

Object Storage Service

.NET SDK Developer Guide

Issue 01
Date 2025-11-12



Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2025. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Cloud Computing Technologies Co., Ltd.

Trademarks and Permissions



HUAWEI and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Contents

1 Before You Start	1
2 SDK Download Links	3
3 Example Programs	4
4 Quick Start	5
4.1 Setting Up an OBS Environment	5
4.2 Installing the SDK	8
4.3 Obtaining Endpoints	9
4.4 Initializing an Instance of ObsClient	9
4.5 Creating a Bucket	10
4.6 Uploading an Object	11
4.7 Downloading an Object	11
4.8 Listing Objects	12
4.9 Deleting an Object	12
4.10 General Examples of ObsClient	13
5 Initialization	15
5.1 Configuring the AK/SK	15
5.2 Creating an Instance of ObsClient	15
5.3 Configuring an Instance of ObsClient	17
5.4 Configuring SDK Logging	18
6 Bucket Management	20
6.1 Creating a Bucket	20
6.2 Listing Buckets	22
6.3 Deleting a Bucket	23
6.4 Identifying Whether a Bucket Exists	24
6.5 Obtaining Bucket Metadata	24
6.6 Managing Bucket ACLs	25
6.7 Management Bucket Policies	31
6.8 Obtaining a Bucket Location	33
6.9 Obtaining Storage Information About a Bucket	34
6.10 Setting or Obtaining a Bucket Quota	35
6.11 Setting or Obtaining the Storage Class of a Bucket	36

7 Object Upload	39
7.1 Object Upload Overview	39
7.2 Performing a Streaming Upload	40
7.3 Performing a File-Based Upload	41
7.4 Performing an Asynchronous Upload	42
7.5 Obtaining Upload Progress	43
7.6 Creating a Folder	44
7.7 Setting Object Properties	45
7.8 Performing a Multipart Upload	49
7.9 Configuring Lifecycle Management	56
7.10 Performing an Appendable Upload	57
7.11 Performing a Multipart Copy	58
7.12 Performing a Resumable Upload	59
8 Object Download	63
8.1 Object Download Overview	63
8.2 Performing a Streaming Download	63
8.3 Performing a Partial Download	64
8.4 Performing an Asynchronous Download	65
8.5 Obtaining Download Progress	67
8.6 Performing a Conditioned Download	68
8.7 Rewriting Response Headers	69
8.8 Obtaining Custom Metadata	71
8.9 Downloading an Archive Object	72
8.10 Performing a Resumable Download	73
8.11 Processing an Image	76
9 Object Management	78
9.1 Obtaining Object Properties	78
9.2 Managing Object ACLs	79
9.3 Listing Objects	82
9.4 Deleting Objects	87
9.5 Copying an Object	89
9.6 HEAD Object	92
10 Temporarily Authorized Access	94
10.1 Using a Temporary URL for Authorized Access	94
11 Versioning Management	105
11.1 Versioning Overview	105
11.2 Setting Versioning Status for a Bucket	105
11.3 Viewing Versioning Status of a Bucket	108
11.4 Obtaining a Versioning Object	108
11.5 Copying a Versioning Object	109
11.6 Restoring a Specific Archive Object Version	110

11.7 Listing Versioning Objects.....	111
11.8 Setting or Obtaining a Versioning Object ACL.....	116
11.9 Deleting Versioning Objects.....	118
12 Lifecycle Management.....	120
12.1 Lifecycle Management Overview.....	120
12.2 Setting Lifecycle Rules.....	121
12.3 Viewing Lifecycle Rules.....	123
12.4 Deleting Lifecycle Rules.....	124
13 CORS.....	125
13.1 CORS Overview.....	125
13.2 Setting CORS Rules.....	125
13.3 Viewing CORS Rules.....	127
13.4 Deleting CORS Rules.....	128
14 Access Logging.....	129
14.1 Logging Overview.....	129
14.2 Enabling Bucket Logging.....	129
14.3 Viewing Bucket Logging.....	130
14.4 Disabling Bucket Logging.....	131
15 Static Website Hosting.....	133
15.1 Static Website Hosting Overview.....	133
15.2 Website File Hosting.....	133
15.3 Setting Website Hosting.....	134
15.4 Viewing Website Hosting Settings.....	137
15.5 Deleting Website Hosting Settings.....	138
16 Tag Management.....	139
16.1 Tagging Overview.....	139
16.2 Setting Bucket Tags.....	139
16.3 Viewing Bucket Tags.....	140
16.4 Deleting Bucket Tags.....	141
17 Server-Side Encryption.....	142
17.1 Server-Side Encryption Overview.....	142
17.2 Encryption Description.....	142
17.3 Example of Encryption.....	143
18 Troubleshooting.....	146
18.1 OBS Server-Side Error Codes.....	146
18.2 Log Analysis.....	154
18.3 SDK Custom Exceptions.....	155
18.4 SDK Common Response Headers.....	155
19 FAQ.....	156

19.1 What Do I Do If the Process Hangs Occasionally?..... 156

1 Before You Start

This section describes version updates and compatibility and important notes about Object Storage Service (OBS) SDK for .NET.

Version Updates

[Table 1-1](#) describes the version updates and compatibility of OBS SDK for .NET.

Table 1-1 Version updates and compatibility of OBS SDK for .NET

Version	Update	Description	Compatible or Not
v3.22.11.4	-	Provided the .NET Framework version in NuGet.	Yes
v3.22.11.3	-	Provided the .NET Core version in NuGet.	Yes
v3.22.3	New feature	Supported Content-Disposition standard metadata interfaces.	Yes
Earlier than v3.22.3	-	End of matching. You are advised to upgrade the version.	-

Compatibility

- For more information about version updates, see [ChangeLog](#).
- Recommended .NET versions: .NET Framework 3.5, 4.0, or 4.5; .NET Core 2.0 or 3.1; .NET 6 or 7
- The namespaces used in earlier versions (2.x.x) are reorganized and all public APIs are now in namespaces **OBS** and **OBS.Model**.
- API functions are redesigned and not compatible with those provided by earlier versions (2.x.x).

Important Notes

- Ensure that you are familiar with OBS basic concepts, such as [buckets](#), [objects](#), and [access keys \(AKs/SKs\)](#).
- You can see [General Examples of ObsClient](#) to learn how to call OBS .NET SDK APIs in a general manner.
- After an API is called using an instance of **ObsClient**, if no exception is thrown, the return value is valid. If an exception is thrown, the operation fails, and you can obtain the exception details from the instance of [ObsException](#).
- After an API is successfully called using an instance of **ObsClient**, an instance of [HeaderResponse](#) that contains response headers will be returned.

2 SDK Download Links

SDK Source Codes and API Documentation

- Latest version of OBS .NET SDK source code: [Download](#)
- Earlier versions of OBS .NET SDK: [Download](#)
- OBS .NET SDK API document: [OBS .NET SDK API Reference](#)

3 Example Programs

OBS .NET SDK provides abundant example programs for user reference and direct use. You can obtain example programs from folder **demo** in the OBS .NET SDK development package.

Example programs include:

Sample Code	Describes
BucketOperationsSample	How to use bucket-related APIs.
ObjectOperationsSample	How to use object-related APIs.
TemporarySignatureSample	How to use URLs for authorized access.

4 Quick Start

4.1 Setting Up an OBS Environment

Step 1 Sign up for a cloud service account.

Create an account to use OBS. If you already have one, use it instead.

1. Open a browser.
2. Visit the [Huawei Cloud official website](#).
3. In the upper right corner of the page, click **Register**.
4. Enter the registration information and click **Register**.

Step 2 Enable OBS.

Top up your account before you can use OBS.

1. Log in to the [management console](#).
2. Click **Billing & Costs** from the top menu bar. The **Billing Center** page is displayed.
3. Choose **Funds Management > Top Up**. The **Top Up** page is displayed.
4. Top up your account.
5. After the top-up is complete, close the dialog box and go back to the homepage.
6. Choose **Service List > Object Storage Service** to access OBS Console.

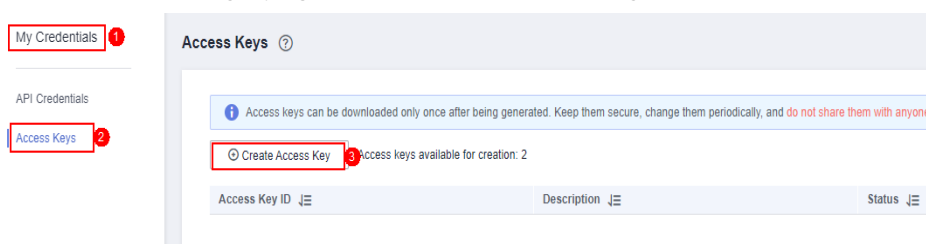
Step 3 Create access keys.

OBS employs access keys (AK and SK) for signature verification to ensure that only authorized accounts can access specified OBS resources. Detailed explanations of access keys are as follows:

- AK is short for Access Key ID. One AK maps to only one user but one user can have multiple AKs. OBS authenticates users by their AKs.
- SK is short for Secret Access Key, which is used to access OBS. You can generate authentication information based on SKs and request headers. An SK maps to an AK, and they group into a pair.

Access keys are permanent. There are also temporary security credentials (consisting of an AK/SK pair and a security token). Each user can create a maximum of two valid AK/SK pairs. Temporary security credentials can only be used to access OBS within the specified validity period. Once they expire, they must be requested again. For security purposes, you are advised to use temporary security credentials to access OBS. If you want to use permanent access keys, periodically update them.

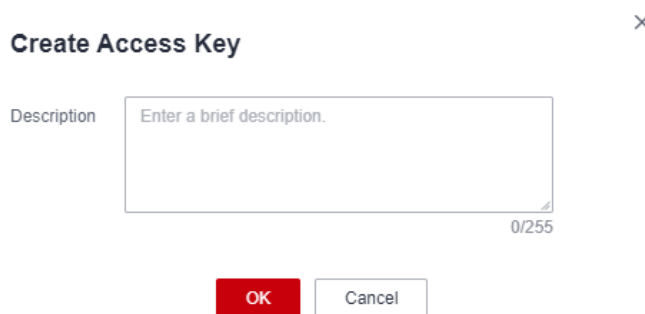
- To get permanent access keys, do as follows:
 - a. Log in to the [management console](#).
 - b. In the upper right corner, hover your cursor over the username and choose **My Credentials**.
 - c. On the **My Credentials** page, click **Access Keys** in the navigation pane.
 - d. On the **Access Keys** page, click **Create Access Key**.



NOTE

Each user can create a maximum of two valid AK/SK pairs.

- e. In the **Create Access Key** dialog box, enter a description (recommended), and click **OK**.



- f. (Optional) In the displayed **Identity Verification** dialog box, select a verification method, enter the verification code, and click **OK**.

Identity Verification

The administrator has enabled operation protection. For account security purposes, verify your identity to proceed with the operation. If you do not require operation protection for critical operations, request the administrator to disable it.

Verification Method SMS Email Virtual MFA device ?

Email .com [Change](#)

Verification Code

- g. In the displayed dialog box, click **Download** to save the access keys to your browser's default download path.

✓ Access key created successfully.

This is the only time you can download and view the access key. If the access key is lost, you must create a new one.

- h. Open the downloaded **credentials.csv** file to obtain the access keys (AK and SK).

NOTE

- In the **credentials.csv** file, the AK is the value in the **Access Key ID** column, and the SK is the one in the **Secret Access Key** column.
 - Keep the access keys properly to prevent information leakage. If you click **Cancel** in the download dialog box, the access keys will not be downloaded and cannot be downloaded later. You can create new access keys if required.
- To get temporary security credentials, refer to the following:
Temporary security credentials are issued by the system and are only valid for 15 minutes to 24 hours. They follow the principle of least privilege. When using temporary security credentials, you must use an AK/SK pair and a security token together.
To obtain them, see [Obtaining a Temporary AK/SK and a Security Token](#).

NOTICE

OBS is a global service. When obtaining temporary access keys, set the token scope to **domain** to apply the token to global services. Global services are not differentiated by any project or region.

----End

4.2 Installing the SDK

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

Method 1: Using the NuGet Package Manager in Visual Studio. The procedure is as follows:

- Step 1** Open Visual Studio and choose **Tools > NuGet Package Manager > Manage NuGet Packages for Solution**.
- Step 2** Search for **esdk** and in the results, select **HuaweiCloud.ESDK.OBS** if you want to install .NET Framework or **HuaweiCloud.ESDK.OBS.Core** if you want to install .NET Core.

----End

Method 2: Installing the OBS .NET SDK. The example here uses the latest version.

- Step 1** Download the OBS .NET SDK development package. For details, see [SDK Download Links](#).
- Step 2** Decompress the development package to obtain folder **demo** (sample code) and file **Log4Net.config** (log configuration file). In the **release** folder containing all SDK versions, select the latest version and decompress it. In the decompressed folder, you can obtain folder **en** (including files **log4net.dll**, the third-party log library file, and **esdk_obs_net.dll**, the SDK library file) and file **Log4Net.config** (log configuration file). Start Visual Studio, open the project solution, and open the **esdk_obs_net.sln** or **esdk_obs_net_core.sln** file in the directory based on the project framework. Set the solution to **Release** and generate a solution. You can find **esdk_obs_net.dll** and **log4net.dll** files in **bin/Release** in the decompression directory.
- Step 3** Start Visual Studio and choose **FILE > New > Project > Templates > Visual C# > Console Application** to create a Console Application project.
- Step 4** Right-click the new project and choose **References > Add Reference** and import file **esdk_obs_net.dll**.

----End

4.3 Obtaining Endpoints

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

- You can click [here](#) to view the endpoints and regions enabled for OBS.

NOTICE

The SDK allows you to pass endpoints with or without the protocol name. Suppose the endpoint you obtained is **your-endpoint**. The endpoint passed when initializing an instance of **ObsClient** can be **http://your-endpoint**, **https://your-endpoint**, or **your-endpoint**.

4.4 Initializing an Instance of ObsClient

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

Each time you want to send an HTTP/HTTPS request to OBS, you must create an instance of **ObsClient**. Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Use the instance to access OBS.
```

NOTE

For more information, see chapter "Initialization."

For details about logging configuration, see [Configuring SDK Logging](#).

4.5 Creating a Bucket

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

A bucket is a global namespace of OBS and is a data container. It functions as a root directory of a file system and can store objects. The following code shows how to create a bucket:

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

NOTE

- Bucket names are globally unique. Ensure that the bucket you create is named differently from any other bucket.
- A bucket name must comply with the following rules:
 - Contains 3 to 63 characters, chosen from lowercase letters, digits, hyphens (-), and periods (.), and starts with a digit or letter.
 - Cannot be an IP address or similar.
 - Cannot start or end with a hyphen (-) or period (.)
 - Cannot contain two consecutive periods (.), for example, **my..bucket**.
 - Cannot contain periods (.) and hyphens (-) adjacent to each other, for example, **my-.bucket** or **my.-bucket**.
- If you create buckets of the same name, no error will be reported and the bucket properties comply with those set in the first creation request.
- For more information, see [Creating a Bucket](#).

NOTICE

- During bucket creation, if the endpoint you use corresponds to the default region CN North-Beijing1 (cn-north-1), specifying a region is not a must. If the endpoint you use corresponds to any other region, except the default one, you must set the region to the one that the used endpoint corresponds to. For more information about the valid regions, see [Regions and Endpoints](#).
- You can specify a region for the bucket during bucket creation. For details, see [Creating a Bucket with Parameters Specified](#).

4.6 Uploading an Object

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

This example uploads string **Hello OBS** to bucket **bucketname** as object **objectname**.

The example code is as follows:

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

NOTE

- For more information, see [Object Upload Overview](#).

4.7 Downloading an Object

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

This example downloads object **objectname** from bucket **bucketname**.

The example code is as follows:

```
GetObjectRequest request = new GetObjectRequest()
{
    BucketName = "bucketname",
    ObjectKey = "objectname",
};
using (GetObjectResponse response = client.GetObject(request))
{
    //Save the object locally.
    string dest = "savepath";
    if (!File.Exists(dest))
    {
        response.WriteResponseStreamToFile(dest);
    }
}
```

NOTE

- For more information, see [Object Download Overview](#).

4.8 Listing Objects

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

After objects are uploaded, you may want to view the objects contained in a bucket. Sample code is as follows:

```
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);
}
```

NOTE

- You can call `ListObjectsResponse.ObsObjects` to obtain the descriptions of all objects.
- In the previous sample code, 1000 objects will be listed, by default.
- For more information, see [Listing Objects](#).

4.9 Deleting an Object

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

This example deletes object **objectname** from bucket **bucketname**.

The example code is as follows:

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

NOTE

- This example only deletes a single object. To delete objects in a batch, traverse objects and list to-be-deleted objects on your own.
- For details about deletion, see [Deleting Objects](#).

4.10 General Examples of ObsClient

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

When you call an API using an instance of **ObsClient**, if no exception is thrown, the return value is valid, and a sub-class instance of **ObsWebServiceResponse** (SDK common response headers) is returned. If any exception is thrown, the operation failed, and you can obtain the exception details from the returned instance of **ObsException**. **ObsClient** supports synchronous and asynchronous API callings. Examples are as follows:

Synchronous Call

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);

// Call the API for creating a bucket in synchronous mode.
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);
}
```

Asynchronous Call

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
```

```
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);

// Call the API for creating a bucket in asynchronous mode.
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 Initialization

5.1 Configuring the AK/SK

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

To use OBS, you need a valid pair of AK and SK for signature authentication.

