Object Storage Service

Node.js SDK Developer Guide

Issue 01

Date 2023-03-14

HUAWEI TECHNOLOGIES CO., LTD.

Security Declaration

Vulnerability

Huawei's regulations on product vulnerability management are subject to the *Vul. Response Process.* For details about this process, visit the following web page:
https://www.huawei.com/en/psirt/vul-response-process
For vulnerability information, enterprise customers can visit the following web page:
https://securitybulletin.huawei.com/enterprise/en/security-advisory

Contents

1 SDK Download Links

Download Address

- Latest version of OBS Node.js SDK source code: **Download**
- Earlier versions of OBS Node.js SDK: **Download**

Compatibility

- Recommended Node.js versions: Node 0.12.*x*, Node4.*x*, Node6.*x*, Node8.*x*, and Node10.*x*
- Interface functions: Not completely compatible with earlier versions (2.1.*x*). The following table describes the changes.

Interface Function	Description
ObsClient.listBuckets	In the response parameters, the type of the **InterfaceResult.Buckets** field changes to **Array**, and the **InterfaceResult.Buckets.Bucket** field is deleted.
ObsClient.setBucketAcl	In the request parameters, the type of the **Grants** field changes to **Array**, and the **Grants.Grant** field is deleted.
ObsClient.getBucketAcl	In the response parameters, the type of the **InterfaceResult.Grants** field changes to **Array**, and the **InterfaceResult.Grants.Grant** field is deleted.
ObsClient.setObjectAcl	In the request parameters, the type of the **Grants** field changes to **Array**, and the **Grants.Grant** field is deleted.
ObsClient.getObjectAcl	In the response parameters, the type of the **InterfaceResult.Grants** field changes to **Array**, and the **InterfaceResult.Grants.Grant** field is deleted.

Interface Function	Description
ObsClient.setBucketLogging	In the request parameters, the type of the **LoggingEnabled.TargetGrants** field changes to **Array**, and the **LoggingEnabled.TargetGrants.Grant** field is deleted.
ObsClient.getBucketLogging	In the response parameters, the type of the **InterfaceResult.LoggingEnabled.TargetGrants** field changes to **Array**, and the **InterfaceResult.LoggingEnabled.TargetGrants.Grant** field is deleted.
ObsClient.setBucketWebsite	In the request parameters, the type of the **RoutingRules** field changes to **Array**, and the **RoutingRules.RoutingRule** field is deleted.
ObsClient.getBucketWebsite	In the response parameters, the type of the **InterfaceResult.RoutingRules** field changes to **Array**, and the **InterfaceResult.RoutingRules.RoutingRule** field is deleted.
ObsClient.setBucketCors	In the request parameters, the **CorsRule** field is renamed as **CorsRules**.
ObsClient.getBucketCors	In the response parameters, the **InterfaceResult.CorsRule** field is renamed as **InterfaceResult.CorsRules**.
ObsClient.setBucketTagging	In the request parameters, the type of the **TagSet** field changes to **Array**, and the **TagSet.Tag** field is deleted.
ObsClient.getBucketTagging	In the response parameters, the type of the **InterfaceResult.TagSet** field changes to **Array**, and the **InterfaceResult.TagSet.Tag** field is deleted.

2 Quick Start

2.1 Before You Start

NOTICE

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

- Ensure that you are familiar with OBS basic concepts, such as **buckets**, **objects**, and **access keys (AK and SK)**.
- You can see **General Examples of ObsClient** to learn how to call OBS Node.js SDK APIs in a general manner.
- **ObsClient** supports API calling results returned via a callback function or the **Promise** object.
- Some features are available only in some regions. If the HTTP status code of an API is 405, check whether the region supports this feature.

2.2 Setting Up an OBS Environment

NOTICE

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

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**.

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

h. Open the downloaded file **credentials.csv** to obtain the 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

2.3 Preparing a Development Environment

NOTICE

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

- Download the recommended version from the **Node.js's official website** and install it.
- Download the latest version of Eclipse IDE for JavaScript and Web Developer from **Eclipse's official website** and install it.

2.4 Installing the SDK

NOTICE

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

(Recommended) Installing the SDK Using npm

Step 1 Run the **npm -V** command to check the npm version and ensure that the npm is installed.

Step 2 Run the **npm install esdk-obs-nodejs** command to start the installation.

----End

NOTE

- On a Windows operating system, the message "Not internal or external command" is displayed when you run the npm command. In this case, add the npm installation directory (generally the installation directory of Node.js) to the Path environment variable.
- You may need to restart the computer for the environment variables to take effect.
- If you use npm to install dependencies and the network malfunctions, use proxies.

Installing the SDK with the Source Code

The following procedures use OBS Node.js SDK of the latest version as an example.

Step 1 Download the OBS Node.js SDK by referring to **SDK Download Links**.

Step 2 Decompress the development package to obtain folder **examples** (sample code), folder **lib** (SDK source code), file **package.json** (dependency configuration file), and file **README.txt** (feature description file of SDK versions).

