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```


18 异常处理

18.1 OBS 服务端错误码

在向OBS服务端发出请求后，如果遇到错误，会在响应中包含响应的错误码描述错误信息。详细的错误码及其对应的描述和HTTP状态码见下表：

错误码	描述	HTTP状态码
AccessDenied	拒绝访问。	403 Forbidden
AccessForbidden	权限不足。	403 Forbidden
AccountProblem	用户的账户出现异常（过期、冻结等），不能成功地完成操作。	403 Forbidden
AllAccessDisabled	用户无权限执行某操作。	403 Forbidden
AmbiguousGrantByEmailAddress	用户提供的Email地址关联的账户超过了1个。	400 Bad Request
BadDigest	客户端指定的对象内容的MD5值与系统接收到的内容MD5值不一致。	400 Bad Request
BadDomainName	域名不合法。	400 Bad Request
BadRequest	请求参数不合法。	400 Bad Request
BucketAlreadyExists	请求的桶名已经存在。桶的命名空间是系统中所有用户共用的，选择一个不同的桶名再重试一次。	409 Conflict
BucketAlreadyOwnedByYou	发起该请求的用户已经创建过了这个名字的桶，并拥有这个桶。	409 Conflict

错误码	描述	HTTP状态码
BucketNotEmpty	用户尝试删除的桶不为空。	409 Conflict
CredentialsNotSupported	该请求不支持证书验证。	400 Bad Request
CustomDomainAreadyExist	配置了已存在的域。	400 Bad Request
CustomDomainNotExist	操作的域不存在。	400 Bad Request
DeregisterUserId	用户已经注销。	403 Forbidden
EntityTooSmall	用户试图上传的对象大小小于系统允许的最小大小。	400 Bad Request
EntityTooLarge	用户试图上传的对象大小超过了系统允许的最大大小。	400 Bad Request
FrozenUserId	用户被冻结。	403 Forbidden
IllegalVersioningConfigurationException	请求中的版本配置无效。	400 Bad Request
IllegalLocationConstraintException	配置了与所在Region不匹配的区域限制。	400 Bad Request
InArrearOrInsufficientBalance	用户欠费或余额不足而没有权限进行某种操作。	403 Forbidden
IncompleteBody	请求体不完整。	400 Bad Request
IncorrectNumberOfFilesInPostRequest	每个POST请求都需要带一个上传的文件。	400 Bad Request
InlineDataTooLarge	Inline Data超过了允许的最大长度。	400 Bad Request
InsufficientStorageSpace	存储空间不足。	403 Forbidden
InternalError	系统遇到内部错误，请重试。	500 Internal Server Error
InvalidAccessKeyId	系统记录中不存在客户提供的Access Key Id。	403 Forbidden
InvalidAddressingHeader	用户必须指定匿名角色。	N/A
InvalidArgument	无效的参数。	400 Bad Request
InvalidBucketName	请求中指定的桶名无效。	400 Bad Request
InvalidBucket	请求访问的桶已不存在。	400 Bad Request

错误码	描述	HTTP状态码
InvalidBucketState	无效的桶状态。	409 Conflict
InvalidBucketStoragePolicy	修改桶策略时，提供的新策略不合法。	400 Bad Request
InvalidDigest	HTTP头中指定的Content-MD5值无效。	400 Bad Request
InvalidEncryptionAlgorithmError	错误的加密算法。	400 Bad Request
InvalidLocationConstraint	创建桶时，指定的location不合法。	400 Bad Request
InvalidPart	一个或多个指定的段无法找到。这些段可能没有上传，或者指定的entity tag与段的entity tag不一致。	400 Bad Request
InvalidPartOrder	段列表的顺序不是升序，段列表必须按段号升序排列。	400 Bad Request
InvalidPayer	所有对这个对象的访问已经无效了。	403 Forbidden
InvalidPolicyDocument	表单中的内容与策略文档中指定的条件不一致。	400 Bad Request
InvalidRange	请求的range不可获得。	416 Client Requested Range Not Satisfiable
InvalidRedirectLocation	无效的重定向地址。	400 Bad Request
InvalidRequest	无效请求。	400 Bad Request
InvalidRequestBody	POST请求体无效。	400 Bad Request
InvalidSecurity	提供的安全证书无效。	403 Forbidden
InvalidStorageClass	用户指定的Storage Class无效。	400 Bad Request
InvalidTargetBucketForLogging	delivery group对目标桶无ACL权限。	400 Bad Request
InvalidURI	无法解析指定的URI。	400 Bad Request
KeyTooLong	提供的Key过长。	400 Bad Request