For details, see [Setting Up an OBS Environment](#).

After obtaining the AK/SK pair, you can start initialization by taking these steps:

- [Creating an Instance of ObsClient](#)
- [Configuring an Instance of ObsClient](#)
- [Configuring SDK Logging](#)

5.2 Creating an Instance of ObsClient

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

ObsClient functions as the .NET client for accessing OBS. It offers callers a series of APIs for interaction with OBS and is used for managing and operating resources, such as buckets and objects, stored in OBS. To use OBS .NET SDK to send a request to OBS, you need to initialize an instance of **ObsClient** and modify the default configurations in **ObsConfig** based on needs.

- If you use the endpoint to create an instance of **ObsClient**, all parameters are in their default values and cannot be modified.

- Using permanent AK (access key ID)/SK(secret access key)

```
// Initialize configuration parameters.  
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them  
// in the configuration file or environment variables. In this example, the AK/SK are stored in  
// environment variables for identity authentication. Before running this example, configure  
// environment variables AccessKeyID and SecretAccessKey.  
// Obtain an AK/SK pair on the management console. For details, see https://  
support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.  
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",  
EnvironmentVariableTarget.Machine);  
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",  
EnvironmentVariableTarget.Machine);  
// Create an instance of ObsClient.  
ObsClient client = new ObsClient(accessKey, secretKey, "https://your-endpoint");  
// Use the instance to access OBS.
```

- Using temporary access credentials (AKs/SKs and security tokens)

```
// Initialize configuration parameters.  
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them  
// in the configuration file or environment variables. In this example, the AK/SK are stored in  
// environment variables for identity authentication. Before running this example, configure  
// environment variables AccessKeyID and SecretAccessKey.  
// Obtain an AK/SK pair on the management console. For details, see https://  
support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.  
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",  
EnvironmentVariableTarget.Machine);  
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",  
EnvironmentVariableTarget.Machine);  
string securityToken= "your_securityToken"  
// Create an instance of ObsClient.  
ObsClient client = new ObsClient(accessKey, secretKey,securityToken, "https://your-endpoint");  
// Use the instance to access OBS.
```

- If you use the **ObsConfig** configuration class to create an instance of **ObsClient**, you can set any parameters as needed during the creation. After the instance has been created, the parameters cannot be modified. For parameter details, see [Configuring an Instance of ObsClient](#)

```
// Create an instance of ObsConfig.  
ObsConfig config = new ObsConfig();  
config.Endpoint = "https://your-endpoint";  
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the  
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables  
// for identity authentication. Before running this example, configure environment variables AccessKeyID and  
// SecretAccessKey.  
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/  
intl/en-us/usermanual-ca/ca\_01\_0003.html.  
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",  
EnvironmentVariableTarget.Machine);  
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",  
EnvironmentVariableTarget.Machine);  
// Create an instance of ObsClient.  
ObsClient client = new ObsClient(accessKey, secretKey, config);  
// Use the instance to access OBS.
```

NOTE

- The project can contain one or more instances of **ObsClient**.
- **ObsClient** is thread-safe and can be simultaneously used by multiple threads.

5.3 Configuring an Instance of ObsClient

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

When you call the **ObsConfig** configuration class to create an instance of **ObsClient**, you can configure the agent, timeout duration, maximum allowed number of connections, and some other parameters listed in the following table.

Parameter	Description	Recommended Value
Endpoint	Endpoint for accessing OBS, which contains the protocol type, domain name (or IP address), and port number. For example, https://your-endpoint:443. For security purposes, you are advised to use HTTPS.	N/A
Timeout	Timeout duration for synchronous callings, in ms. The default value is -1 , which indicates no timeout limitations.	N/A
ReadWriteTimeout	Timeout duration for transmitting data at the Socket layer, in ms. The default value is 60000 .	[10000, 60000]
AsyncSocketTimeout	Timeout duration for asynchronous callings, in ms. The default value is -1 , which indicates no timeout limitations.	N/A
MaxIdleTime	Allowed connection idle time, in ms. If a connection exceeds the specified value, the connection will be closed. The default value is 30000 .	Default
ConnectionLimit	Maximum number of concurrently opened HTTP connections. The default value is 1000.	Default
MaxErrorRetry	Maximum number of retry attempts (caused by abnormal requests, 500 errors, and 503 errors). The default value is 3 .	[1, 5]

Parameter	Description	Recommended Value
ReceiveBufferSize	Size of the socket reception buffer. The default value is 8192 .	[8192, 65536]
SecurityProtocol-Type	Encryption protocol type when HTTPS is used.	N/A
ProxyHost	Host address of the proxy server.	N/A
ProxyPort	Port ID of the proxy server.	N/A
ProxyUserName	User name used for connecting to the proxy server.	N/A
ProxyPassword	Password used for connecting to the proxy server.	N/A
ProxyDomain	Domain to which the proxy belongs	N/A
ValidateCertificate	Whether to verify the server certificate. The default value is false .	N/A
BufferSize	Read or write cache size used for uploading an object to the socket stream, in bytes. The default value is 8192 .	Default
KeepAlive	Whether to use persistent connections to access OBS. The default value is true .	N/A

NOTE

- Parameters whose recommended value is **N/A** need to be set based on your needs.
- If the network is unstable, you are advised to set larger values for **Timeout**, **AsyncSocketTimeout**, and **ReadWriteTimeout**.
- If the value of **Endpoint** does not contain any protocol, HTTPS is used by default.
- For the sake of high DNS resolution performance and OBS reliability, you can set **Endpoint** only to the domain name of OBS, instead of the IP address.

5.4 Configuring SDK Logging

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

OBS .NET SDK provides the logging function, based on the Apache Log4net open library. You can add log configuration files to enable the logging function. The procedure is as follows:

- Step 1** Add a reference to **log4net.dll** in the project.
- Step 2** Copy configuration file **Log4Net.config** to **Debug** or **Release** under directory **bin** of the project, to ensure that the configuration file and the project's executable files are in the same directory.
- Step 3** Modify log levels in file **Log4Net.config** based on needs.

----End

 **NOTE**

- Without these operations, the logging function is in the disabled state and no logs will be generated.
- For details about SDK logs, see [Log Analysis](#).
- The save path of log files defaults to be that of the project's executable files and can be changed in **Log4Net.config**.

6 Bucket Management

6.1 Creating a Bucket

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call `ObsClient.CreateBucket` to create a bucket.

Creating a Bucket with Parameters Specified

When creating a bucket, you can specify the ACL, storage class, and location for the bucket. OBS provides three storage classes for buckets. For details, see [Setting or Obtaining the Storage Class of a Bucket](#). Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Create a bucket.
try
{
    CreateBucketRequest request = new CreateBucketRequest
    {
        BucketName = "bucketname",
        // Set the bucket location.
        Location = "bucketLocation",
        // Set the storage class to Archive.
        StorageClass = StorageClassEnum.Cold,
    }
}
```

```
// Set the ACL for the bucket to public read (the default state is private).
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);
}
```

NOTE

- Bucket names are globally unique. Ensure that the bucket you create is named differently from any other bucket.
- A bucket name must comply with the following rules:
 - Contains 3 to 63 characters, chosen from lowercase letters, digits, hyphens (-), and periods (.), and starts with a digit or letter.
 - Cannot be an IP address or similar.
 - Cannot start or end with a hyphen (-) or period (.)
 - Cannot contain two consecutive periods (.), for example, **my.bucket**.
 - Cannot contain periods (.) and hyphens (-) adjacent to each other, for example, **my-.bucket** or **my.-bucket**.
- If you create buckets of the same name in a region, no error will be reported and the bucket properties comply with those set in the first creation request.
- The bucket created in the previous example is of the default **ACL (private)**, in the OBS Standard storage class, and in the default region.

NOTICE

- During bucket creation, if the endpoint you use corresponds to the default region CN North-Beijing1 (cn-north-1), specifying a region is not a must. If the endpoint you use corresponds to any other region, except the default one, you must set the region to the one that the used endpoint corresponds to. For more information about the valid regions, see [Regions and Endpoints](#).
 - You can specify a region for the bucket during bucket creation. For details, see [Creating a Bucket with Parameters Specified](#).
-

Creating a Bucket Directly

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
```

```
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Create a bucket.
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 Listing Buckets

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call **ObsClient.ListBuckets** to list buckets. Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// List buckets.
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);
}
```

 NOTE

- Obtained bucket names are listed in the lexicographical order.
- Set **ListBucketsRequest.IsQueryLocation** to **true** and then you can query the bucket location when listing buckets.

6.3 Deleting a Bucket

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call **ObsClient.DeleteBucket** to delete a bucket. Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
//Delete a bucket.
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);
}
```

 NOTE

- Only empty buckets (without objects and part fragments) can be deleted.
- Bucket deletion is a non-idempotence operation and an error will be reported if the to-be-deleted bucket does not exist.

6.4 Identifying Whether a Bucket Exists

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call `ObsClient.HeadBucket` to identify whether a bucket exists.

This example checks whether bucket `bucketname` exists.

The example code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config)
// Identify whether a bucket exists.
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);
}
```

NOTE

If an exception is thrown and the returned HTTP status code is **404**, the bucket does not exist.

6.5 Obtaining Bucket Metadata

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call **ObsClient.GetBucketMetadata** to obtain the metadata of a bucket.

This example returns the metadata of bucket **bucketname**.

The example code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Obtain the bucket metadata.
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);
}
```

NOTE

- To handle the error codes possibly returned during the operation, see [OBS Server-Side Error Codes](#).

6.6 Managing Bucket ACLs

Access control lists (ACLs) allow resource owners to grant other accounts the permissions to access resources. By default, only the resource owner has full control over resources when a bucket or object is created. That is, the bucket creator has full control over the bucket, and the object uploader has full control over the object. Other accounts do not have the permissions to access resources. If resource owners want to grant other accounts the read and write permissions on resources, they can use ACLs. ACLs grant permissions to accounts. After an account is granted permissions, both the account and its IAM users can access the resources.

For more information, see [ACLs](#).

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

A bucket ACL can be configured in any of the following ways:

1. Specify a pre-defined ACL when creating a bucket.
2. Call `ObsClient.SetBucketAcl` to specify a pre-defined ACL.
3. Call `ObsClient.SetBucketAcl` to specify a user-defined ACL.

The following table lists the five permission types supported by OBS.

Permission	Description	Value in OBS .NET SDK
READ	A grantee with this permission for a bucket can obtain the list of objects in the bucket and the metadata of the bucket. A grantee with this permission for an object can obtain the object content and metadata.	PermissionEnum.Read
WRITE	A grantee with this permission for a bucket can upload, overwrite, and delete any object in the bucket. Such permission for an object is not applicable.	PermissionEnum.Write
READ_ACP	A grantee with this permission can obtain the ACL of a bucket or object. A bucket or object owner has this permission permanently.	PermissionEnum.ReadAcp
WRITE_ACP	A grantee with this permission can update the ACL of a bucket or object. A bucket or object owner has this permission permanently. A grantee with this permission can modify the access control policy and thus the grantee obtains full access permissions.	PermissionEnum.WriteAcp

Permission	Description	Value in OBS .NET SDK
FULL_CONTROL	<p>A grantee with this permission for a bucket has READ, WRITE, READ_ACP, and WRITE_ACP permissions for the bucket.</p> <p>A grantee with this permission for an object has READ, WRITE, READ_ACP, and WRITE_ACP permissions for the object.</p>	PermissionEnum.FullControl

There are five access control policies pre-defined in OBS, as described in the following table:

Policy	Description	Value in OBS .NET SDK
private	The owner of a bucket or object has the FULL_CONTROL permission for the bucket or object. Other users have no permission to access the bucket or object.	CannedAclEnum.Private
public-read	<p>If this permission is set for a bucket, everyone can obtain the list of objects, multipart uploads, and object versions in the bucket, as well as metadata of the bucket.</p> <p>If this permission is set for an object, everyone can obtain the content and metadata of the object.</p>	CannedAclEnum.PublicRead
public-read-write	<p>If this permission is set for a bucket, everyone can obtain the object list in the bucket, multipart uploads in the bucket, metadata of the bucket; upload objects; delete objects; initialize multipart uploads; upload parts; combine parts; copy parts; and abort multipart uploads.</p> <p>If this permission is set for an object, everyone can obtain the content and metadata of the object.</p>	CannedAclEnum.PublicReadWrite

Policy	Description	Value in OBS .NET SDK
public-read-delivered	If this permission is set for a bucket, everyone can obtain the object list, multipart uploads, and bucket metadata in the bucket, and obtain the content and metadata of the objects in the bucket. This permission cannot be set for objects.	CannedAclEnum.PublicReadDelivered
public-read-write-delivered	If this permission is set for a bucket, everyone can obtain the object list in the bucket, multipart uploads in the bucket, metadata of the bucket; upload objects; delete objects; initialize multipart uploads; upload parts; combine parts; copy parts; abort multipart uploads; and obtain content and metadata of objects in the bucket. This permission cannot be set for objects.	CannedAclEnum.PublicReadWriteDelivered

Specifying a Pre-defined ACL During Bucket Creation

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Create a bucket.
try
{
    CreateBucketRequest request = new CreateBucketRequest
    {
        BucketName = "bucketname",
        // Set the bucket ACL to public read and write.
        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);
}
```

Setting a Pre-defined ACL for a Bucket

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
//Set the bucket ACL.
try
{
    SetBucketAclRequest request = new SetBucketAclRequest
    {
        BucketName = "bucketname",
        // Set the bucket ACL to be private.
        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);
}
```

Setting a User-defined Bucket ACL

The following code shows how to set a user-defined ACL for a bucket:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
//Set the bucket ACL.
try
{
    //Set the bucket owner.
    Owner owner = new Owner
    {
        Id = "ownerid", //ID of the domain to which the owner belongs
    };
    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);
}
```

NOTE

The owner or grantee ID needed in the ACL indicates the account ID, which can be viewed on the **My Credentials** page of OBS Console.