Step 3 On the command-line interface (CLI), go to the directory under which the SDK development package is decompressed, and run the **npm install** command to install dependency libraries. The **node_modules** folder will be generated.

Step 4 (Optional) In the Eclipse JavaScript project, import the source code: Open Eclipse JavaScript IDE and choose **Import** > **Projects from Folder or Archive**. For **Import source**, select the directory under which the SDK development package is decompressed.

----End

NOTE

After the installation, the directory structure is similar to the following:

```
├── examples
├── lib
├── node_modules
├── package.json
└── README.txt
```

2.5 Initializing an Instance of ObsClient

NOTICE

If you have any questions during 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:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
      //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using 
hard coding may result in leakage.
      //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
      access_key_id: process.env.ACCESS_KEY_ID,
      secret_access_key: process.env.SECRET_ACCESS_KEY,
      server : 'https://your-endpoint'
});

// Use the instance to access OBS.

// Close the ObsClient instance.
// obsClient.close();
```

CAUTION

- JavaScript is an asynchronous programming language. Therefore, you cannot call the close method when accessing OBS.
- An ObsClient instance cannot be used again after it is closed by calling **obsClient.close**.

NOTE

- For more information, see chapter "Initialization."
- For logging configuration, see **Configuring SDK Logging**.

2.6 Creating a Bucket

NOTICE

If you have any questions during 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:

```
obsClient.createBucket({
       Bucket : 'bucketname',
}, (err, result) => {
       if(err){
              console.error('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

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.
  - 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.

2.7 Uploading an Object

NOTICE

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

Sample code:

```
obsClient.putObject({
       Bucket : 'bucketname',
       Key : 'objectname',
       Body : 'Hello OBS'
}, (err, result) => {
       if(err){
              console.error('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

NOTE

For more information, see **Object Upload Overview**.

2.8 Downloading an Object

NOTICE

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

Sample code:

```
obsClient.getObject({
       Bucket : 'bucketname',
       Key : 'objectname'
}, (err, result) => {
       if(err){
              console.error('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
              if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                     console.log(result.InterfaceResult.Content.toString());
              }
       }
});
```

NOTE

For more information, see **Object Download Overview**.

2.9 Listing Objects

NOTICE

If you have any questions during 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:

```
obsClient.listObjects({
       Bucket : 'bucketname'
}, (err, result) => {
       if(err){
              console.error('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
              if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                     const { Contents = [] } = result.InterfaceResult;
                     for(let i=0;i<Contents.length;i++){
                           console.log('Contents[' + i + ']:');
                           console.log('Key-->' + Contents[i]['Key']);
                           console.log('LastModified-->' + Contents[i]['LastModified']);
                           console.log('Size-->' + result.InterfaceResult.Contents[i]['Size']);
                     }
              }
       }
});
```

NOTE

- In the sample code, 1000 objects will be listed, by default.
- For more information, see **Listing Objects**.

2.10 Deleting an Object

NOTICE

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

Sample code:

```
obsClient.deleteObject({
       Bucket : 'bucketname',
       Key : 'objectname'
}, (err, result) => {
       if(err){
              console.error('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

2.11 General Examples of ObsClient

NOTICE

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

Result Returned via a Callback Function

ObsClient returns the results by using a callback function that contains two parameters in sequence: the exception information parameter and the **SDK common result object** parameter. If the exception information parameter in the callback function is not null, an error occurs during the API calling. Otherwise, the API is called. In such conditions, you need to obtain the HTTP status code from the **SDK common result object** parameter to check whether the operation is successful. Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

// Construct request parameters for bucket operations.
var requestParam1 = {
       Bucket : 'bucketname'
       // Other fields.
};

var callback1 = (err, result) => {
       // Process the result of a bucket-related API call.
};

// Call the APIs for bucket operations, such as creating a bucket.
obsClient.createBucket(requestParam1, callback1);

// Construct request parameters for object operations.
var requestParam2 = {
       Bucket : 'bucketname',
       Key : 'objectname'
       // Other fields.
};

var callback2 = (err, result) => {
       // Process the result of an object-related API call.
};
```

```
// Call an object-related API, such as the API for downloading an object.
obsClient.getObject(requestParam2, callback2);
```

NOTE

For APIs used for bucket operations, the **Bucket** parameter contained in the request object indicates the bucket name. For APIs used for object operations, the **Bucket** and **Key** parameters contained in the request object specify the bucket name and object name, respectively.

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

// Call APIs to perform operations, such as uploading an object.
obsClient.putObject({
       Bucket : 'bucketname',
       Key : 'objectname',
       Body : 'Hello OBS'
}, (err, result) => {
       // If the err parameter is not null, an error occurs during the API calling.
       if(err){
              console.log('Error-->' + err);
       }else{
              // If the exception information is null, the API call is complete. In such conditions, you need to
check the HTTP status code.
              if(result.CommonMsg.Status < 300){// The operation is successful.
                     if(result.InterfaceResult){
                           // Process the business logic after the operation is successful.
                     }
              }else{// The operation fails. Obtain details about the exception.
                     console.log('Code-->' + result.CommonMsg.Code);
                     console.log('Message-->' + result.CommonMsg.Message);
                     console.log('HostId-->' + result.CommonMsg.HostId);
                     console.log('RequestId-->' + result.CommonMsg.RequestId);
              }
       }
});
```