错误码	描述	HTTP状态码
MalformedACLError	提供的XML格式错误，或者不符合要求的格式。	400 Bad Request
MalformedError	请求中携带的XML格式不正确。	400 Bad Request
MalformedLoggingStatus	Logging的XML格式不正确。	400 Bad Request
MalformedPolicy	Bucket policy检查不通过。	400 Bad Request
MalformedPOSTRequest	POST请求的请求体不是结构化良好的多段或形式化数据。	400 Bad Request
MalformedQuotaError	Quota的XML格式不正确。	400 Bad Request
MalformedXML	当用户发送了一个配置项的错误格式的XML会出现这样的错误。错误消息是：“The XML you provided was not well-formed or did not validate against our published schema.”。	400 Bad Request
MaxMessageLengthExceeded	请求消息过长。	400 Bad Request
MaxPostPreDataLengthExceededError	在上传文件前面的POST请求域过大。	400 Bad Request
MetadataTooLarge	元数据消息头超过了允许的最大元数据大小。	400 Bad Request
MethodNotAllowed	指定的方法不允许操作在请求的资源上。 对应返回的Message为：Specified method is not supported.	405 Method Not Allowed
MissingContentLength	必须要提供HTTP消息头中的Content-Length字段。	411 Length Required
MissingRegion	请求中缺少Region信息，且系统无默认Region。	400 Bad Request
MissingRequestBodyError	当用户发送一个空的XML文档作为请求时会发生。错误消息是：“Request body is empty.”。	400 Bad Request

错误码	描述	HTTP状态码
MissingRequiredHeader	请求中缺少必要的头域。	400 Bad Request
MissingSecurityHeader	请求缺少一个必须的头。	400 Bad Request
NoSuchBucket	指定的桶不存在。	404 Not Found
NoSuchBucketPolicy	桶policy不存在。	404 Not Found
NoSuchCORSConfiguration	CORS配置不存在。	404 Not Found
NoSuchCustomDomain	请求的用户域不存在。	404 Not Found
NoSuchKey	指定的Key不存在。	404 Not Found
NoSuchLifecycleConfiguration	请求的LifeCycle不存在。	404 Not Found
NoSuchPolicy	给定的policy名字不存在。	404 Not Found
NoSuchUpload	指定的多段上传不存在。Upload ID不存在，或者多段上传已经终止或完成。	404 Not Found
NoSuchVersion	请求中指定的version ID与现存的所有版本都不匹配。	404 Not Found
NoSuchWebsiteConfiguration	请求的Website不存在。	404 Not Found
NotImplemented	用户提供的消息头功能上还没有实现。	501 Not Implemented
NotSignedUp	账户未在系统中注册，必须先在系统中注册了才能使用该账户。	403 Forbidden
OperationAborted	另外一个冲突的操作当前正作用在这个资源上，请重试。	409 Conflict
PermanentRedirect	尝试访问的桶必须使用指定的节点，请将以后的请求发送到这个节点。	301 Moved Permanently
PreconditionFailed	用户指定的先决条件中至少有一项没有包含。	412 Precondition Failed
Redirect	临时重定向。	307 Moved Temporarily
RequestIsNotMultiPartContent	桶POST必须是闭式的多段/表单数据。	400 Bad Request

错误码	描述	HTTP状态码
RequestTimeout	用户与Server之间的socket连接在超时时间内没有进行读写操作。	400 Bad Request
RequestTimeTooSkewed	请求的时间与服务器的时间相差太大。	403 Forbidden
RequestTorrentOfBucketError	不允许请求桶的torrent文件。	400 Bad Request
ServiceNotImplemented	请求的方法服务端没有实现。	501 Not Implemented
ServiceNotSupported	请求的方法服务端不支持。	409 Conflict
ServiceUnavailable	服务器过载或者内部错误异常。	503 Service Unavailable
SignatureDoesNotMatch	请求中带的签名与系统计算得到的签名不一致。检查您的访问密钥（AK和SK）和签名计算方法。	403 Forbidden
SlowDown	请降低请求频率。	503 Service Unavailable
System Capacity Not enough	系统空间不足异常。	403 Forbidden
TooManyCustomDomains	配置了过多的用户域。	400 Bad Request
TemporaryRedirect	当DNS更新时，请求将被重定向到桶。	307 Moved Temporarily
TooManyBuckets	用户拥有的桶的数量达到了系统的上限，并且请求试图创建一个新桶。	400 Bad Request
TooManyObjectCopied	用户单个对象被拷贝的数量超过系统上限。	400 Bad Request
TooManyWrongSignature	因高频错误请求被拒绝服务。	400 Bad Request
UnexpectedContent	该请求不支持带内容字段。	400 Bad Request
UnresolvableGrantByEmailAddress	用户提供的Email与记录中任何账户的都不匹配。	400 Bad Request
UserKeyMustBeSpecified	请求中缺少用户的AK信息。	400 Bad Request
WebsiteRedirect	Website请求缺少bucketName。	301 Moved Permanently

错误码	描述	HTTP状态码
KMS.DisabledException	SSE-KMS加密方式下，主密钥被禁用。	400 Bad Request
KMS.NotFoundException	SSE-KMS加密方式下，主密钥不存在。	400 Bad Request
RestoreAlreadyInProgress	对象正在恢复，请求冲突。	409 Conflict
ObjectHasAlreadyRestored	已经恢复的对象，禁止缩短恢复保存时间。	409 Conflict
InvalidObjectState	恢复对象不是归档存储对象。	403 Forbidden
InvalidTagError	配置桶标签时，提供了无效的Tag。	400 Bad Request
NoSuchTagSet	指定的桶没有设置标签。	404 Not Found