Obtaining a Bucket ACL

You can call `ObsClient.GetBucketAcl` to obtain the bucket ACL. Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
//Obtain the bucket 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)
        {
```


 NOTE

For details about the format (JSON character string) of bucket policies, see the *Object Storage Service API Reference*.

Obtaining a Bucket Policy

You can call **ObsClient.GetBucketPolicy** to obtain a bucket policy. Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
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);
}
```

Deleting a Bucket Policy

You can call **ObsClient.DeleteBucketPolicy** to delete a bucket policy. Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
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 Obtaining a Bucket Location

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call **ObsClient.GetBucketLocation** to obtain the location of a bucket.

This example returns the region of bucket **bucketname**.

The example code is as follows:

```
// Initialize configuration parameters.  
ObsConfig config = new ObsConfig();  
config.Endpoint = "https://your-endpoint";  
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the  
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables  
// for identity authentication. Before running this example, configure environment variables AccessKeyID and  
// SecretAccessKey.  
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.  
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",  
EnvironmentVariableTarget.Machine);  
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",  
EnvironmentVariableTarget.Machine);  
// Create an instance of ObsClient.  
ObsClient client = new ObsClient(accessKey, secretKey, config);  
// Obtain the bucket location.  
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);  
}
```

NOTE

When creating a bucket, you can specify its location. For details, see [Creating a Bucket](#).

6.9 Obtaining Storage Information About a Bucket

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

The storage information about a bucket includes the used capacity of and the number of objects in the bucket.

You can call **ObsClient.GetBucketStorageInfo** to obtain the bucket storage information.

This example returns the storage information of bucket **bucketname**.

The example code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Obtain the storage information about a bucket.
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);
}
```

NOTE

- To handle the error codes possibly returned during the operation, see [OBS Server-Side Error Codes](#).

6.10 Setting or Obtaining a Bucket Quota

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

Setting a Bucket Quota

You can call **ObsClient.SetBucketQuota** to set the bucket quota. Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Set the bucket quota.
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);
}
```

NOTE

A bucket quota must be a non-negative integer expressed in bytes. The maximum value is $2^{63} - 1$.

Obtaining the Bucket Quota

You can call **ObsClient.GetBucketQuota** to obtain the bucket quota. Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
```

configuration file or environment variables. In this example, the AK/SK are stored in environment variables for identity authentication. Before running this example, configure environment variables **AccessKeyID** and **SecretAccessKey**.

// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca_01_0003.html.

```
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Obtain the bucket quota.
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 Setting or Obtaining the Storage Class of a Bucket

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

OBS allows you to set storage classes for buckets. The storage class of an object defaults to be that of its residing bucket. Different storage classes meet different needs for storage performance and costs. There are three types of storage class for buckets, as described in the following table:

Storage Class	Description	Value in OBS .NET SDK
OBS Standard	Features low access latency and high throughput and is applicable to storing frequently-accessed (multiple times per month) hotspot or small objects (< 1 MB) requiring quick response.	StorageClassEnum.Standard
OBS Infrequent Access	Is applicable to storing semi-frequently accessed (less than 12 times a year) data requiring quick response.	StorageClassEnum.Warm

Storage Class	Description	Value in OBS .NET SDK
OBS Archive	Is applicable to archiving rarely-accessed (once a year) data.	StorageClassEnum.Cold

For more information, see [Bucket Storage Classes](#).

Setting the Storage Class for a Bucket

You can call **ObsClient.SetBucketStoragePolicy** to set the storage class for a bucket. Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
Environment.VariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
Environment.VariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Set the storage class for a bucket.
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);
}
```

Obtaining the Storage Class of a Bucket

You can call **ObsClient.GetBucketStoragePolicy** to obtain the storage class. Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
Environment.VariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
```

```
EnvironmentVariableTarget.Machine);  
// Create an instance of ObsClient.  
ObsClient client = new ObsClient(accessKey, secretKey, config);  
// Obtain the storage class of a bucket.  
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 Object Upload

7.1 Object Upload Overview

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

In OBS, objects are basic data units that users can perform operations on. OBS .NET SDK provides abundant APIs for object upload in the following methods:

- [Performing a Streaming Upload](#)
- [Performing a File-Based Upload](#)
- [Performing an Asynchronous Upload](#)
- [Performing a Multipart Upload](#)
- [Performing an Appendable Upload](#)
- [Performing a Resumable Upload](#)

The SDK supports the upload of objects whose size ranges from 0 KB to 5 GB. If a file is smaller than 5 GB, streaming upload, appendable upload, and file-based upload are applicable. If the file is larger than 5 GB, multipart upload (whose part size is smaller than 5 GB) is suitable.

If you grant anonymous users the read permission for an object during the upload, anonymous users can access the object through a URL after the upload is complete. The object URL is in the format of **`https://bucket.name.domain.name/directory levels/object name`**. If the object resides in the root directory of the bucket, its URL does not contain directory levels.

7.2 Performing a Streaming Upload

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

Streaming upload uses **System.IO.Stream** as the data source of an object. You can call **ObsClient.PutObject** to upload the data streams to OBS. Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Upload a stream.
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);
}
```

NOTE

- To upload a large file, you are advised to use [multipart upload](#).
- The content to be uploaded cannot exceed 5 GB.

7.3 Performing a File-Based Upload

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

File-based upload uses local files as the data source of objects.

This example uploads local file **localfile** to bucket **bucketname** as object **objectname**.

The example code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an ObsClient instance.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Upload a file.
try
{
    PutObjectRequest request = new PutObjectRequest()
    {
        BucketName = "bucketname", //Name of the bucket for storing the file to be uploaded
        ObjectKey = "objectname", //Name of the file to be uploaded (The file name is the complete path of
// the file in the bucket, for example, folder/test.txt. The path does not contain the bucket name.)
        FilePath = "localfile", //Path of the local file to be uploaded, in which the file name must be specified
    };
    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);
}
```

NOTE

- The content to be uploaded cannot exceed 5 GB.

7.4 Performing an Asynchronous Upload

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call **ObsClient.BeginPutObject** and **ObsClient.EndPutObject** to upload an object asynchronously.

This example asynchronously uploads local file **localfile** to bucket **bucketname** as object **objectname**.

Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Upload a file in asynchronous mode.
try
{
    PutObjectRequest request = new PutObjectRequest()
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
        FilePath = "localfile", // Path of the local file to be uploaded. The file name must be specified.
    };
    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);
}
```

NOTE

- For more information, see [Object Upload Overview](#).

7.5 Obtaining Upload Progress

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call **PutObjectRequest.UploadProgress** to register a **System.EventHandler** callback function to obtain upload progress.

This example uploads local file **localfile** to bucket **bucketname** as object **objectname** and uses **System.EventHandler** to monitor the upload progress.

The example code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
Environment.VariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
Environment.VariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Upload a file.
try
{
    PutObjectRequest request = new PutObjectRequest()
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
        FilePath = "localfile", // Path of the local file uploaded. The file name must be specified.
    };
    // Represent the progress by showing how many bytes have been uploaded.
    request.ProgressType = ProgressTypeEnum.ByBytes;
    // Refresh the upload progress each time 1 MB data is uploaded.
    request.ProgressInterval = 1024 * 1024;

    // Register the upload progress callback function.
    request.UploadProgress += delegate(object sender, TransferStatus status){
        // Obtain the average upload rate.
        Console.WriteLine("AverageSpeed: {0}", status.AverageSpeed / 1024 + "KB/S");
        // Obtain the upload progress in percentage.
        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);
}
```

 NOTE

- You can query the upload progress when uploading an object in streaming, file-based, asynchronous, resumable, or appendable mode, or uploading a part.

7.6 Creating a Folder

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

There is no folder concept in OBS. All elements in buckets are objects. To create a folder in OBS is essentially to create an object whose size is 0 and whose name ends with a slash (/). Such objects have no difference from other objects and can be downloaded and deleted, except that they are displayed as folders in OBS Console.

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Create a folder.
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);
}
```

 NOTE

- To create a folder in OBS is to create an object whose size is 0 and whose name ends with a slash (/), in essential.
- To create a multi-level folder, you only need to create the folder with the last level. For example, if you want to create a folder named **src1/src2/src3/**, create it directly, no matter whether the **src1/** and **src1/src2/** folders exist.

7.7 Setting Object Properties

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can set properties for an object when uploading it. Object properties include the MIME type, MD5 value (for verification), storage class, and customized metadata. You can set properties for an object that is being uploaded in streaming, file-based, or multipart mode or when [copying the object](#).

The following table describes object properties.

Property Name	Description	Default Value
Content-Type	Indicates the MIME type of the object, which defines the type and network code of the object as well as in which mode and coding will the browser read the object.	binary/octet-stream
Content-MD5	Indicates the base64-encoded digest of the object data. It is provided to the OBS server to verify data integrity.	N/A
Storage class	Indicates the storage class of the object. Different storage classes meet different needs for storage performance and costs. The value defaults to be the same as the object's residing bucket and can be changed.	N/A
Customized metadata	Indicates the user-defined description of the object. It is used to facilitate the customized management on the object.	N/A

Setting the MIME Type for an Object

You can call `PutObjectRequest.ContentType` to set the MIME type for an object. Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Upload a file.
try
{
    PutObjectRequest request = new PutObjectRequest()
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
        FilePath = "localfile", //Path of the local file to upload. The path must contain the file name.
        ContentType = "image/jpeg", //MIME type of the object
    };
    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);
}
```

NOTE

If the MIME type is not specified, the SDK will automatically identify the MIME type according to the name suffix of the uploaded object. For example, if the name suffix of an object is `.xml` (`.html`), the object will be identified as an `application/xml` (`text/html`) file.

Setting the MD5 Value for an Object

You can call `PutObjectRequest.ContentMd5` to set the MD5 value for an object. Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Upload a file.
try
{
    PutObjectRequest request = new PutObjectRequest
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
        FilePath = "localfile", // Path of the local file uploaded. The file name must be specified.
    };
}
```

```
        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);
}
```

NOTE

- The MD5 value of an object must be a base64-encoded digest.
- The OBS server will compare this MD5 value with the MD5 value obtained by object data calculation. If the two values are not the same, the upload fails with HTTP status code **400** returned.
- If the MD5 value is not specified, the OBS server will skip MD5 value verification.

Setting the Storage Class for an Object

You can call **PutObjectRequest.StorageClass** to set the storage class for an object. Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Upload a file.
try
{
    PutObjectRequest request = new PutObjectRequest
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
        FilePath = "localfile", // Path of the local file uploaded. The file name must be specified.
        StorageClass = StorageClassEnum.Warm, // Object storage class
    };
    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);
}
```

 NOTE

- If you have not set the storage class for an object, the storage class of the object will be the same as that of its residing bucket.
- OBS provides objects with three storage classes which are consistent with [those](#) provided for buckets.
- Before downloading an Archive object, you must restore it.

Customizing Metadata for an Object

You can call **PutObjectRequest.Metadata** to customize metadata for an object. Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Upload a file.
try
{
    PutObjectRequest request = new PutObjectRequest
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
        FilePath = "localfile", // Path of the local file uploaded. The file name must be specified.
    };
    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);
}
```

 NOTE

- An object can have multiple pieces of metadata. The total metadata size cannot exceed 8 KB.
- The customized object metadata can be obtained by using **ObsClient.GetObjectMetadata**. For details, see [Obtaining Object Properties](#).
- When you call **ObsClient.GetObject** to download an object, its customized metadata will also be downloaded.

7.8 Performing a Multipart Upload

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

To upload a large file, multipart upload is recommended. Multipart upload is applicable to many scenarios, including:

- Files to be uploaded are larger than 100 MB.
- The network condition is poor. Connection to the OBS server is constantly down.
- Sizes of files to be uploaded are uncertain.

Multipart upload consists of three phases:

- Step 1** Initialize a multipart upload (**ObsClient.InitiateMultipartUpload**).
- Step 2** Upload parts one by one or concurrently (**ObsClient.UploadPart**).
- Step 3** Combine parts (**ObsClient.CompleteMultipartUpload**) or abort the multipart upload (**ObsClient.AbortMultipartUpload**).

----End

Initiating a Multipart Upload

Before using a multipart upload, you need to first initiate it. This operation will return an upload ID (globally unique identifier) created by the OBS server to identify the multipart upload. You can use this upload ID to initiate related operations, such as aborting a multipart upload, listing multipart uploads, and listing uploaded parts.

You can call **ObsClient.InitiateMultipartUpload** to initiate a multipart upload.

NOTE

- Call **InitiateMultipartUploadRequest** to specify the name and owning bucket of the uploaded object.
- In **InitiateMultipartUploadRequest**, you can specify the MIME type, storage class, and customized metadata for the object.
- The upload ID of the multipart upload returned by **InitiateMultipartUploadResponse.UploadId** will be used in follow-up operations.

Uploading Parts

After initiating a multipart upload, you can specify the object name and upload ID to upload a part. Each upload part has a part number (ranging from **1** to **10000**). For parts with the same upload ID, their part numbers are unique and identify

their relative location in the object. If you use the same part number to upload two parts, the latter one uploaded will overwrite the former one. Except for the last uploaded part whose size ranges from 0 to 5 GB, sizes of the other parts range from 100 KB to 5 GB. Parts can be uploaded in random order, or even through different processes or machines. OBS will combine them into the final object based on their part numbers.

You can call **ObsClient.UploadPart** to upload a part.

NOTE

- Except the part last uploaded, other parts must be larger than 100 KB. Part sizes will not be verified during upload because which one is last uploaded is not identified until parts are combined.
- OBS will return ETags (MD5 values) of the received parts to users.
- Part numbers range from 1 to 10000. If a part number exceeds this range, OBS will return error **400 Bad Request**.
- The minimum part size supported by an OBS 3.0 bucket is 100 KB, and the minimum part size supported by an OBS 2.0 bucket is 5 MB. You are advised to perform multipart upload to OBS 3.0 buckets.

Combining Parts

After all parts are uploaded, call the API for combining parts to generate the object. Before this operation, valid part numbers and ETags of all parts must be sent to OBS. After receiving this information, OBS verifies the validity of each part one by one. After all parts pass the verification, OBS combines these parts to form the final object.

You can call **ObsClient.CompleteMultipartUpload** to combine parts.

NOTE

Part numbers can be inconsecutive.

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
try
{
    // 1. Initialize a multipart upload.
    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. Upload parts.
string filePath = "localfile";// Path of the local file uploaded. The file name must be specified.
long contentLength = new FileInfo(filePath).Length;
long partSize = 15 * (long)Math.Pow(2, 20); // The part size is 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. Combine parts.

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);
}
```

Aborting a Multipart Upload

After a multipart upload is aborted, you cannot use its upload ID to perform any operation and the uploaded parts will be deleted by OBS.

When an object is being uploaded in multi-part mode or an object fails to be uploaded, parts are generated in the bucket. These parts occupy your storage space. You can cancel the multi-part uploading task to delete unnecessary parts, thereby saving the storage space.

You can call **ObsClient.AbortMultipartUpload** to abort a multipart upload. Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
```

```
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
//Abort a multipart upload.
try
{
    AbortMultipartUploadRequest request = new AbortMultipartUploadRequest
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
        UploadId = "uploadId",//ID of the multipart upload to be aborted
    };
    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);
}
```

Listing Uploaded Parts

You can call **ObsClient.ListParts** to list successfully uploaded parts of a multipart upload.

The following table describes the parameters involved in this API.

Parameter	Description	Property in OBS .NET SDK
BucketName	Bucket name	ListPartsRequest.BucketName
ObjectKey	Object name	ListPartsRequest.ObjectKey
UploadId	Upload ID, which globally identifies a multipart upload. The value is in the returned result of InitiateMultipartUpload .	ListPartsRequest.UploadId
MaxParts	Maximum number of parts that can be listed per page.	ListPartsRequest.MaxParts
PartNumberMarker	Part number after which listing parts begins. Only parts whose part numbers are larger than this value will be listed.	ListPartsRequest.PartNumberMarker

- Listing parts in simple mode

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
```

```
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Simple listing
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);
}
```

NOTE

Information about a maximum of 1000 parts can be listed each time. If an upload of the specific upload ID contains more than 1000 parts and **ListPartsResult.isTruncated** is **true** in the returned result, not all parts are listed. In such cases, you can use **ListPartsRespon.NextPartNumberMarker** to obtain the start position for next listing.

- Listing all parts

The following sample code lists more than 1,000 parts:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// List all parts.
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);
}
```

Listing Multipart Uploads

You can call **ObsClient.ListMultipartUploads** to list multipart uploads. The following table describes related parameters.