Result Returned via the Promise Object

ObsClient supports results returned via the **Promise** object. If no exception is caught by the **catch** method of the **Promise** object, the API calling is complete. In such conditions, you need to obtain the HTTP status code from the **SDK Common Result Object** to check whether the operation is successful. Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');
// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using 
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

// Construct request parameters for bucket operations.
var requestParam1 = {
       Bucket : 'bucketname'
       // Other fields.
};

// Call the APIs for bucket operations, such as creating a bucket.
var promise1 = obsClient.createBucket(requestParam1);
promise1.then((result) => {
   // Process the API call result.
}).catch((err)=>{
   // Rectify the fault.
});

// Construct request parameters for object operations.
var requestParam2 = {
       Bucket : 'bucketname',
       Key : 'objectname'
       // Other fields.
};

// Call an object-related API, such as the API for downloading an object.
var promise2 = obsClient.getObject(requestParam2);
promise2.then((result) => {
   // Process the API call result.
}).catch((err)=>{
   // Rectify the fault.
});
```

NOTE

For APIs used for bucket operations, the **Bucket** parameter contained in the request object indicates the bucket name. For APIs used for object operations, the **Bucket** and **Key** parameters contained in the request object specify the bucket name and object name, respectively.

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using 
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

// Call APIs to perform operations, such as uploading an object.
obsClient.putObject({
       Bucket : 'bucketname',
       Key : 'objectname',
       Body : 'Hello OBS'
}).then((result) => {
    // If no exception occurs and the API call is complete, check the HTTP status code.
    if(result.CommonMsg.Status < 300){// Operation succeeded
        if(result.InterfaceResult){
            // Process the business logic after the operation is successful.
        }
}else{// The operation fails. Obtain details about the exception.
        console.log('Code-->' + result.CommonMsg.Code);
        console.log('Message-->' + result.CommonMsg.Message);
        console.log('HostId-->' + result.CommonMsg.HostId);
        console.log('RequestId-->' + result.CommonMsg.RequestId);
    }
}).catch((err) => {
    // An exception occurred after the API is called.
    console.error('Error-->' + err);
});
```

2.12 Pre-defined Constants

NOTICE

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

OBS Node.js SDK provides a group of pre-defined constants that can be directly used. You can call **obsClient.enums** or **ObsClient.prototype.enums** to obtain pre-defined constant objects. For more information, see the *Object Storage Service Node.js SDK API Reference*.

3 Initialization

3.1 Configuring the AK and SK

NOTICE

If you have any questions during 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 and SK, you can create an instance of ObsClient to call SDK APIs.

3.2 Creating an Instance of ObsClient

NOTICE

If you have any questions during 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 Node.js 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 Node.js SDK to send a request to OBS, you need to initialize an instance of **ObsClient** and modify parameters related to initial configurations of the instance based on actual needs.

By Using the Constructor

- Sample code for creating an ObsClient instance using permanent access keys (AKs/SKs):

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance by using the constructor.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways.
Using hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

// Use the instance to access OBS.

// Close the ObsClient instance.
// obsClient.close();
```

- Sample code for creating an ObsClient instance using temporary security credentials (AKs/SKs and security tokens):

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance by using the constructor.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways.
Using hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       security_token: process.env.SECURITY_TOKEN,
       server : 'https://your-endpoint'
});

// Use the instance to access OBS.

// Close the ObsClient instance.
// obsClient.close();
```

By Using the Factory Method

- Sample code for creating an ObsClient instance using permanent access keys (AKs/SKs):

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Initialize an ObsClient instance by using the factory method.
var obsClient = new ObsClient();
obsClient.factory({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways.
Using hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       security_token: process.env.SECURITY_TOKEN,
       server : 'https://your-endpoint'
});
```

```
// Use the instance to access OBS.

// Close the ObsClient instance.
// obsClient.close();
```

- Sample code for creating an ObsClient instance using temporary security credentials (AKs/SKs and security tokens):

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Initialize an ObsClient instance by using the factory method.
var obsClient = new ObsClient();
obsClient.factory({
      //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways.
Using hard coding may result in leakage.
      //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
      access_key_id: process.env.ACCESS_KEY_ID,
      secret_access_key: process.env.SECRET_ACCESS_KEY,
      security_token: process.env.SECURITY_TOKEN,

      server : 'https://your-endpoint'
});

// Use the instance to access OBS.

// Close the ObsClient instance.
// obsClient.close();
```

NOTE

- The project can contain one or more ObsClient instances.
- An ObsClient instance cannot be used again after it is closed by calling the close method.