18.2 日志分析

日志路径

OBS .NET SDK生成的日志文件路径是通过Log4Net.config配置文件指定的，一般将该配置文件放于工程编译生成的可执行文件所在目录。

日志级别

当系统出现问题需要定位且当前的日志无法满足要求时，可以通过修改日志的级别来获取更多的信息。其中DEBUG日志信息最丰富，ERROR日志信息最少。

具体说明如下：

- DEBUG：调试级别。
- INFO：信息级别。
- WARN：告警级别。
- ERROR：错误级别。

分析方法

分析定位问题或查看运行状态时，可根据日期查找到相应的日志文件，再通过日志文件的日志记录进行分析。

分析的具体步骤如下：

步骤1 查找日志信息。

根据错误发生时间及操作的相关信息，日志文件目录下查找相关的接口日志，日志示例如下。


```
2018-05-23 21:55:02,103 [9] INFO  - ListObjectsRequest begin.
2018-05-23 21:55:02,526 [9] INFO  - Send http request end, cost 385 ms
2018-05-23 21:55:02,536 [9] ERROR - Rethrowing as a ObsException error in PerformRequest
Request error, StatusCode:404, ErrorCode:NoSuchBucket, ErrorMessage:The specified bucket does not exist,
RequestId:0403000001638D4819383F2D4A2B2C50, HostId:N8OMsHew7O/
LMHua8qpm49geWphVJI6l2mnnzUIYwQwHAuzJw/kmV+O4iIcf0GRR
2018-05-23 21:55:02,548 [9] ERROR - ListObjectsRequest exception code: NoSuchBucket, with message:
Request error
2018-05-23 21:55:02,553 [9] INFO  - ListObjectsRequest end, cost 449 ms
```

步骤2 根据错误日志信息分析错误原因。

例如，从日志文件中获取到错误码"NoSuchBucket"，通过查看对照**OBS服务端错误码**，得知实际的错误信息为“指定bucket不存在”。

----结束

18.3 SDK 自定义异常

SDK自定义异常（ObsException）是由ObsClient统一抛出的异常。通常是OBS服务端错误，包含**OBS错误码**、错误信息等，便于用户定位问题，并做出适当的处理。

ObsException通常包含以下错误信息：

- ObsException.StatusCode：HTTP状态码。
- ObsException.ErrorCode：OBS服务端错误码。
- ObsException.ErrorMessage：OBS服务端错误描述。
- ObsException.RequestId：OBS服务端返回的请求ID。
- ObsException.HostId：请求的服务端ID。

18.4 SDK 公共响应头

调用ObsClient类的相关接口成功后，返回结果均是公共响应头（ObsWebServiceResponse）的子类。

公共响应头类中包含了HTTP/HTTPS的响应头等信息。

此示例用于创建桶时，从公共响应头中获取request-id。

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);
// 创建桶
try
{
    CreateBucketRequest request = new CreateBucketRequest()
    {
        BucketName = "bucketname",
    };
```

```
    ObsWebServiceResponse response = client.CreateBucket(request);
    // 从公共响应头中获取request-id
    Console.WriteLine("RequestId: {0}", response.RequestId);
    foreach(KeyValuePair<string,string> entry in res.Headers)
    {
        Console.WriteLine("{0}:{1}", entry.Key, entry.Value);
    }
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```


19 常见问题

19.1 如何解决进程偶现卡死的问题?

问题现象：在调用.NET SDK方法时，进程偶现卡死的情况。

解决办法：如果遇到该问题，可以将方法调用加上using方式。

以下示例展示使用using方式调用接口。

```
// 初始化配置参数
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量AccessKeyID和SecretAccessKey。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// 创建ObsClient实例
ObsClient client = new ObsClient(accessKey, secretKey, config);

try
{
    GetObjectMetadataRequest request = new GetObjectMetadataRequest();
    // 指定存储桶名称
    request.BucketName = "bucketname";
    // 指定对象，此处以 example/objectname 为例
    request.ObjectKey = "example/objectname";
    // 获取对象元数据
    using (GetObjectMetadataResponse response = client.GetObjectMetadata(request)) {
        Console.WriteLine("Get object metadata response: {0}", response.StatusCode);
        // 获取对象的ETag值
        Console.WriteLine("Object etag {0}:  ", response.ETag);
        // 获取对象的版本号
        Console.WriteLine("Object versionId {0}:  ", response.VersionId);
        // 获取对象数据的长度，单位是字节
        Console.WriteLine("Object contentLength {0}:  ", response.ContentLength);
    }
}
catch (ObsException ex)
{
    Console.WriteLine("Message: {0}", ex.Message);
}
```