Parameter	Description	Property in OBS .NET SDK
BucketName	Bucket name	ListMultipartUploadsRequest.BucketName
Prefix	Prefix that the object names in the multipart uploads to be listed must contain	ListMultipartUploadsRequest.Prefix
Delimiter	Character used to group object names involved in multipart uploads. If the object name contains the Delimiter parameter, the character string from the first character to the first delimiter in the object name is grouped under a single result element, CommonPrefix . (If a prefix is specified in the request, the prefix must be removed from the object name.)	ListMultipartUploadsRequest.Delimiter
MaxUploads	Maximum number of multipart uploads listed in the response body. The value ranges from 1 to 1000 . If the value exceeds 1000 , only 1,000 multipart uploads are returned.	ListMultipartUploadsRequest.MaxUploads
KeyMarker	Object name to start with when listing multipart uploads	ListMultipartUploadsRequest.KeyMarker

Parameter	Description	Property in OBS .NET SDK
UploadIdMarker	Upload ID after which the multipart upload listing begins. It is effective only when used with KeyMarker so that multipart uploads after UploadIdMarker of KeyMarker will be listed.	ListMultipartUploadsRequest.UploadIdMarker

- Listing multipart uploads in simple mode

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
//List multipart uploads in simple mode.
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);
}
```

NOTE

- Information about a maximum of 1000 multipart uploads can be listed each time. If a bucket contains more than 1000 multipart uploads and **ListMultipartUploadsResponse.isTruncated** is **true**, not all uploads are listed. In such cases, you can use **ListMultipartUploadsResponse.NextKeyMarker** and **ListMultipartUploadsResponse.NextUploadIdMarker** to obtain the start position for next listing.
- If you want to obtain all multipart uploads in a bucket, you can list them in paging mode.
- Listing all multipart uploads in paging mode

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
```

```
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
//List all multipart uploads.
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 Configuring Lifecycle Management

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

When uploading an object or initializing a multipart upload, you can directly set the expiration time for the object. Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
```

```
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);

try
{
    PutObjectRequest request = new PutObjectRequest()
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
        FilePath = "localfile", // Path of the local file uploaded. The file name must be specified.
        Expires = 30 // When uploading an object, set the object to expire after 30 days.
    };
    PutObjectResponse response = client.PutObject(request);
    Console.WriteLine("put object response: {0}", response.StatusCode);

    InitiateMultipartUploadRequest initiateRequest = new InitiateMultipartUploadRequest
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
        // When initializing a multipart upload, set the object to expire 60 days after combination.
        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);
}
```

NOTE

- The previous mode specifies the time duration in days after which an object will expire. The OBS server automatically clears expired objects.
- The object expiration time set in the preceding method takes precedence over the bucket lifecycle rule.

7.10 Performing an Appendable Upload

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

Appendable upload allows you to upload an object in appendable mode and then append data to the object. You can call **ObsClient.AppendObject** to perform an appendable upload. Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
```

```
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
try
{
    // Upload an object in appendable mode.
    AppendObjectRequest request = new AppendObjectRequest();
    request.BucketName = "bucketname";
    request.ObjectKey = "objectkey";
    request.InputStream = new MemoryStream(Encoding.UTF8.GetBytes("Hello OBS"));

    AppendObjectResponse response = client.AppendObject(request);

    // Append data to the object.
    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);

    // Use the API for obtaining object properties to get the start position for next appending.
    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);
}
```

NOTE

- Objects uploaded using **ObsClient.PutObject**, referred to as normal objects, can overwrite objects uploaded using **ObsClient.AppendObject**, referred to as appendable objects. Data cannot be appended to an appendable object once the object has been overwritten by a normal object.
- When you upload an object for the first time in append mode, an exception will be reported (HTTP status code **409**) if an object of the same name exists.
- The ETag returned for each append upload is the ETag for the uploaded content, rather than that of the whole object.
- Data appended each time can be up to 5 GB, and 10000 times of appendable uploads can be performed on a single object.
- After an appendable upload is successful, you can use **AppendObjectResponse.NextPosition** or call **ObsClient.GetObjectMetadata** to get the start position for next appending.

7.11 Performing a Multipart Copy

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

As a special case of multipart upload, multipart copy implements multipart upload by copying the whole or part of an object in a bucket. You can call **ObsClient.CopyPart** to copy parts.

This example copies object **sourceobjectname** from bucket **sourcebucketname** to bucket **destbucketname** as object **destobjectname**.

The example code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Copy parts.
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 Performing a Resumable Upload

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

Uploading large files often fails due to poor network conditions or program breakdowns. It is a waste of resources to restart the upload process upon an upload failure, and the restarted upload process may still suffer from the unstable network. To resolve such issues, you can use the API for resumable upload, whose working principle is to divide the to-be-uploaded file into multiple parts and upload them separately. The upload result of each part is recorded in a checkpoint file in real time. Only when all parts are successfully uploaded, the result indicating a successful upload is returned. Otherwise, an exception is thrown to remind you of calling the API again for re-uploading. Based on the upload status

of each part recorded in the checkpoint file, the re-uploading will upload the parts failed to be uploaded previously, instead of uploading all parts. By virtue of this, resources are saved and efficiency is improved.

You can call **ObsClient.UploadFile** to perform a resumable upload. The following table describes the parameters involved in this API.

Parameter	Description	Property in OBS .NET SDK
BucketName	(Mandatory) Bucket name	UploadFileRequest.BucketName
ObjectKey	(Mandatory) Object name	UploadFileRequest.ObjectKey
UploadFile	(Mandatory) Path to the local file to be uploaded	UploadFileRequest.UploadFile
UploadPartSize	Part size, in bytes. The value ranges from 5 MB (default) to 5 GB.	UploadFileRequest.UploadPartSize
EnableCheckpoint	Whether to enable the resumable upload mode. The default value is false , which indicates that this mode is disabled.	UploadFileRequest.EnableCheckpoint
CheckpointFile	File used to record the upload progress. This parameter is effective only in the resumable upload mode. If the value is null, the file is in the same directory as the local file to be uploaded.	UploadFileRequest.CheckpointFile
Metadata	Customized metadata of the object	UploadFileRequest.Metadata
EnableCheckSum	Whether to verify the content of the to-be-uploaded file. This parameter is effective only in the resumable upload mode. The default value is false , which indicates that the content will not be verified.	UploadFileRequest.EnableCheckSum
TaskNum	Maximum number of parts that can be concurrently uploaded. The default value is 1 .	UploadFileRequest.TaskNum
UploadProgress	Upload progress callback function	UploadFileRequest.UploadProgress

Parameter	Description	Property in OBS .NET SDK
ProgressType	Mode for representing the upload progress	UploadFileRequest.ProgressType
ProgressInterval	Interval for refreshing the upload progress	UploadFileRequest.ProgressInterval
UploadEventHandler	Upload event callback function	UploadFileRequest.UploadEventHandler

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Perform a resumable upload.
try
{
    UploadFileRequest request = new UploadFileRequest
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
        // Specify the local file to be uploaded.
        UploadFile = "localpath",
        // Set the part size to 10 MB.
        UploadPartSize = 10 * 1024 * 1024,
        // Enable the resumable download mode.
        EnableCheckpoint = true,
    };
    // Represent the progress by showing how many bytes have been uploaded.
    request.ProgressType = ProgressTypeEnum.ByBytes;
    // Refresh the upload progress each time 1 MB data is uploaded.
    request.ProgressInterval = 1024 * 1024;

    // Register the upload progress callback function.
    request.UploadProgress += delegate(object sender, TransferStatus status){
        // Obtain the average upload rate.
        Console.WriteLine("AverageSpeed: {0}", status.AverageSpeed / 1024 + "KB/S");
        // Obtain the upload progress in percentage.
        Console.WriteLine("TransferPercentage: {0}", status.TransferPercentage);
    };

    // Register the upload event callback function.
    request.UploadEventHandler += delegate(object sender, ResumableUploadEvent e){
        // Obtain the upload events.
        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);  
}
```

 **NOTE**

- The API for resumable upload, which is implemented based on **multipart upload**, is an encapsulated and enhanced version of multipart upload.
- This API saves resources and improves efficiency upon the re-upload, and speeds up the upload process by concurrently uploading parts. Because this API is transparent to users, users are free from concerns about internal service details, such as the creation and deletion of checkpoint files, division of objects, and concurrent upload of parts.
- The default value of the **EnableCheckpoint** parameter is **false**, which indicates that the resumable upload mode is disabled. In such cases, the API for resumable upload degrades to the simple encapsulation of multipart upload, and no checkpoint file will be generated.
- The **CheckpointFile** and **EnableCheckSum** parameters are valid only when **EnableCheckpoint** is **true**.

8 Object Download

8.1 Object Download Overview

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

OBS .NET SDK provides abundant APIs for downloading objects in the following modes:

- [Performing a Streaming Download](#)
- [Performing a Partial Download](#)
- [Performing an Asynchronous Download](#)
- [Performing a Conditioned Download](#)
- [Performing a Resumable Download](#)

You can call `ObsClient.GetObject` to download an object.

8.2 Performing a Streaming Download

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

Sample code:

```
// Initialize configuration parameters.  
ObsConfig config = new ObsConfig();  
config.Endpoint = "https://your-endpoint";
```

```
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
Environment.VariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
Environment.VariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Download an object.
try
{
    GetObjectRequest request = new GetObjectRequest()
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
    };
    using (GetObjectResponse response = client.GetObject(request))
    {
        string dest = "savepath";
        if (!File.Exists(dest))
        {
            // Write the data streams into the file.
            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);
}
```

NOTE

GetObjectResponse.OutputStream (**System.IO.Stream** type) is the response stream in **GetObjectResponse**. You can obtain the object content to a local file or memory via **GetObjectResponse.OutputStream**. Alternatively, you can call **GetObjectResponse.WriteResponseStreamToFile** provided by OBS .NET SDK to download the object content to a local file.

NOTICE

Object response streams obtained by **GetObjectResponse.OutputStream** must be closed explicitly using a **GetObjectResponse.OutputStream.Close()** call. Otherwise, resource leakage may occur.

8.3 Performing a Partial Download

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

When only partial data of an object is required, you can download data falling within a specific range.

If the specified range is from 0 to 1,000, data from byte 0 to byte 1,000, 1,001 bytes in total, are returned. If the specified range is invalid, the entire object will be returned.

This example downloads the content of **objectname** in **bucketname** from byte 10 to byte 200.

The sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Download an object.
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 Performing an Asynchronous Download

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call **ObsClient.BeginGetObject** and **ObsClient.EndGetObject** to download an object asynchronously.

This example downloads object **objectname** from bucket **bucketname** asynchronously.

Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Download an object in asynchronous mode.
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))
                {
                    // Write the data streams into the file.
                    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);
}
```

NOTE

- For more information, see [Performing a Streaming Download](#).

8.5 Obtaining Download Progress

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call `GetObjectRequest.DownloadProgress` to register the `System.EventHandler` callback function to obtain download progress.

This example downloads `objectname` from `bucketname` and uses `System.EventHandler` to monitor the download progress.

The example code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Download an object.
try
{
    GetObjectRequest request = new GetObjectRequest()
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
    };
    // Represent the progress by showing how many bytes have been downloaded.
    request.ProgressType = ProgressTypeEnum.ByBytes;
    // Refresh the download progress each time 1 MB data is downloaded.
    request.ProgressInterval = 1024 * 1024;

    // Register the download progress callback function.
    request.DownloadProgress += delegate(object sender, TransferStatus status){
        // Obtain the average download rate.
        Console.WriteLine("AverageSpeed: {0}", status.AverageSpeed / 1024 + "KB/S");
        // Obtain the download progress in percentage.
        Console.WriteLine("TransferPercentage: {0}", status.TransferPercentage);
    };
    using (GetObjectResponse response = client.GetObject(request))
    {
        string dest = "savepath";
        if (!File.Exists(dest))
        {
            // Write the data streams into the file.
            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);
}
```

 **NOTE**

- You can obtain the download progress when downloading an object in streaming, partial, asynchronous, or resumable mode.

8.6 Performing a Conditioned Download

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

When downloading an object, you can specify one or more conditions. Only when the conditions are met, the object will be downloaded. Otherwise, an exception indicating a download failure will be thrown.

You can set the following conditions:

Parameter	Description	Property in OBS .NET SDK
If-Modified-Since	Returns the object if it has been modified since the specified time; otherwise, an error is returned.	GetObjectRequest.IfModifiedSince
If-Unmodified-Since	Returns the object if it has not been modified since the specified time; otherwise, an error is returned.	GetObjectRequest.IfUnmodifiedSince
If-Match	Returns the source object if its ETag is the same as the one specified by this parameter; otherwise, an exception is thrown.	GetObjectRequest.IfMatch
If-None-Match	Returns the source object if its ETag is different from the one specified by this parameter; otherwise, an exception is thrown.	GetObjectRequest.IfNoneMatch

 NOTE

- The ETag of an object is the MD5 check value of the object.
- If the download request includes **IfUnmodifiedSince** or **IfMatch** and the specified condition is not met, an exception will be thrown with HTTP status code **412 Precondition Failed** returned.
- If the download request includes **IfModifiedSince** or **IfNoneMatch** and the specified condition is not met, an exception will be thrown with HTTP status code **304 Not Modified** returned.

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Download an object in conditioned mode.
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 Rewriting Response Headers

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

When downloading an object, you can rewrite some HTTP/HTTPS response headers. The following table lists rewritable response headers.

Parameter	Description	Property in OBS .NET SDK
ContentType	Rewrites Content-Type in HTTP/HTTPS responses.	GetObjectRequest.ResponseHeaderOverrides.ContentType
ContentLanguage	Rewrites Content-Language in HTTP/HTTPS responses.	GetObjectRequest.ResponseHeaderOverrides.ContentLanguage
Expires	Rewrites Expires in HTTP/HTTPS responses.	ObjectReplaceMetadata.Expires
CacheControl	Rewrites Cache-Control in HTTP/HTTPS responses.	GetObjectRequest.ResponseHeaderOverrides.CacheControl
ContentDisposition	Rewrites Content-Disposition in HTTP/HTTPS responses.	GetObjectRequest.ResponseHeaderOverrides.ContentDisposition
ContentEncoding	Rewrites Content-Encoding in HTTP/HTTPS responses.	GetObjectRequest.ResponseHeaderOverrides.ContentEncoding

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Rewrite response headers when downloading an object.
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("Content-Type: {0}", response.ContentType);
    response.Dispose();
}
```

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

8.8 Obtaining Custom Metadata

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

After an object is successfully downloaded, its custom data is returned.

This example obtains the custom metadata of **objectname** in **bucketname**.

The example code is as follows:

```
// Initialize configuration parameters.  
ObsConfig config = new ObsConfig();  
config.Endpoint = "https://your-endpoint";  
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the  
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables  
// for identity authentication. Before running this example, configure environment variables AccessKeyID and  
// SecretAccessKey.  
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.  
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",  
EnvironmentVariableTarget.Machine);  
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",  
EnvironmentVariableTarget.Machine);  
// Create an instance of ObsClient.  
ObsClient client = new ObsClient(accessKey, secretKey, config);  
// Download an object.  
try  
{  
    GetObjectRequest request = new GetObjectRequest()  
    {  
        BucketName = "bucketname",  
        ObjectKey = "objectname",  
    };  
    using (GetObjectResponse response = client.GetObject(request))  
    {  
        //Obtain the custom metadata of the object.  
        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 Downloading an Archive Object

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

Before you can download an Archive object, you must restore it. Archive objects can be restored in either of the following ways.