3.3 Configuring an Instance of ObsClient

NOTICE

If you have any questions during 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 the following initialization parameters to configure an instance of **ObsClient**.

Parameter	Description	Recommended Value
access_key_id	AK	N/A
secret_access_key	SK	N/A

Parameter	Description	Recommended Value
server	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
max_retry_count	Maximum number of retries when an HTTP/HTTPS connection is abnormal. The default value is **3**.	[1, 5]
timeout	Timeout period (in seconds) of an HTTP/HTTPS request. The default value is **60**.	[10, 60]
ssl_verify	Whether to verify server-side certificates. Possible values are: ● Path to the server-side root certificate file in **.pem** format ● **true**: The default CAs are used to verify the server-side certificate. ● **false**: The server-side certificates will not be verified. The default value is **false**.	N/A
long_conn_param	Persistent connection mode (in seconds). If the value is equal to or larger than **0**, the persistent connection mode is enabled and this value is used as the **initial delay** of the TCP Keep-Alive packets. By default, this parameter is left blank, which indicates that persistent connection mode is disabled.	N/A
is_cname	Whether to use self-defined domain name to access OBS. The default value is **false**.	N/A

NOTE

- Parameters whose recommended value is **N/A** need to be set according to the actual conditions.
- If the network is unstable, you are advised to set a larger value for **timeout**.
- If the value of **server** does not contain any protocol, HTTPS is used by default.

NOTICE

- If the persistent connection mode is enabled, you must call **ObsClient.close** to close **ObsClient** explicitly to reclaim connection resources.
- For the sake of high DNS resolution performance and OBS reliability, you can set **server** only to the domain name of OBS, instead of the IP address.

3.4 Configuring SDK Logging

NOTICE

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

OBS Node.js SDK provides the logging function based on Log4js. You can call **ObsClient.initLog** to enable and configure logging. Sample code is as follows:

```
obsClient.initLog({
file_full_path:'./logs/OBS-SDK.log', //Set the path to the log file.
      max_log_size:20480, //Set the size of the log file, in bytes.
      backups:10, //Set the maximum number of log files that can be stored.
      level:'warn', //Set the log level.
      log_to_console:true //Set whether to print the log to console.
});
```

NOTE

- The logging function is disabled by default. You need to enable it manually.
- For details about SDK logs, see **Log Analysis**.

4 Bucket Management

4.1 Creating a Bucket

NOTICE

If you have any questions during 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. 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 **Storage Class**. Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

// Create a bucket.
obsClient.createBucket({
       Bucket : 'bucketname',
       // Set the access permission for the bucket to public-read (the default value is private.)
       ACL : obsClient.enums.AclPublicRead,
       //Set the storage class to Archive.
       StorageClass : obsClient.enums.StorageClassStandard,
       // Configure the bucket region.
       Location : 'bucketlocation',


}, (err, result) => {
       if(err){
           console.error('Error-->' + err);
       }else{
           console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

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.
  - 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 where the global domain resides.
- You can use parameter **ACL** to specify the access permission, use **StorageClass** to specify the storage class, and use **Location** to specify the location for 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.

4.2 Listing Buckets

NOTICE

If you have any questions during 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:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using 
hard coding may result in leakage.
```

```
    //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
    access_key_id: process.env.ACCESS_KEY_ID,
    secret_access_key: process.env.SECRET_ACCESS_KEY,
    server : 'https://your-endpoint'
});

// List buckets.
obsClient.listBuckets({
    QueryLocation : true
}, (err, result) => {
    if(err){
        console.error('Error-->' + err);
    }else{
        console.log('Status-->' + result.CommonMsg.Status);
        if(result.CommonMsg.Status < 300 && result.InterfaceResult){
            console.log('RequestId-->' + result.InterfaceResult.RequestId);
            console.log('Owner:');
            console.log('ID-->' + result.InterfaceResult.Owner.ID);
            console.log('Name-->' + result.InterfaceResult.Owner.Name);
            console.log('Buckets:');
             for(let i=0;i<result.InterfaceResult.Buckets.length;i++){
                console.log('Bucket[' + i + ']:');
                console.log('BucketName-->' + result.InterfaceResult.Buckets[i].BucketName);
                console.log('CreationDate-->' + result.InterfaceResult.Buckets[i].CreationDate);
                console.log('Location-->' + result.InterfaceResult.Buckets[i].Location);
             }
        }
    }
});
```

NOTE

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

4.3 Deleting a Bucket

NOTICE

If you have any questions during 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:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
    //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
    access_key_id: process.env.ACCESS_KEY_ID,
    secret_access_key: process.env.SECRET_ACCESS_KEY,
    server : 'https://your-endpoint'
});

// Delete a bucket.
obsClient.deleteBucket({
    Bucket : 'bucketname'
}, (err, result) => {
    if(err){
        console.error('Error-->' + err);
    }else{
        console.log('Status-->' + result.CommonMsg.Status);
    }
});
```

CAUTION

After being deleted, there is a 30 minute delay before the name of the deleted bucket can be used as that of a new bucket or parallel file system in other regions.