Option	Description	Value in OBS .NET SDK
Expedited restore	Data can be restored within 1 to 5 minutes.	RestoreTierEnum.Expedited
Standard restore	Data can be restored within 3 to 5 hours. This is the default option.	RestoreTierEnum.Standard

⚠ CAUTION

To prolong the validity period of the Archive data restored, you can repeatedly restore the data, but you will be billed for each restoration. After a second restore, the validity period of Standard object copies will be prolonged, and you need to pay for storing these copies during the prolonged period.

You can call **ObsClient.RestoreObject** to restore Archive objects. Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
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;
```

```
// This parameter is optional. By default, the latest object version is restored. You can set versionId to restore a specified object version.
// 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);
}
```

NOTE

- The object specified in **ObsClient.RestoreObject** must be in the OBS Archive storage class. Otherwise, an exception will be thrown when you call this API.
- **RestoreObjectRequest.Days** specifies the retention period (1 to 30 days) of the restored object.
- **RestoreObjectRequest.Tier** specifies the restore option, which indicates the time spend on restoring an object.

8.10 Performing a Resumable Download

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

Downloading large files often fails due to poor network conditions or program breakdowns. It is a waste of resources to restart the download process upon a download failure, and the restarted download process may still suffer from the unstable network. To resolve such issues, you can use the API for resumable download, whose working principle is to divide the to-be-downloaded file into multiple parts and download them separately. The download result of each part is recorded in a checkpoint file in real time. Only when all parts are successfully downloaded, the result indicating a successful download is returned. Otherwise, an exception is thrown to remind you of calling the API again for re-downloading. Based on the download status of each part recorded in the checkpoint file, the re-downloading will download the parts failed to be downloaded previously, instead of downloading all parts. By virtue of this, resources are saved and efficiency is improved.

You can call **ObsClient.DownloadFile** to perform a resumable download. The following table describes the parameters involved in this API.

Parameter	Description	Property in OBS .NET SDK
BucketName	(Mandatory) Bucket name	DownloadFileRequest.BucketName
ObjectKey	(Mandatory) Object name	DownloadFileRequest.ObjectKey

Parameter	Description	Property in OBS .NET SDK
DownloadFile	Full path of the local directory to which the object is downloaded. If the value is null, the downloaded object is saved in the directory where the program is executed.	DownloadFileRequest.DownloadFile
DownloadPartSize	Part size, in bytes. The value ranges from 5 MB (default) to 5 GB.	DownloadFileRequest.DownloadPartSize
EnableCheckpoint	Whether to enable the resumable upload mode. The default value is false , which indicates that this mode is disabled.	DownloadFileRequest.EnableCheckpoint
CheckpointFile	File used to record the download progress. This parameter is effective only in the resumable download mode. If the value is null, the file is in the same local directory as the downloaded object.	DownloadFileRequest.CheckpointFile
VersionId	Object version ID	DownloadFileRequest.VersionId
TaskNum	Maximum number of parts that can be concurrently downloaded. The default value is 1 .	DownloadFileRequest.TaskNum
DownloadProgress	Download progress callback function	DownloadFileRequest.DownloadProgress
ProgressType	Mode for representing the download progress	DownloadFileRequest.ProgressType
ProgressInterval	Interval for refreshing the download progress	DownloadFileRequest.ProgressInterval
DownloadEventHandler	Download event callback function	DownloadFileRequest.DownloadEventHandler

Sample code:

```
// Initialize configuration parameters.  
ObsConfig config = new ObsConfig();  
config.Endpoint = "https://your-endpoint";
```

```
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Perform a resumable download.
try
{
    DownloadFileRequest request = new DownloadFileRequest
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
        // Specify the local file to be downloaded.
        DownloadFile = "savepath",
        // Set the part size to 10 MB.
        DownloadPartSize = 1024 * 1024 * 10,
        // Enable the resumable download mode.
        EnableCheckpoint = true,
    };

    // Represent the progress by showing how many bytes have been downloaded.
    request.ProgressType = ProgressTypeEnum.ByBytes;
    // Refresh the download progress each time 1 MB data is downloaded.
    request.ProgressInterval = 1024 * 1024;

    // Register the download progress callback function.
    request.DownloadProgress += delegate(object sender, TransferStatus status){
        // Obtain the average download rate.
        Console.WriteLine("AverageSpeed: {0}", status.AverageSpeed / 1024 + "KB/S");
        // Obtain the download progress in percentage.
        Console.WriteLine("TransferPercentage: {0}", status.TransferPercentage);
    };

    // Register the download event callback function.
    request.DownloadEventHandler += delegate(object sender, ResumableDownloadEvent e){
        // Obtain the download events.
        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);
}
```

 NOTE

- The API for resumable download, which is implemented based on [partial download](#), is an encapsulated and enhanced version of partial download.
- This API saves resources and improves efficiency upon the re-download, and speeds up the download process by concurrently downloading parts. Because this API is transparent to users, users are free from concerns about internal service details, such as the creation and deletion of checkpoint files, division of objects, and concurrent download of parts.
- The default value of the **EnableCheckpoint** parameter is **false**, which indicates that the resumable download mode is disabled. In such cases, the API for resumable download degrades to the simple encapsulation of partial download, and no checkpoint file will be generated.
- **CheckpointFile** is effective only when **EnableCheckpoint** is **true**.

8.11 Processing an Image

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

OBS can be used to process images in a stable, secure, efficient, easy to use, and cost-efficient manner. If the object to be downloaded is an image, you can pass the image processing parameters to operate it, including cutting and resizing it as well as putting a watermark and converting the format.

For more information, see [Image Processing Feature Guide](#).

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);

try
{
    GetObjectRequest request = new GetObjectRequest()
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
        // Resize and rotate the image in sequence.
        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);  
}
```

 **NOTE**

- Use **GetObjectRequest.ImageProcess** to specify the image processing parameters.
- Image processing parameters can be processed in cascading mode. This indicates that multiple commands can be performed on an image in sequence.

9 Object Management

9.1 Obtaining Object Properties

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call **ObsClient.GetObjectMetadata** to obtain object properties, including the last modification time, version ID, and customized metadata. Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);

try
{
    GetObjectMetadataRequest request = new GetObjectMetadataRequest();
    // Specify a bucket name.
    request.BucketName = "bucketname";
    // Specify an object (example/objectname as an example).
    request.ObjectKey = "example/objectname";
    // Obtain the object metadata.
    using (GetObjectMetadataResponse response = client.GetObjectMetadata(request)) {
        Console.WriteLine("Get object metadata response: {0}", response.StatusCode);
    }
    // Obtain ETag of the object.
    Console.WriteLine("Object etag {0}: ", response.ETag);
    // Obtain the object version ID.
    Console.WriteLine("Object versionId {0}: ", response.VersionId);
}
```

```
// Obtain the length of the object data, in bytes.
    Console.WriteLine("Object contentLength {0}: ", response.ContentLength);
}
}
catch (ObsException ex)
{
    Console.WriteLine("Message: {0}", ex.Message);
}
```

NOTE

- Due to HTTP coding restrictions, non-ASCII characters cannot be sent. The SDK will decode the information in response headers using URL decoding rules.

9.2 Managing Object ACLs

Access control lists (ACLs) allow resource owners to grant other accounts the permissions to access resources. By default, only the resource owner has full control over resources when a bucket or object is created. That is, the bucket creator has full control over the bucket, and the object uploader has full control over the object. Other accounts do not have the permissions to access resources. If resource owners want to grant other accounts the read and write permissions on resources, they can use ACLs. ACLs grant permissions to accounts. After an account is granted permissions, both the account and its IAM users can access the resources.

For more information, see [ACLs](#).

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

An object ACL can be configured in any of the following ways:

1. Specify a pre-defined ACL during object upload.
2. Call `ObsClient.SetObjectAcl` to specify a pre-defined ACL.
3. Call `ObsClient.SetObjectAcl` to specify a user-defined ACL.

Specifying a Pre-defined ACL During Object Upload

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
```

```
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Set a pre-defined ACL for an object during the upload.
try
{
    PutObjectRequest request = new PutObjectRequest
    {
        BucketName = "bucketname",
        ObjectKey = "objectname",
        // Set the object ACL to public read and write.
        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);
}
```

Setting a Pre-defined ACL for an Object

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Set a pre-defined ACL for the object.
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);
}
```

Setting a User-defined Object ACL

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/
```

```
intl/en-us/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Set a user-defined object ACL.
try
{
    SetObjectAclRequest request = new SetObjectAclRequest();
    request.BucketName = "bucketname";
    request.ObjectKey = "objectname";
    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);
}
```

NOTE

The owner or grantee ID needed in the ACL indicates the account ID, which can be viewed on the **My Credentials** page of OBS Console.

Obtaining an Object ACL

You can call `ObsClient.GetObjectAcl` to obtain the ACL of an object. Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Obtain the ACL of an object.
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 Listing Objects

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call **ObsClient.ListObjects** to list objects in a bucket.

The following table describes the parameters involved in this API.

Parameter	Description	Property in OBS .NET SDK
BucketName	Bucket name	ListObjectsRequest.BucketName
Prefix	Name prefix that the objects to be listed must contain	ListObjectsRequest.Prefix
Marker	Object name to start with when listing objects in a bucket. All objects are listed in the lexicographical order.	ListObjectsRequest.Marker
MaxKeys	Maximum number of objects listed in the response body. The value ranges from 1 to 1000 . If the value exceeds 1000 , only 1,000 objects are returned.	ListObjectsRequest.MaxKeys

Parameter	Description	Property in OBS .NET SDK
Delimiter	<p>Character used to group object names. If the object name contains the Delimiter parameter, the character string from the first character to the first delimiter in the object name is grouped under a single result element, CommonPrefix. (If a prefix is specified in the request, the prefix must be removed from the object name.)</p> <p>For a parallel file system, if this parameter is not specified, all the content in the directory is recursively listed by default, and subdirectories are also listed. In big data scenarios, parallel file systems usually have deep directory levels and each directory has a large number of files. In such case, you are advised to configure [delimiter="/"] to list the content in the current directory, but not list subdirectories, thereby improving the listing efficiency.</p>	ListObjectsRequest.Delimiter

Listing Objects in Simple Mode

The following sample code shows how to list objects in simple mode. A maximum of 1000 objects can be returned.

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Simple listing
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);
}
```

NOTE

Information about a maximum of 1000 objects can be listed each time. If a bucket contains more than 1000 objects and **ListObjectsResponse.IsTruncated** is **true** in the returned result, not all objects are listed. In such cases, you can use **ListObjectsResponse.NextMarker** to obtain the start position for next listing.

Listing Objects by Specifying the Number

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Specify the number.
try
{
    ListObjectsRequest request = new ListObjectsRequest();
    request.BucketName = "bucketname";
    // Set the number of objects to be listed to 100.
    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);
}
```

Listing Objects by Specifying a Prefix

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/
```

```
intl/en-us/usermanual-ca/ca_01_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// List objects by specifying a prefix.
try
{
    ListObjectsRequest request = new ListObjectsRequest();
    request.BucketName = "bucketname";
    //Specify the prefix.
    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);
}
```

Listing Objects by Specifying the Start Position

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKeyID= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// List objects by specifying the start position.
try
{
    ListObjectsRequest request = new ListObjectsRequest();
    request.BucketName = "bucketname";
    //Specify the start position for listing.
    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);
}
```

Listing All Objects in Paging Mode

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
//List all objects in paging mode.
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);
}
```

Listing All Objects in a Folder

There is no folder concept in OBS. All elements in buckets are objects. Folders are actually objects whose sizes are 0 and whose names end with a slash (/). When you set a folder name as the prefix, objects in this folder will be listed. Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
//List all objects in a folder.
try
{
    ListObjectsRequest request = new ListObjectsRequest();
    ListObjectsResponse response;
    request.BucketName = "bucketname";
    request.MaxKeys = 1000;
```

```
// Set the prefix to dir/.
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 Deleting Objects

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

NOTE

Exercise caution when performing this operation. If the versioning function is disabled for the bucket where the object is located, the object cannot be restored after being deleted.

Deleting a Single Object

You can call **ObsClient.DeleteObject** to delete a single object. Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Delete an object.
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);
    }
}
```

Batch Deleting Objects

You can call **ObsClient.DeleteObjects** to delete objects in a batch.

A maximum of 1000 objects can be deleted each time. Two response modes are supported: **verbose** (detailed) and **quiet** (brief).

- In verbose mode (default mode), the returned response includes the deletion result of each requested object.
- In quiet mode, the returned response includes only results of objects failed to be deleted.

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Delete objects in a batch.
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 Copying an Object

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

The object copy operation can create a copy for an existing object in OBS.

You can call **ObsClient.CopyObject** to copy an object. When copying an object, you can specify properties and ACL for it.

NOTE

- If the source object to be copied is in the Archive storage class, you must restore it first.

Copying an Object in Simple Mode

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);

// Copy an object.
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);
}
```

Rewriting Object Properties

The following sample code shows how to rewrite object properties.

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
//Rewrite object properties.
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);
}
}
```

Copying an Object by Specifying Conditions

When copying an object, you can specify one or more restriction conditions. If the conditions are met, the object will be copied. Otherwise, an exception will be thrown and the copy will fail.

You can set the following conditions:

Parameter	Description	Property in OBS .NET SDK
Copy-Source-If-Modified-Since	Copies the source object if it has been modified since the specified time; otherwise, an exception is thrown.	CopyObjectRequest.IfModifiedSince
Copy-Source-If-Unmodified-Since	Copies the source object if it has not been modified since the specified time; otherwise, an exception is thrown.	CopyObjectRequest.IfUnmodifiedSince

Parameter	Description	Property in OBS .NET SDK
Copy-Source-If-Match	Copies the source object if its ETag is the same as the one specified by this parameter; otherwise, an exception is thrown.	CopyObjectRequest.IfMatch
Copy-Source-If-None-Match	Copies the source object if its ETag is different from the one specified by this parameter; otherwise, an exception is thrown.	CopyObjectRequest.IfNoneMatch

 **NOTE**

- The ETag of the source object is the MD5 check value of the source object.
- If the object copy request includes **IfUnmodifiedSince**, **IfMatch**, **IfModifiedSince**, or **IfNoneMatch**, and the specified condition is not met, the copy will fail and an exception will be thrown with HTTP status code **412 Precondition Failed** returned.
- **IfModifiedSince** and **IfNoneMatch** can be used together and so do **IfUnmodifiedSince** and **IfMatch**.

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);

// Set the condition for restricting the copy.
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);
}
```

Rewriting an Object ACL

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
Environment.VariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
Environment.VariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Rewrite the object 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 HEAD Object

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call **ObsClient.HeadObject** to check whether a specified object exists.

The example code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
Environment.VariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
Environment.VariableTarget.Machine);
// Create an instance of ObsClient.
```

```
ObsClient client = new ObsClient(accessKey, secretKey, config);  
// Check whether the specified object exists.  
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);  
}
```

 **NOTE**

- If an exception is thrown and the returned HTTP status code is **404**, the bucket does not exist.

10 Temporarily Authorized Access

10.1 Using a Temporary URL for Authorized Access

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

ObsClient allows you to create a URL whose **Query** parameters are carried with authentication information by specifying the AK and SK, HTTP method, and request parameters. You can provide other users with this URL for temporary access. When generating a URL, you need to specify the validity period of the URL to restrict the access duration of visitors.

If you want to grant other users the permission to perform other operations on buckets or objects (for example, upload or download objects), generate a URL with the corresponding request (for example, to upload an object using the URL that generates the PUT request) and provide the URL for other users.

The following table lists operations can be performed through a signed URL.

Operation	HTTP Request Method (Value in OBS .NET SDK)	Sub-resource (Value in OBS .NET SDK)	Bucket Name Required	Object Name Required
PUT Bucket	HttpVerb.PUT	N/A	Yes	No
GET Buckets	HttpVerb.GET	N/A	No	No
DELETE Bucket	HttpVerb.DELETE	N/A	Yes	No

Operation	HTTP Request Method (Value in OBS .NET SDK)	Sub-resource (Value in OBS .NET SDK)	Bucket Name Required	Object Name Required
GET Objects	HttpVerb.GET	N/A	Yes	No
GET Object versions	HttpVerb.GET	SubResourceEnum.Versions	Yes	No
List Multipart uploads	HttpVerb.GET	SubResourceEnum.Uploads	Yes	No
Obtain Bucket Metadata	HttpVerb.HEAD	N/A	Yes	No
GET Bucket location	HttpVerb.GET	SubResourceEnum.Location	Yes	No
GET Bucket storage info	HttpVerb.GET	SubResourceEnum.StorageInfo	Yes	No
PUT Bucket quota	HttpVerb.PUT	SubResourceEnum.Quota	Yes	No
GET Bucket quota	HttpVerb.GET	SubResourceEnum.Quota	Yes	No
PUT Bucket storage Policy	HttpVerb.PUT	SubResourceEnum.StoragePolicy	Yes	No
GET Bucket storage Policy	HttpVerb.GET	SubResourceEnum.StoragePolicy	Yes	No
PUT Bucket ACL	HttpVerb.PUT	SubResourceEnum.Acl	Yes	No

Operation	HTTP Request Method (Value in OBS .NET SDK)	Sub-resource (Value in OBS .NET SDK)	Bucket Name Required	Object Name Required
GET Bucket ACL	HttpVerb.GET	SubResourceEnum.Acl	Yes	No
PUT Bucket logging	HttpVerb.PUT	SubResourceEnum.Logging	Yes	No
GET Bucket logging	HttpVerb.GET	SubResourceEnum.Logging	Yes	No
PUT Bucket policy	HttpVerb.PUT	SubResourceEnum.Policy	Yes	No
GET Bucket policy	HttpVerb.GET	SubResourceEnum.Policy	Yes	No
DELETE Bucket policy	HttpVerb.DELETE	SubResourceEnum.Policy	Yes	No
PUT Bucket lifecycle	HttpVerb.PUT	SubResourceEnum.Lifecycle	Yes	No
GET Bucket lifecycle	HttpVerb.GET	SubResourceEnum.Lifecycle	Yes	No
DELETE Bucket lifecycle	HttpVerb.DELETE	SubResourceEnum.Lifecycle	Yes	No
PUT Bucket website	HttpVerb.PUT	SubResourceEnum.Website	Yes	No
GET Bucket website	HttpVerb.GET	SubResourceEnum.Website	Yes	No
DELETE Bucket website	HttpVerb.DELETE	SubResourceEnum.Website	Yes	No

Operation	HTTP Request Method (Value in OBS .NET SDK)	Sub-resource (Value in OBS .NET SDK)	Bucket Name Required	Object Name Required
PUT Bucket versioning	HttpVerb.PUT	SubResourceEnum.Versioning	Yes	No
GET Bucket versioning	HttpVerb.GET	SubResourceEnum.Versioning	Yes	No
PUT Bucket cors	HttpVerb.PUT	SubResourceEnum.Cors	Yes	No
GET Bucket cors	HttpVerb.GET	SubResourceEnum.Cors	Yes	No
DELETE Bucket cors	HttpVerb.DELETE	SubResourceEnum.Cors	Yes	No
PUT Bucket tagging	HttpVerb.PUT	SubResourceEnum.Tagging	Yes	No
GET Bucket tagging	HttpVerb.GET	SubResourceEnum.Tagging	Yes	No
DELETE Bucket tagging	HttpVerb.DELETE	SubResourceEnum.Tagging	Yes	No
PUT Object	HttpVerb.PUT	N/A	Yes	Yes
Append Object	HttpVerb.POST	SubResourceEnum.Append	Yes	Yes
GET Object	HttpVerb.GET	N/A	Yes	Yes
PUT Object - Copy	HttpVerb.PUT	N/A	Yes	Yes
DELETE Object	HttpVerb.DELETE	N/A	Yes	Yes

Operation	HTTP Request Method (Value in OBS .NET SDK)	Sub-resource (Value in OBS .NET SDK)	Bucket Name Required	Object Name Required
DELETE Objects	HttpVerb.POST	SubResourceEnum.Delete	Yes	Yes
Obtain Object Metadata	HttpVerb.HEAD	N/A	Yes	Yes
PUT Object ACL	HttpVerb.PUT	SubResourceEnum.Acl	Yes	Yes
GET Object ACL	HttpVerb.GET	SubResourceEnum.Acl	Yes	Yes
Initiate Multipart Upload	HttpVerb.POST	SubResourceEnum.Uploads	Yes	Yes
Upload Part	HttpVerb.PUT	N/A	Yes	Yes
Upload Part - Copy	HttpVerb.PUT	N/A	Yes	Yes
List Parts	HttpVerb.GET	N/A	Yes	Yes
Complete Multipart Upload	HttpVerb.POST	N/A	Yes	Yes
Abort Multipart Upload	HttpVerb.DELETE	N/A	Yes	Yes
POST Object restore	HttpVerb.POST	SubResourceEnum.Restore	Yes	Yes

To access OBS using a temporary URL generated by the OBS .NET SDK, perform the following steps:

Step 1 Call `ObsClient.CreateTemporarySignature` to generate a signed URL.

Step 2 Use any HTTP library to make an HTTP/HTTPS request to OBS.

----End

⚠ CAUTION

If a CORS or signature mismatch error occurs, refer to the following steps to troubleshoot the issue:

1. If CORS is not configured, you need to configure CORS rules on OBS Console. For details, see [Configuring CORS](#).
2. If the signatures do not match, check whether signature parameters are correct by referring to [Authentication of Signature in a URL](#). For example, during an object upload, the backend uses **Content-Type** to calculate the signature and generate an authorized URL, but if **Content-Type** is not set or is set to an incorrect value when the frontend uses the authorized URL, a CORS error occurs. To avoid this issue, ensure that **Content-Type** fields at both the frontend and backend are kept consistent.

The following content provides examples of accessing OBS using a temporary URL, including bucket creation, as well as object upload, download, listing, and deletion.

Creating a Bucket

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKeyID = Environment.GetEnvironmentVariable("AccessKeyID",
    EnvironmentVariableTarget.Machine);
string secretKey = Environment.GetEnvironmentVariable("SecretAccessKey",
    EnvironmentVariableTarget.Machine);
// Create an ObsClient instance.
ObsClient client = new ObsClient(accessKey, secretKey, config);

// Specify the validity period of the URL to 3600 seconds.
long expireSeconds = 3600;

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

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

// Use a PUT request to create a bucket.
HttpRequest webRequest = WebRequest.Create(response.SignUrl) as HttpRequest;
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()));
}
```

Uploading an Object

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an ObsClient instance.
ObsClient client = new ObsClient(accessKey, secretKey, config);

// Specify the validity period of the URL to 3600 seconds.
long expireSeconds = 3600;

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

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

// Use a PUT request to upload an object.
HttpRequest webRequest = WebRequest.Create(response.SignUrl) as HttpRequest;
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);
}

HttpResponse webResponse = null;
try
{
    webResponse = webRequest.GetResponse() as HttpResponse;
}
catch (WebException ex)
{
    webResponse = ex.Response as HttpResponse;
}
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()));
}
```

Downloading an Object

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an ObsClient instance.
ObsClient client = new ObsClient(accessKey, secretKey, config);
```

```
// Specify the validity period of the URL to 3600 seconds.
long expireSeconds = 3600;

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

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

// Make a GET request to download an object.
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()));
}
```

Listing Objects

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an ObsClient instance.
ObsClient client = new ObsClient(accessKey, secretKey, config);
```

```
// Specify the validity period of the URL to 3600 seconds.
long expireSeconds = 3600;

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

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

// Use a GET request to obtain the object list.
HttpRequest webRequest = WebRequest.Create(response.SignUrl) as HttpRequest;
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);
    }
}

HttpResponse webResponse = null;
try
{
    webResponse = webRequest.GetResponse() as HttpResponse;
}
catch (WebException ex)
{
    webResponse = ex.Response as HttpResponse;
}
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()));
}
```

Deleting an Object

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an ObsClient instance.
ObsClient client = new ObsClient(accessKey, secretKey, config);
```

```
// Specify the validity period of the URL to 3600 seconds.
long expireSeconds = 3600;

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

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

// Use a DELETE request to delete an object.
HttpRequest webRequest = WebRequest.Create(response.SignUrl) as HttpRequest;
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);
    }
}

HttpResponse webResponse = null;
try
{
    webResponse = webRequest.GetResponse() as HttpResponse;
}
catch (WebException ex)
{
    webResponse = ex.Response as HttpResponse;
}
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()));
}
```

 **NOTE**

HttpVerb is an enumeration type defined in OBS .NET SDK, whose value indicates the request method types.

11 Versioning Management

11.1 Versioning Overview

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can use versioning to store multiple versions of an object in a bucket.

When versioning is enabled for a bucket, OBS keeps multiple versions of an object in the bucket, allowing you to easily retrieve and restore historical versions or recover data in the event of accidental changes or application failures.

By default, versioning is disabled for new OBS buckets. In this case, if a newly uploaded object is using the name of the previously uploaded one, the new object will overwrite the previous one.

For details, see [Versioning](#).

11.2 Setting Versioning Status for a Bucket

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call **ObsClient.SetBucketVersioning** to set the versioning status for a bucket. OBS supports two versioning statuses.

Versioning Status	Description	Value in OBS .NET SDK
Enabled	<ol style="list-style-type: none"> 1. OBS creates a unique version ID for each uploaded object. Namesake objects are not overwritten and are distinguished by their own version IDs. 2. Objects can be downloaded by specifying the version ID. By default, the latest object is downloaded if no version ID is specified. 3. Objects can be deleted by specifying the version ID. If an object is deleted with no version ID specified, the object will generate a delete marker with a unique version ID but is not physically deleted. 4. Objects of the latest version in a bucket are returned by default after ObsClient.ListObjects is called. You can call ObsClient.ListVersions to list a bucket's objects with all version IDs. 5. Except for delete markers, storage space occupied by objects with all version IDs is billed. 	VersionStatusEnum.Enabled

Versioning Status	Description	Value in OBS .NET SDK
Suspended	<ol style="list-style-type: none"> Existing objects with version IDs are not affected. OBS creates version ID null to an uploaded object and the object will be overwritten after a namesake one is uploaded Objects can be downloaded by specifying the version ID. By default, the latest object is downloaded if no version ID is specified. Objects can be deleted by version ID. If an object is deleted with no version ID specified, the object is only attached with a deletion mark and version ID null. Objects with version ID null are physically deleted. Except for delete markers, storage space occupied by objects with all version IDs is billed. 	VersionStatusEnum.Suspended

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Set the versioning status.
try
{
    SetBucketVersioningRequest request = new SetBucketVersioningRequest();
    request.BucketName = "bucketname";
    request.Configuration = new VersioningConfiguration();
    //Enabling versioning.
    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 Viewing Versioning Status of a Bucket

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call **ObsClient.GetBucketVersioning** to view the versioning status of a bucket.

This example obtains the versioning status of bucket **bucketname**.

The example code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// View the versioning status.
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);
}
```

NOTE

- To handle the error codes possibly returned during the operation, see [OBS Server-Side Error Codes](#).

11.4 Obtaining a Versioning Object

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call **ObsClient.GetObject** to obtain an object version by specifying the version ID (**VersionId**).

This example downloads object **objectname** from bucket **bucketname** by specifying **VersionId**.

The example code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Obtain versioning objects.
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);
        //Save the file locally.
        if (!File.Exists("savePath"))
        {
            response.WriteResponseStreamToFile("savePath");
        }
    }
}
catch (ObsException ex)
{
    Console.WriteLine("ErrorCode: {0}", ex.ErrorCode);
    Console.WriteLine("ErrorMessage: {0}", ex.ErrorMessage);
}
```

NOTE

- If the version ID is null, the object of the latest version will be downloaded, by default.

11.5 Copying a Versioning Object

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call **ObsClient.CopyObject** to copy an object version by specifying the version ID (**SourceVersionId**).

This example specifies **SourceVersionId** to copy **sourceobjectname** from **sourcebucketname** to **destbucketname** as **destobjectname**.

The example code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Copy a versioning object.
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);
}
```

NOTE

- To handle the error codes possibly returned during the operation, see [OBS Server-Side Error Codes](#).

11.6 Restoring a Specific Archive Object Version

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call **ObsClient.RestoreObject** to restore an Archive object version by specifying **VersionId**.

This example specifies **versionId** to restore Archive object **destobjectname** in **destbucketname** as a Standard object.

The example code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
```

```
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Restore a specific Archive object version.
try
{
    RestoreObjectRequest request = new RestoreObjectRequest();
    request.BucketName = "bucketname";
    request.ObjectKey = "objectname";
    request.Days = 5;
    // Restore a versioned object at an expedited speed.
    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);
}
```

CAUTION

To prolong the validity period of the Archive data restored, you can repeatedly restore the data, but you will be billed for each restoration. After a second restore, the validity period of Standard object copies will be prolonged, and you need to pay for storing these copies during the prolonged period.

11.7 Listing Versioning Objects

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call **ObsClient.ListVersions** to list versioning objects.

The following table describes the parameters involved in this API.

Parameter	Description	Property in OBS .NET SDK
BucketName	Bucket name	ListVersionsRequest.BucketName

Parameter	Description	Property in OBS .NET SDK
Prefix	Name prefix that the objects to be listed must contain	ListVersionsRequest.Prefix
KeyMarker	Object name to start with when listing versioning objects in a bucket. All versioning objects following this parameter are listed in the lexicographical order.	ListVersionsRequest.KeyMarker
MaxKeys	Maximum number of listed versioning objects. The value ranges from 1 to 1000 . If the specified value exceeds 1000 , only 1,000 versioning objects are returned by default.	ListVersionsRequest.MaxKeys
Delimiter	Character used to group object names. If the object name contains the Delimiter parameter, the character string from the first character to the first delimiter in the object name is grouped under a single result element, CommonPrefix . (If a prefix is specified in the request, the prefix must be removed from the object name.)	ListVersionsRequest.Delimiter
VersionIdMarker	Object name to start with when listing versioning objects in a bucket. All versioning objects are listed in the lexicographical order by object name and version ID. This parameter must be used together with KeyMarker .	ListVersionsRequest.VersionIdMarker

 NOTE

- If the value of **VersionIdMarker** is not a version ID specified by **KeyMarker**, **VersionIdMarker** is ineffective.
- The returned result of **ObsClient.ListVersions** includes the versioning objects and delete markers.

Listing Versioning Objects in Simple Mode

The following sample code shows how to list versioning objects in simple mode. A maximum of 1000 versioning objects can be returned.