NOTE

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

4.4 Identifying Whether a Bucket Exists

NOTICE

If you have any questions during 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. Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using 
hard coding may result in leakage.
    //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
    access_key_id: process.env.ACCESS_KEY_ID,
    secret_access_key: process.env.SECRET_ACCESS_KEY,
    server : 'https://your-endpoint'
});

obsClient.headBucket({
    Bucket : 'bucketname'
}, (err, result) => {
    if(err){
        console.error('Error-->' + err);
    }else{
        if(result.CommonMsg.Status < 300){
            console.log('Bucket exists');
        }else if(result.CommonMsg.Status === 404){
            console.log('Bucket does not exist');
        }
    }
});
```

4.5 Obtaining Bucket Metadata

NOTICE

If you have any questions during 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. Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using hard coding may result in leakage.
    //Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
    access_key_id: process.env.ACCESS_KEY_ID,
    secret_access_key: process.env.SECRET_ACCESS_KEY,
    server : 'https://your-endpoint'
});

// Obtain the bucket metadata.
obsClient.getBucketMetadata({
    Bucket : 'bucketname',
    Origin : 'http://www.a.com',
}, (err, result) => {
    if(err){
        console.error('Error-->' + err);
    }else{
        console.log('Status-->' + result.CommonMsg.Status);
        if(result.CommonMsg.Status < 300 && result.InterfaceResult){
            console.log('StorageClass-->' + result.InterfaceResult.StorageClass);
            console.log('AllowOrigin-->' + result.InterfaceResult.AllowOrigin);
            console.log('MaxAgeSeconds-->' + result.InterfaceResult.MaxAgeSeconds);
            console.log('ExposeHeader-->' + result.InterfaceResult.ExposeHeader);
            console.log('AllowMethod-->' + result.InterfaceResult.AllowMethod);
            console.log('AllowHeader-->' + result.InterfaceResult.AllowHeader);
        }
    }
});
```

4.6 Managing Bucket ACLs

NOTICE

If you have any questions during 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 access control policy during bucket creation.
2. Call **ObsClient.setBucketAcl** to specify a pre-defined access control policy.
3. Call **ObsClient.setBucketAcl** to set the ACL directly.

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

Permission	Description	Value in OBS Node.js SDK
READ	A grantee with this permission for a bucket can obtain the list of objects in and metadata of the bucket. A grantee with this permission for an object can obtain the object content and metadata.	ObsClient.enums.PermissionRead
WRITE	A grantee with this permission for a bucket can upload, overwrite, and delete any object in the bucket. This permission is not applicable to objects.	ObsClient.enums.PermissionWrite
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.	ObsClient.enums.PermissionReadAcp

Permission	Description	Value in OBS Node.js SDK
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.	ObsClient.enums.PermissionWriteAcp
FULL_CONTROL	A grantee with this permission for a bucket has **READ**, **WRITE**, **READ_ACP**, and **WRITE_ACP** permissions for the bucket. A grantee with this permission for an object has **READ**, **WRITE**, **READ_ACP**, and **WRITE_ACP** permissions for the object.	ObsClient.enums.PermissionFullControl

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

Permission	Description	Value in OBS Node.js SDK
private	Indicates that 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.	ObsClient.enums.AclPrivate
public-read	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. If this permission is set for an object, everyone can obtain the content and metadata of the object.	ObsClient.enums.AclPublicRead

Permission	Description	Value in OBS Node.js SDK
public-read-write	If this permission is granted on a bucket, anyone can obtain the object list, multipart tasks, and metadata, and can upload or delete objects, initialize multipart upload tasks, upload parts, merge parts, copy parts, and cancel multipart upload tasks. If this permission is set for an object, everyone can obtain the content and metadata of the object.	ObsClient.enums.AclPublicReadWrite
public-read-delivered	If this permission is set for a bucket, everyone can obtain the object list, multipart tasks, 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.	ObsClient.enums.AclPublicReadDelivered
public-read-write-delivered	If this permission is set for a bucket, everyone can obtain the object list in the bucket, multipart tasks 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.	ObsClient.enums.AclPublicReadWriteDelivered

Specifying a Pre-defined Access Control Policy During Bucket Creation

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

// Create a bucket.
obsClient.createBucket({
       Bucket : 'bucketname',
       // Set the bucket ACL to public-read-write.
       ACL : obsClient.enums.AclPublicReadWrite
}, (err, result) => {
       if(err){
              console.error('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

Setting a Pre-defined Access Control Policy for a Bucket

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

// Use the pre-defined access control policy to set bucket permissions.
obsClient.setBucketAcl({
       Bucket : 'bucketname',
       // Set the bucket ACL to private.
       ACL : obsClient.enums.AclPrivate
}, (err, result) => {
       if(err){
              console.error('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

NOTE

Use the **ACL** parameter to specify the ACL for a bucket.