```
// Initialize configuration parameters.  
ObsConfig config = new ObsConfig();  
config.Endpoint = "https://your-endpoint";
```

```
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
//List versioning objects.
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);
}
```

NOTE

Information about a maximum of 1000 versioning objects can be listed each time. If a bucket contains more than 1000 objects and **ListVersionsResponse.IsTruncated** is **true** in the returned result, not all versioning objects are listed. In such cases, you can use **ListVersionsResponse.NextKeyMarker** and **ListVersionsResponse.NextVersionIdMarker** to obtain the start position for next listing.

Listing Versioning Objects by Specifying the Number

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
//List versioning objects by specifying the number.
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);
}
```

Listing Versioning Objects by Specifying a Prefix

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
//List versioning objects by specifying a prefix.
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);
}
```

Listing Versioning Objects by Specifying the Start Position

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
```

```
ObsClient client = new ObsClient(accessKey, secretKey, config);
//Specify the start position for listing.
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);
}
```

Listing All Versioning Objects in Paging Mode

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
//List all versioning objects in paging mode.
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);
}
```

Listing All Versioning Objects in a Folder

There is no folder concept in OBS. All elements in buckets are objects. Folders are actually objects whose sizes are 0 and whose names end with a slash (/). When you set a folder name as the prefix, objects in this folder will be listed. Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
Environment.VariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
Environment.VariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
//List all versioning objects in paging mode.
try
{
    ListVersionsRequest request = new ListVersionsRequest();
    request.BucketName = "bucketname";
    request.MaxKeys = 1000;
    // Set the prefix to dir/.
    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 Setting or Obtaining a Versioning Object ACL

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

Setting an ACL for an Object Version

You can call **ObsClient.SetObjectAcl** to pass the version ID (**VersionId**) to set an ACL for an object version. Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Set an ACL for an object version.
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);
}
```

NOTE

The owner or grantee ID needed in the ACL indicates the account ID, which can be viewed on the **My Credentials** page of OBS Console.

Obtaining the ACL of an Object Version

You can call **ObsClient.GetObjectAcl** to pass the version ID (**VersionId**) to obtain the ACL of an object version. Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
```

```
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Obtain the ACL of a versioning object.
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 Deleting Versioning Objects

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

Deleting a Single Versioning Object

You can call **ObsClient.DeleteObject** to pass a version ID (**VersionId**) to delete an object version.

This example deletes object **objectname** from bucket **bucketname** by specifying **VersionId**.

The example code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
Environment.VariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
```

```
EnvironmentVariableTarget.Machine);  
// Create an instance of ObsClient.  
ObsClient client = new ObsClient(accessKey, secretKey, config);  
// Delete a versioning object.  
try  
{  
    DeleteObjectRequest request = new DeleteObjectRequest()  
    {  
        BucketName = "bucketname",  
        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);  
}
```

Deleting Versioning Objects in a Batch

You can call **ObsClient.DeleteObjects** to batch delete specific versions of an object by passing the **VersionId** value of each version to delete.

This example deletes objects **objectname1** and **objectname2** from bucket **bucketname** in a batch by specifying their version IDs.

The example code is as follows:

```
// Initialize configuration parameters.  
ObsConfig config = new ObsConfig();  
config.Endpoint = "https://your-endpoint";  
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the  
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables  
// for identity authentication. Before running this example, configure environment variables AccessKeyID and  
// SecretAccessKey.  
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.  
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",  
EnvironmentVariableTarget.Machine);  
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",  
EnvironmentVariableTarget.Machine);  
// Create an instance of ObsClient.  
ObsClient client = new ObsClient(accessKey, secretKey, config);  
// Delete two versioning objects.  
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 Lifecycle Management

12.1 Lifecycle Management Overview

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

OBS allows you to set lifecycle rules for buckets to automatically transition the storage class of an object and delete expired objects, to effectively use storage features and optimize the storage space. You can set multiple lifecycle rules to match different object name prefixes. A lifecycle rule must contain:

- Rule ID, which uniquely identifies the rule
- A shared prefix of object names that are under the control of this rule
- Transition policy of an object of the latest version, which can be specified in either mode:
 - a. How many days after the object is created
 - b. Transition date
- Expiration time of the latest version of an object, which can be specified in either of the following method:
 - a. How many days after the object is created
 - b. Expiration date
- Transition policy of a noncurrent object version, which can be specified in the following mode:
 - How many days after the object becomes a noncurrent object version
- Expiration time of a noncurrent object version, which can be specified in the following mode:
 - How many days after the object becomes a noncurrent object version
- Identifier specifying whether the setting is effective

For more information, see [Lifecycle Management](#).

NOTE

- An object will be automatically deleted by the OBS server once it expires.
- The time set in the transition policy of an object must be earlier than its expiration time, and the time set in the transition policy of a noncurrent object version must be earlier than its expiration time.
- The configured expiration time and transition policy for a noncurrent object version will take effect only when the versioning is enabled or suspended for a bucket.

12.2 Setting Lifecycle Rules

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call `ObsClient.SetBucketLifecycle` to set lifecycle rules.

Setting an Object Transition Policy

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
//Set an object transition policy.
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);
    // Specify that objects whose names contain the specified prefix will be transitioned 30 days after
creation.
    transition.Days = 30;
    // Specify the storage class that the object will be transitioned to.
    transition.StorageClass = StorageClassEnum.Warm;
    NoncurrentVersionTransition noncurrentVersionTransition = new NoncurrentVersionTransition();
    rule1.NoncurrentVersionTransitions.Add(noncurrentVersionTransition);
    // Specify that objects whose names contain the specified prefix will be transitioned 60 days after
```

```
becoming noncurrent versions.
    noncurrentVersionTransition.NoncurrentDays = 60;
    // Specify the storage class that a noncurrent version will be transitioned to.
    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);
}
```

Setting an Object Expiration Time

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Set the object expiration time.
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();
    //Specify that objects whose names contain the prefix will expire 60 days after creation.
    rule1.Expiration.Days = 60;
    // Specify that objects whose names contain the prefix will expire 60 days after becoming noncurrent
versions.
    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);
}
```

NOTE

- Each time you call this API to configure a lifecycle rule, the lifecycle rules you previously configured will be overwritten.

12.3 Viewing Lifecycle Rules

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call **ObsClient.GetBucketLifecycle** to view lifecycle rules.

This example views the lifecycle configuration of bucket **bucketname**.

The example code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
Environment.VariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
Environment.VariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// View the bucket lifecycle.
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);
}
```

NOTE

- To handle the error codes possibly returned during the operation, see [OBS Server-Side Error Codes](#).

12.4 Deleting Lifecycle Rules

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call **ObsClient.DeleteBucketLifecycle** to delete lifecycle rules.

This example deletes the lifecycle configuration of bucket **bucketname**.

The example code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Delete bucket lifecycles.
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);
}
```

NOTE

- To handle the error codes possibly returned during the operation, see [OBS Server-Side Error Codes](#).

13 CORS

13.1 CORS Overview

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

Cross-origin access refers to access between different domains. Restricting cross-origin access is a browser policy for security purposes, that is, the same-origin policy.

Due to this same-origin policy, JavaScript in origin A cannot operate objects in origin B or C.

The same-origin policy requires the protocols, domain names (or IP addresses), and ports are all the same. If the protocols, domain names, and ports (if specified) of the two web pages are the same, the two web pages are in the same origin.

CORS allows web application programs in one origin to access resources in another. OBS provides developers with APIs for facilitating cross-origin resource access.

For more information, see [CORS](#).

13.2 Setting CORS Rules

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call **ObsClient.SetBucketCors** to set CORS rules for a bucket. If the bucket is configured with CORS rules, the newly set ones will overwrite the existing ones.

This example sets a CORS rule for bucket **bucketname**.

The example code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Set the CORS rules for a bucket.
try
{
    SetBucketCorsRequest request = new SetBucketCorsRequest();
    request.BucketName = "bucketname";
    request.Configuration = new CorsConfiguration();
    CorsRule rule = new CorsRule();
    rule.Id = "20170820";
    // Specify the origin of the cross-domain request.
    rule.AllowedOrigins.Add("http://www.a.com");
    rule.AllowedOrigins.Add("http://www.b.com");
    // Specify whether headers specified in Access-Control-Request-Headers in the OPTIONS request can
be used.
    rule.AllowedHeaders.Add("x-obs-header");
    // Specify the request method, which can be GET, PUT, DELETE, POST, or 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);
    // Specify response headers that users can access using application programs.
    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);
}
```

NOTE

- **AllowedOrigins**, **AllowedMethods**, and **AllowedHeaders**, respectively, can contain up to one wildcard character (*). Wildcard characters (*) indicate that all origins, operations, or headers are allowed.

13.3 Viewing CORS Rules

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call **ObsClient.GetBucketCors** to view CORS rules of a bucket.

This example views the CORS rule of bucket **bucketname**.

The example code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// View CORS rules.
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);  
}
```

NOTE

- To handle the error codes possibly returned during the operation, see [OBS Server-Side Error Codes](#).

13.4 Deleting CORS Rules

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call **ObsClient.DeleteBucketCors** to delete CORS rules of a bucket.

This example deletes the CORS rule of bucket **bucketname**.

The example code is as follows:

```
// Initialize configuration parameters.  
ObsConfig config = new ObsConfig();  
config.Endpoint = "https://your-endpoint";  
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the  
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables  
// for identity authentication. Before running this example, configure environment variables AccessKeyID and  
SecretAccessKey.  
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.  
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",  
EnvironmentVariableTarget.Machine);  
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",  
EnvironmentVariableTarget.Machine);  
// Create an instance of ObsClient.  
ObsClient client = new ObsClient(accessKey, secretKey, config);  
// Delete CORS rules.  
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);  
}
```

NOTE

- To handle the error codes possibly returned during the operation, see [OBS Server-Side Error Codes](#).

14 Access Logging

14.1 Logging Overview

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

OBS allows you to configure access logging for buckets. After the configuration, access to buckets will be recorded in the format of logs. These logs will be saved in specific buckets in OBS.

You can enable OBS logging for bucket analysis or audit purposes. With access logs, a bucket owner can analyze the characteristics, types, or trends of requests sent to the bucket.

With logging enabled, OBS automatically logs access requests for the bucket and writes the generated log files into a specified bucket.

You need to specify a bucket for storing log files when enabling logging for a bucket. Log files can be stored in any bucket (including the logged bucket itself) you own in the region where the logged bucket is.

For more information, see [Logging](#).

14.2 Enabling Bucket Logging

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call **ObsClient.SetBucketLogging** to enable bucket logging

NOTICE

The source bucket and target bucket of logging must be in the same region.

NOTE

If the bucket is in the OBS Infrequent Access or Archive storage class, it cannot be used as the target bucket.

The **Agency** field indicates the name of the IAM agency for OBS created by the owner of the target bucket. For details about how to create an IAM agency, see the *IAM User Guide*.

Enabling Bucket Logging

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
try
{
    // Set the source bucket 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 Viewing Bucket Logging

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call **ObsClient.GetBucketLogging** to view the logging configuration of a bucket.

This example views the logging configuration of bucket **bucketname**.

The example code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// View bucket logging.
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);
}
```

NOTE

- To handle the error codes possibly returned during the operation, see [OBS Server-Side Error Codes](#).

14.4 Disabling Bucket Logging

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

To disable logging for a bucket is to call **ObsClient.SetBucketLogging** to delete the logging configuration.

This example disables logging for bucket **bucketname**.

The example code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
```

```
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
try
{
    SetBucketLoggingRequest putrequest = new SetBucketLoggingRequest();
    putrequest.BucketName = "bucketname";//Source bucket
    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);
}
```

 **NOTE**

- To handle the error codes possibly returned during the operation, see [OBS Server-Side Error Codes](#).

15 Static Website Hosting

15.1 Static Website Hosting Overview

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

Static websites typically only contain static web pages and some scripts that can run on clients (such as JavaScript and Flash). In contrast, dynamic websites depend on scripts that need to be processed on the server side, including PHP, JSP, and ASP.Net.

To host your static website on OBS, you can upload static website files to your bucket as objects, configure the public read permission for the objects, and then configure static website hosting for your bucket.

After this, when third-party users access your websites, they actually access the objects in your bucket in OBS.

When using static website hosting, you can configure request redirection to redirect specific or all requests.

For more information, see [Static Website Hosting](#).

15.2 Website File Hosting

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can perform the following to implement website file hosting:

- Step 1** Upload the website files to your bucket in OBS as objects and set the MIME type for the objects.
- Step 2** Set the object ACL to public read.
- Step 3** Access the objects using a browser.

----End

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
try
{
    //Set the MIME type for an object.
    PutObjectRequest request = new PutObjectRequest
    {
        BucketName = "bucketname",
        ObjectKey = "test.html",
        FilePath = "localfile.html", // Path of the local file uploaded. The file name must be specified.
        CannedAcl = CannedAclEnum.PublicRead, //Set the object ACL to public read.
        ContentType = "text/html",
    };
    // Upload an object.
    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);
}
```

NOTE

You can use <https://bucketname.your-endpoint/test.html> in a browser to access files hosted using the sample code.

15.3 Setting Website Hosting

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call `ObsClient.SetBucketWebsite` to set website hosting on a bucket.

Configuring the Default Homepage and Error Pages

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
//Configure the homepage and error pages.
try
{
    SetBucketWebsiteRequest request = new SetBucketWebsiteRequest();
    request.BucketName = "bucketname";
    request.Configuration = new WebsiteConfiguration();
    //Configure the default homepage.
    request.Configuration.IndexDocument = "index.html";
    //Configure the error pages.
    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);
}
```

Configuring the Redirection Rules

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Set the redirection rule.
try
{
    SetBucketWebsiteRequest request = new SetBucketWebsiteRequest();
    request.BucketName = "bucketname";
    request.Configuration = new WebsiteConfiguration(); //Configure the default homepage.
    request.Configuration.IndexDocument= "index.html";
    //Configure the error pages.
}
```

```
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);
}
```

Configuring Redirection for All Requests

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Configure redirection for all requests.
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 Viewing Website Hosting Settings

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call **ObsClient.GetBucketWebsite** to view the hosting settings of a bucket.

This example views the hosting configuration of bucket **bucketname**.

The example code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// View hosting settings.
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);
}
```

NOTE

- To handle the error codes possibly returned during the operation, see [OBS Server-Side Error Codes](#).

15.5 Deleting Website Hosting Settings

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call **ObsClient.DeleteBucketWebsite** to delete the hosting settings of a bucket.

This example deletes the hosting configuration of bucket **bucketname**.

The example code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Delete hosting settings.
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);
}
```

NOTE

- To handle the error codes possibly returned during the operation, see [OBS Server-Side Error Codes](#).

16 Tag Management

16.1 Tagging Overview

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

Tags are used to identify and classify OBS buckets.

16.2 Setting Bucket Tags

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call **ObsClient.SetBucketTagging** to set bucket tags. Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
```

```
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Set bucket tags.
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);
}
```

NOTE

- A bucket can have up to 10 tags.
- The key and value pair of a tag can be composed of Unicode characters.

16.3 Viewing Bucket Tags

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call **ObsClient.GetBucketTagging** to view bucket tags. Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Obtain bucket tags.
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 Deleting Bucket Tags

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

You can call **ObsClient.DeleteBucketTagging** to delete bucket tags. Sample code is as follows:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
Environment.VariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
Environment.VariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Delete bucket tags.
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 Server-Side Encryption

17.1 Server-Side Encryption Overview

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

OBS supports server-side encryption.

For more information, see [Server-Side Encryption](#).

17.2 Encryption Description

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

The following table lists APIs related to server-side encryption:

OBS .NET SDK API Function	Description	Supported Encryption Type
ObsClient.PutObject	Sets the encryption algorithm and key during object upload to enable server-side encryption.	SSE-KMS SSE-C

OBS .NET SDK API Function	Description	Supported Encryption Type
ObsClient.GetObject	Sets the decryption algorithm and key during object download to decrypt the object.	SSE-C
ObsClient.CopyObject	<ol style="list-style-type: none">1. Sets the decryption algorithm and key for decrypting the source object during object copy.2. Sets the encryption algorithm and key during object copy to enable the encryption algorithm for the target object.	SSE-KMS SSE-C
ObsClient.GetObjectMetadata	Sets the decryption algorithm and key when obtaining the object metadata to decrypt the object.	SSE-C
ObsClient.InitiateMultipartUpload	Sets the encryption algorithm and key when initializing a multipart upload to enable server-side encryption for the final object generated.	SSE-KMS SSE-C
ObsClient.UploadPart	Sets the encryption algorithm and key during multipart upload to enable server-side encryption for parts.	SSE-C
ObsClient.CopyPart	<ol style="list-style-type: none">1. Sets the decryption algorithm and key for decrypting the source object during partial object copy.2. Sets the encryption algorithm and key during partial object copy to enable the encryption algorithm for the target object part.	SSE-C

17.3 Example of Encryption

NOTICE

If you have any questions during the development, post them on the [Issues](#) page of GitHub. For details about parameters and usage of each API, see the [API Reference](#).