Directly Setting a Bucket ACL

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
```

```
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

// Directly set the bucket ACL.
obsClient.setBucketAcl({
       Bucket : 'bucketname',
       // Set the bucket owner.
       Owner:{'ID':'ownerid'},
       Grants:[
            // Grant all permissions to a specified user.
           { Grantee : {Type : 'CanonicalUser',ID : 'userid'}, Permission : obsClient.enums.PermissionFullControl},
            // Grant the READ permission to all users.
           { Grantee : {Type : 'Group',URI : obsClient.enums.GroupAllUsers}, Permission :
obsClient.enums.PermissionRead}
      ]
}, (err, result) => {
       if(err){
              console.error('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

NOTE

- Use the **Owner** parameter to set the bucket owner and use the **Grants** parameter to grant permissions for authorized users.
- 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.
- OBS buckets support the following grantee group:
  - All users: ObsClient.enums.GroupAllUsers

Obtaining a Bucket ACL

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

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.getBucketAcl({
        Bucket : 'bucketname',
}, (err, result) => {
        if(err){
                console.error('Error-->' + err);
        }else{
                console.log('Status-->' + result.CommonMsg.Status);
                if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                        console.log('RequestId-->' + result.InterfaceResult.RequestId);
                        console.log('Owner[ID]-->' + result.InterfaceResult.Owner.ID);
                        console.log('Grants:');
                        for(let i=0;i<result.InterfaceResult.Grants.length;i++){
                               console.log('Grant[' + i + ']:');
                               console.log('Grantee[ID]-->' + result.InterfaceResult.Grants[i]['Grantee']['ID']);
                               console.log('Grantee[URI]-->' + result.InterfaceResult.Grants[i]['Grantee']['URI']);
                               console.log('Permission-->' + result.InterfaceResult.Grants[i]['Permission']);
                        }
                }
        }
});
```

4.7 Managing Bucket Policies

NOTICE

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

Besides bucket ACLs, bucket owners can use bucket policies to centrally control access to buckets and objects in buckets.

For more information, see **Bucket Policy**.

Setting a Bucket Policy

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

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
        //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
        //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
        access_key_id: process.env.ACCESS_KEY_ID,
        secret_access_key: process.env.SECRET_ACCESS_KEY,
        server : 'https://your-endpoint'
});

// Bucket name
const bucketName = 'bucketname';
// Bucket policy
const policy =  "{\"Statement\":[{\"Principal\":\"*\",\"Effect\":\"Allow\",\"Action\":\"ListBucket\",\"Resource
\":\""+bucketName+"\"}]}";
// Configure a bucket policy.
```

```
obsClient.setBucketPolicy({
       Bucket: bucketName,
       Policy: policy
}, function(err, result) {
       if(err){
              console.error('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

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:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

// Obtain the bucket policy.
obsClient.getBucketPolicy({
       Bucket : 'bucketname',
}, (err, result) => {
       if(err){
              console.error('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
              if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                     console.log('Policy-->' + result.InterfaceResult.Policy);
              }
       }
});
```

Deleting a Bucket Policy

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

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
```

```
        //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
        access_key_id: process.env.ACCESS_KEY_ID,
        secret_access_key: process.env.SECRET_ACCESS_KEY,
        server : 'https://your-endpoint'
});

// Delete a bucket policy.
obsClient.deleteBucketPolicy({
        Bucket : 'bucketname'
}, (err, result) => {
        if(err){
                console.error('Error-->' + err);
        }else{
                console.log('Status-->' + result.CommonMsg.Status);
        }
});
```

4.8 Obtaining a Bucket Location

NOTICE

If you have any questions during 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. Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
        //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
        //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
        access_key_id: process.env.ACCESS_KEY_ID,
        secret_access_key: process.env.SECRET_ACCESS_KEY,
        server : 'https://your-endpoint'
});

obsClient.getBucketLocation({
        Bucket : 'bucketname',
}, (err, result) => {
        if(err){
                console.error('Error-->' + err);
        }else{
                console.log('Status-->' + result.CommonMsg.Status);
                if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                        console.log('Location-->' + result.InterfaceResult.Location);
                }
        }
});
```

NOTE

When creating a bucket, you can specify its location. For details, see **Creating a Bucket**.

4.9 Obtaining Storage Information About a Bucket

NOTICE

If you have any questions during 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 bucket size and the number of objects in the bucket. You can call **ObsClient.getBucketStorageInfo** to obtain the bucket storage information. Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
    //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
    access_key_id: process.env.ACCESS_KEY_ID,
    secret_access_key: process.env.SECRET_ACCESS_KEY,
    server : 'https://your-endpoint'
});

obsClient.getBucketStorageInfo({
    Bucket : 'bucketname',
}, (err, result) => {
    if(err){
        console.error('Error-->' + err);
    }else{
        console.log('Status-->' + result.CommonMsg.Status);
        if(result.CommonMsg.Status < 300 && result.InterfaceResult){
            console.log('Size-->' + result.InterfaceResult.Size);
            console.log('ObjectNumber-->' + result.InterfaceResult.ObjectNumber);
        }
    }
});
```

4.10 Setting or Obtaining a Bucket Quota

NOTICE

If you have any questions during 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:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using 
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

//Set the bucket quota to 100 MB.
obsClient.setBucketQuota({
       Bucket : 'bucketname',
       StorageQuota: 1024 * 1024 * 100
}, (err, result) => {
       if(err){
              console.error('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

NOTE

- Use the **StorageQuota** parameter to specify the bucket quota.
- A bucket quota must be a non-negative integer expressed in bytes. The maximum value is 2^{63} - 1.

Obtaining a Bucket Quota

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

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using 
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.getBucketQuota({
       Bucket : 'bucketname'
}, (err, result) => {
       if(err){
              console.error('Error-->' + err);
```

```
    }else{
        console.log('Status-->' + result.CommonMsg.Status);
        if(result.CommonMsg.Status < 300 && result.InterfaceResult){
            console.log('StorageQuota-->' + result.InterfaceResult.StorageQuota);
        }
    }
});
```

4.11 Storage Class

NOTICE

If you have any questions during 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. There are three types of storage class for buckets, as described in the following table, catering to various storage performance and cost requirements.

Type	Description	Value in OBS Node.js 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.	ObsClient.enums.StorageClassStandard
OBS Infrequent Access	Is applicable to storing semi-frequently accessed (less than 12 times a year) data requiring quick response.	ObsClient.enums.StorageClassWarm
OBS Archive	Is applicable to archiving rarely-accessed (once a year) data.	ObsClient.enums.StorageClassCold

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:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
```

```
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using 
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.setBucketStoragePolicy({
       Bucket : 'bucketname',
       StorageClass: obsClient.enums.StorageClassWarm
}, (err, result) => {
       if(err){
              console.error('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

NOTE

Use the **StorageClass** parameter to set the storage class for a bucket.

Obtaining the Storage Class of a Bucket

You can call **ObsClient.getBucketStoragePolicy** to obtain the storage class of a bucket. Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using 
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.getBucketStoragePolicy({
       Bucket : 'bucketname'
}, (err, result) => {
       if(err){
              console.error('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
              if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                     console.log('StorageClass-->' + result.InterfaceResult.StorageClass);
              }
       }
});
```

5 Object Upload

5.1 Object Upload Overview

NOTICE

If you have any questions during 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 Node.js SDK provides abundant APIs for object upload in the following methods:

- **Performing a Text-Based Upload**
- **Performing a Streaming Upload**
- **Performing a File-Based Upload**
- **Performing a Multipart Upload**
- **Performing an Appendable Upload**
- **Performing a Resumable Upload**
- **Performing a Browser-Based Upload**

The SDK supports the upload of objects whose size ranges from 0 KB to 5 GB. For streaming upload, appendable upload, and file-based upload, data to be uploaded at a time cannot be larger than 5 GB. If the file is larger than 5 GB, multipart upload (whose part size is smaller than 5 GB) is suitable. Browser-based upload allows files to be uploaded through a browser.

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.

5.2 Performing a Text-Based Upload

NOTICE

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

Text-based upload is used to directly upload character strings. You can call **ObsClient.putObject** to upload character strings to OBS. Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using 
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.putObject({
       Bucket : 'bucketname',
       Key : 'objectname',
       Body : 'Hello OBS'
}, (err, result) => {
       if(err){
              console.error('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

NOTE

- Use the **Body** parameter to specify the character string to be uploaded.
- The content to be uploaded cannot exceed 5 GB.

5.3 Performing a Streaming Upload

NOTICE

If you have any questions during 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 **stream.Readable** as the data source of objects. Sample code is as follows:

Uploading a Network Stream

```
// Import the OBS library.
// Use npm to install the client.
const ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// const ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

// Define the network stream URL.
const url = 'https://www.example.com'
// Import HTTP and HTTPS libraries.
const http = require('http');
const https = require('https');
// Choose the corresponding library based on the URL.
const request = url.startsWith('http') ? http : https;
// Obtain the network stream.
request.get(url, (res) => {
    if (res.statusCode === 200) {
        // Call the PutObject method after the network stream is successfully obtained.
        obsClient.putObject({
            // Bucket name, for example, examplebucket.
            Bucket: 'examplebucket',
            // Enter a complete object path, for example, exampledir/exampleobject.txt. This path cannot contain the bucket name.
            Key: 'exampledir/exampleobject.txt',
            // res is an instance of the http.IncomingMessage class and is a readable stream.
            Body: res
        }, (err, result) => {
            if (err) {
                console.error('Error-->' + err);
                // Handle network exceptions or other exceptions.
            } else {
                console.log('Status-->' + result.CommonMsg.Status);
                // Process the service code.
            }
        });
    }
});
```

Uploading a File Stream