Encrypting an Object to Be Uploaded

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Generate an encryption key.
System.Security.Cryptography.Aes aesEncryption = System.Security.Cryptography.Aes.Create();
aesEncryption.KeySize = 256;
aesEncryption.GenerateKey();
string customerkey = Convert.ToBase64String(aesEncryption.Key);
// Configure the local file to be encrypted before an upload based on the site requirements.
string filePathKms = "D:\\test\\testSseC.zip";
string filePathSseC = "D:\\test\\testSseC.zip";
// Upload the object.
try
{
    // When uploading an object, use the SSE-KMS algorithm to encrypt the object.
    SseKmsHeader kms = new SseKmsHeader();
    kms.Algorithm = SseKmsAlgorithmEnum.Kms;
    PutObjectRequest request1 = new PutObjectRequest
    {
        BucketName = "bucketname",
        ObjectKey = "objectname1",
        FilePath = filePathKms,
        SseHeader = kms,
    };
    client.PutObject(request1);
    // When uploading an object, use the SSE-C algorithm to encrypt the object.
    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);
}
```

Decrypting a To-Be-Download Object

Sample code:

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
// configuration file or environment variables. In this example, the AK/SK are stored in environment variables
// for identity authentication. Before running this example, configure environment variables AccessKeyID and
// SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/
```

```
intl/en-us/usermanual-ca/ca_01_0003.html
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Download an object.
try
{
    // When downloading an object, use the SSE-C algorithm to decrypt it.
    GetObjectRequest request = new GetObjectRequest
    {
        BucketName = "bucketname",
        ObjectKey = "objectname2",
        // The key used here must be the one used for uploading the object.
        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 Troubleshooting

18.1 OBS Server-Side Error Codes

If the OBS server encounters an error when processing a request, a response containing the error code and error description is returned. The following table lists details about each error code and HTTP status code.

Error Code	Description	HTTP Status Code
AccessDenied	Access denied.	403 Forbidden
AccessForbidden	Insufficient permission.	403 Forbidden
AccountProblem	Your account is abnormal (for example, being expired or frozen).	403 Forbidden
AllAccessDisabled	The user has no permission to perform a specific operation.	403 Forbidden
AmbiguousGrantByEmailAddress	The provided email address is associated with more than one account.	400 Bad Request
BadDigest	The specified value of Content-MD5 does not match the value received by OBS.	400 Bad Request
BadDomainName	Invalid domain name.	400 Bad Request
BadRequest	Invalid request parameter.	400 Bad Request

Error Code	Description	HTTP Status Code
BucketAlreadyExists	The requested bucket name already exists. The bucket namespace is shared by all users of OBS. Try again with another bucket name.	409 Conflict
BucketAlreadyOwnedByYou	Your previous request for creating the named bucket succeeded and you already own it.	409 Conflict
BucketNotEmpty	The bucket that you tried to delete is not empty.	409 Conflict
CredentialsNotSupported	This request does not support security credentials.	400 Bad Request
CustomDomainAlreadyExist	The configured domain already exists.	400 Bad Request
CustomDomainNotExist	The domain to be operated does not exist.	400 Bad Request
DeregisterUserId	The user has been deregistered.	403 Forbidden
EntityTooSmall	The size of the object to be uploaded is smaller than the lower limit.	400 Bad Request
EntityTooLarge	The size of the object to be uploaded exceeds the limit.	400 Bad Request
FrozenUserId	The user has been frozen.	403 Forbidden
IllegalVersioningConfigurationException	Invalid versioning configuration in the request	400 Bad Request
IllegalLocationConstraintException	The configured region limitation is inconsistent with the region where it resides.	400 Bad Request
InArrearOrInsufficientBalance	The subscriber owes fees or the account balance is insufficient, and the subscriber does not have the permission to perform an operation.	403 Forbidden

Error Code	Description	HTTP Status Code
IncompleteBody	Incomplete request body.	400 Bad Request
IncorrectNumberOfFilesInPostRequest	Each POST request must contain one file to be uploaded.	400 Bad Request
InlineDataTooLarge	The size of inline data exceeds the upper limit.	400 Bad Request
InsufficientStorageSpace	Insufficient storage space.	403 Forbidden
InternalServerError	An internal error occurs. Try again later.	500 Internal Server Error
InvalidAccessKeyId	The access key ID provided by the customer does not exist in the system.	403 Forbidden
InvalidAddressingHeader	The anonymous role must be specified.	N/A
InvalidArgument	Invalid parameter.	400 Bad Request
InvalidBucketName	The specified bucket name in the request is invalid.	400 Bad Request
InvalidBucket	The bucket to be accessed does not exist.	400 Bad Request
InvalidBucketState	Invalid bucket status.	409 Conflict
InvalidBucketStoragePolicy	An invalid new policy is specified during bucket policy modification.	400 Bad Request
InvalidDigest	The specified Content-MD5 in the HTTP header is invalid.	400 Bad Request
InvalidEncryptionAlgorithmError	Incorrect encryption algorithm.	400 Bad Request
InvalidLocationConstraint	The location specified during bucket creation is invalid.	400 Bad Request
InvalidPart	One or more specified parts are not found. The parts may not be uploaded or the specified entity tags (ETags) do not match the parts' ETags.	400 Bad Request

Error Code	Description	HTTP Status Code
InvalidPartOrder	Parts are not listed in ascending order by part number.	400 Bad Request
InvalidPayer	All accesses to this object are disabled.	403 Forbidden
InvalidPolicyDocument	The content of the form does not meet the conditions specified in the policy document.	400 Bad Request
InvalidRange	The requested range cannot be obtained.	416 Client Requested Range Not Satisfiable
InvalidRedirectLocation	Invalid redirect location.	400 Bad Request
InvalidRequest	Invalid request.	400 Bad Request
InvalidRequestBody	Invalid POST request body.	400 Bad Request
InvalidSecurity	Invalid security credentials.	403 Forbidden
InvalidStorageClass	The specified storage class is invalid.	400 Bad Request
InvalidTargetBucketForLogging	The delivery group has no ACL permission for the target bucket.	400 Bad Request
InvalidURI	The specified URI cannot be resolved.	400 Bad Request
KeyTooLong	The provided key is too long.	400 Bad Request
MalformedACLError	The provided XML file has syntax errors or does not meet the format requirements.	400 Bad Request
MalformedError	The XML format in the request is incorrect.	400 Bad Request
MalformedLoggingStatus	The XML format of Logging is incorrect.	400 Bad Request
MalformedPolicy	The bucket policy does not pass.	400 Bad Request

Error Code	Description	HTTP Status Code
MalformedPOSTRequest	The body of the POST request is in an incorrect format.	400 Bad Request
MalformedQuotaError	The Quota XML format is incorrect.	400 Bad Request
MalformedXML	An XML file of a configuration item is in incorrect format. The error message states: "The XML you provided was not well-formed or did not validate against our published schema."	400 Bad Request
MaxMessageLengthExceeded	The request is too long.	400 Bad Request
MaxPostPreDataLengthExceeded Error	The POST request fields prior to the file to be uploaded are too large.	400 Bad Request
MetadataTooLarge	The size of the metadata header has exceeded the upper limit.	400 Bad Request
MethodNotAllowed	The specified method is not allowed against the requested resource. The message "Specified method is not supported." is returned.	405 Method Not Allowed
MissingContentLength	The HTTP header Content-Length is not provided.	411 Length Required
MissingRegion	The region information is missing in the request, and the default region is required in the system.	400 Bad Request
MissingRequestBodyError	An empty XML file is sent as a request. The error message states: "Request body is empty."	400 Bad Request
MissingRequiredHeader	A required header is missing in the request.	400 Bad Request
MissingSecurityHeader	A required header is missing in the request.	400 Bad Request

Error Code	Description	HTTP Status Code
NoSuchBucket	The bucket does not exist.	404 Not Found
NoSuchBucketPolicy	No bucket policy exists.	404 Not Found
NoSuchCORSConfiguration	No CORS configuration exists.	404 Not Found
NoSuchCustomDomain	The requested user domain does not exist.	404 Not Found
NoSuchKey	The specified key does not exist.	404 Not Found
NoSuchLifecycleConfiguration	The requested Lifecycle does not exist.	404 Not Found
NoSuchPolicy	The specified policy name does not exist.	404 Not Found
NoSuchUpload	The specified multipart upload does not exist. The upload ID does not exist or the multipart upload job has been aborted or completed.	404 Not Found
NoSuchVersion	The specified version ID does not match any existing version.	404 Not Found
NoSuchWebsiteConfiguration	The requested website does not exist.	404 Not Found
NotImplemented	The provided header implies a function that is unavailable.	501 Not Implemented
NotSignedUp	Your account is not signed up in the system. Sign up the account before use.	403 Forbidden
OperationAborted	A conflicting operation is being performed on this resource. Try again later.	409 Conflict
PermanentRedirect	The requested bucket has been permanently redirected to a new URL. All future requests must be sent to the new URL.	301 Moved Permanently
PreconditionFailed	At least one of the specified preconditions is not met.	412 Precondition Failed

Error Code	Description	HTTP Status Code
Redirect	The request is temporarily redirected.	307 Moved Temporarily
RequestIsNotMultiPartContent	A bucket POST request must contain an enclosure-type multipart or the form-data.	400 Bad Request
RequestTimeout	No read or write operation has been performed within the timeout period of the socket connection between the user and the server.	400 Bad Request
RequestTimeTooSkewed	The request time and the server's time differ a lot.	403 Forbidden
RequestTorrentOfBucketError	Requesting the bucket's torrent file is not allowed.	400 Bad Request
ServiceNotImplemented	The request method is not implemented by the server.	501 Not Implemented
ServiceNotSupported	The request method is not supported by the server.	409 Conflict
ServiceUnavailable	The server is overloaded or has internal errors.	503 Service Unavailable
SignatureDoesNotMatch	The provided signature in the request does not match the signature calculated by OBS. Check your AK and SK and signature calculation method.	403 Forbidden
SlowDown	Reduce your request frequency.	503 Service Unavailable
System Capacity Not enough	Insufficient system space.	403 Forbidden
TooManyCustomDomains	Too many user domains are configured.	400 Bad Request
TemporaryRedirect	The request is redirected to the bucket while the domain name server (DNS) is being updated.	307 Moved Temporarily
TooManyBuckets	You have attempted to create more buckets than allowed.	400 Bad Request

Error Code	Description	HTTP Status Code
TooManyObjectCopied	The number of copies of a single object exceeds the upper limit.	400 Bad Request
TooManyWrongSignature	The request is rejected due to high-frequency errors.	400 Bad Request
UnexpectedContent	This request does not support fields with content.	400 Bad Request
UnresolvableGrantByEmailAddress	The provided email address does not match any recorded accounts.	400 Bad Request
UserKeyMustBeSpecified	The user's AK is not carried in the request.	400 Bad Request
WebsiteRedirect	The website request lacks bucketName .	301 Moved Permanently
KMS.DisabledException	The master key is disabled in the SSE-KMS mode.	400 Bad Request
KMS.NotFoundException	The customer master key (CMK) does not exist in SSE-KMS mode.	400 Bad Request
RestoreAlreadyInProgress	The Archive object is being restored. The request conflicts with another one.	409 Conflict
ObjectHasAlreadyRestored	The objects have been restored and the retention period of the objects cannot be shortened.	409 Conflict
InvalidObjectState	The restored object is not an Archive object.	403 Forbidden
InvalidTagError	An invalid tag is provided when configuring the bucket tag.	400 Bad Request
NoSuchTagSet	The specified bucket does not have a tag.	404 Not Found

18.2 Log Analysis

Log Path

Log files of OBS .NET SDK are saved in the path, usually the save directory of executable files of the project, specified in configuration file **Log4Net.config**.

Log Level

When current logs cannot be used to troubleshoot system faults, you can change the log level to obtain more information. You can obtain the most information in **DEBUG** logs and the least information in **ERROR** logs.

Log level description:

- **DEBUG**: debugging level
- **INFO**: information level
- **WARN**: warning level
- **ERROR**: error level

Analysis Method

To analyze and locate a problem or view the running status, you can find the related log file based on the date and view the running status or view logs in the log file for troubleshooting.

The procedure is as follows:

Step 1 Find related logs.

Find related API logs in the log file directory based on the time when the exception occurred and the operation. The following is an example of error logs:

```
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/
LMHua8qpm49geWphVJI6I2mnnzUIYwQwHAuzJw/kmV+O4ilcf0GRR
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
```

Step 2 Analyze the cause based on the error logs.

Assuming that the error code you obtain from the log file is **NoSuchBucket**, you can check it from the table in **OBS Server-Side Error Codes** and know the error information is "The specified bucket does not exist."

----End

18.3 SDK Custom Exceptions

SDK custom exceptions (**ObsException**) are thrown by **ObsClient**. Exceptions are usually OBS server-side errors, including **OBS error codes** and error information and aim to help users locate problems and troubleshoot faults.

ObsException contains the following error information:

- **ObsException.StatusCode**: HTTP status code
- **ObsException.ErrorCode**: OBS server-side error code
- **ObsException.ErrorMessage**: Error description returned by the OBS server
- **ObsException.RequestId**: Request ID returned by the OBS server
- **ObsException.HostId**: Requested server ID

18.4 SDK Common Response Headers

After you call an API in an instance of **ObsClient**, a sub-class of **ObsWebServiceResponse** will be returned.

It includes HTTP/HTTPS response headers.

This example obtains **request-id** from the common response headers when creating a bucket.

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);
// Create a bucket.
try
{
    CreateBucketRequest request = new CreateBucketRequest()
    {
        BucketName = "bucketname",
    };
    ObsWebServiceResponse response = client.CreateBucket(request);
    // Obtain the request ID from the common response headers.
    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 FAQ

19.1 What Do I Do If the Process Hangs Occasionally?

Symptom: The process occasionally hangs when you invoke .NET SDK methods.

Solution: Add "using" before the method.

See the example below.

```
// Initialize configuration parameters.
ObsConfig config = new ObsConfig();
config.Endpoint = "https://your-endpoint";
// Hard-coded or plaintext AK/SK are risky. For security purposes, encrypt your AK/SK and store them in the
configuration file or environment variables. In this example, the AK/SK are stored in environment variables
for identity authentication. Before running this example, configure environment variables AccessKeyID and
SecretAccessKey.
// Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/intl/en-us/usermanual-ca/ca\_01\_0003.html.
string accessKey= Environment.GetEnvironmentVariable("AccessKeyID",
EnvironmentVariableTarget.Machine);
string secretKey= Environment.GetEnvironmentVariable("SecretAccessKey",
EnvironmentVariableTarget.Machine);
// Create an instance of ObsClient.
ObsClient client = new ObsClient(accessKey, secretKey, config);

try
{
    GetObjectMetadataRequest request = new GetObjectMetadataRequest();
    // Specify a bucket name.
    request.BucketName = "bucketname";
    // Specify an object (example/objectname is an example here).
    request.ObjectKey = "example/objectname";
    // Obtain the object metadata.
    using (GetObjectMetadataResponse response = client.GetObjectMetadata(request)) {
        Console.WriteLine("Get object metadata response: {0}", response.StatusCode);
        // Obtain the ETag of the object.
        Console.WriteLine("Object etag {0}: ", response.ETag);
        // Obtain the object version ID.
        Console.WriteLine("Object versionId {0}: ", response.VersionId);
        // Obtain the length of the object data, in bytes.
        Console.WriteLine("Object contentLength {0}: ", response.ContentLength);
    }
}
catch (ObsException ex)
{
```

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