```
// Import the OBS library.
// Use npm to install the client.
const ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// const ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

const fs = require('fs');
// Enter the complete path of the local file and read the data stream from the local file.
// If there is no local path in the complete path, the system uploads the file from the local path of the
project where the sample program is located by default.
const stream = fs.createReadStream('D:\\localpath\\examplefile.txt');
obsClient.putObject({
     // Bucket name, for example, examplebucket.
     Bucket: 'examplebucket',
     // Enter a complete object path, for example, exampledir/exampleobject.txt. This path cannot contain
the bucket name.
     Key: 'exampledir/exampleobject.txt',
     // stream is a file readable stream.
     Body: stream
}, (err, result) => {
       if(err){
              console.error('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

NOTICE

- When you use the **Body** parameter to specify the stream data to be uploaded, the parameter value must be an instance of **stream.Readable**.
- To upload a large file, you are advised to use **multipart upload**.
- The content to be uploaded cannot exceed 5 GB.

5.4 Performing a File-Based Upload

NOTICE

If you have any questions during 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 data sources. The following sample code shows how to perform a file upload:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.putObject({
       Bucket : 'bucketname',
       Key : 'objectname',
         SourceFile : 'localfile'  // Path of the local file to be uploaded. The file name must be specified.
}, (err, result) => {
       if(err){
              console.error('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

NOTE

- Use the **SourceFile** parameter to specify the path to the to-be-uploaded file.
- The **SourceFile** parameter and the **Body** parameter cannot be used together.
- The content to be uploaded cannot exceed 5 GB.

5.5 Creating a Folder

NOTICE

If you have any questions during 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.

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.putObject({
       Bucket : 'bucketname',
       Key : 'parent_directory/'
}, (err, result) => {
       if(err){
              console.error('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

```
// Create an object in the folder.
obsClient.putObject({
      Bucket : 'bucketname',
      Key : 'parent_directory/objectname'
}, (err, result) => {
      if(err){
            console.error('Error-->' + err);
      }else{
            console.log('Status-->' + result.CommonMsg.Status);
      }
});
```

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.

5.6 Setting Object Properties

NOTICE

If you have any questions during 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 object length, MIME type, MD5 value (for verification), storage class, and customized metadata. You can set properties for an object that is being uploaded in text-based, streaming, file-based, or multipart mode or when **copying the object**.

The following table describes object properties.

Property Name	Description	Default Value
Content-Length	Indicates the object length. If the object length exceeds the flow or file length, the object will be truncated.	Actual length of the stream or file
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	Base64-encoded MD5 value of the object data. It is provided for the OBS server to verify data integrity.	None

Property Name	Description	Default Value
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.	None
Customized metadata	Indicates the user-defined description of the object. It is used to facilitate the customized management on the object.	None

Setting the Length for an Object

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.putObject({
       Bucket : 'bucketname',
       Key : 'objectname',
       SourceFile : 'localfile',
       ContentLength : 1024 * 1024 // 1 MB
}, (err, result) => {
       if(err){
              console.error('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

NOTE

Use the **ContentLength** parameter to specify the object length.

Setting the MIME Type for an Object

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');
// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

// Upload an image.
obsClient.putObject({
       Bucket : 'bucketname',
       Key : 'objectname.jpg',
       SourceFile : 'localimage.jpg',
       ContentType : 'image/jpeg'
}, (err, result) => {
       if(err){
              console.error('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

NOTE

- Use the **ContentType** parameter to set the MIME type for an object.
- If this property is not specified, the SDK will automatically identify the MIME type according to the suffix of the object name. For example, if the suffix of the object name is **.xml** (**.html**), the object will be identified as an application/xml (text/html) file.

Setting the MD5 Value for an Object

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.putObject({
       Bucket : 'bucketname',
       Key : 'objectname',
       SourceFile : 'localimage.jpg',
       ContentMD5 : 'your md5 which should be encoded by base64'
}, (err, result) => {
       if(err){
              console.error('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

NOTE

- Use the **ContentMD5** parameter to specify the MD5 value for an object.
- 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 an HTTP **400** error returned.
- If the MD5 value is not specified, the OBS server will skip MD5 value verification.

Setting the Storage Class for an Object

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.putObject({
       Bucket : 'bucketname',
       Key : 'objectname',
       SourceFile : 'localfile',
       // Set the storage class to Archive.

       StorageClass : ObsClient.enums.StorageClassCold
}, (err, result) => {
       if(err){
              console.error('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

NOTE

- Use the **StorageClass** parameter to set the storage class for an object.
- If you do 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

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
```

```
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.putObject({
       Bucket : 'bucketname',
       Key : 'objectname',
       SourceFile : 'localfile',
       Metadata : {'property1':'property-value1', 'property2' : 'property-value2'},
}, (err, result) => {
       if(err){
              console.error('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

NOTE

- Use the **Metadata** parameter to customize the metadata of an object.
- In the preceding code, two pieces of metadata named **property1** and **property2** are customized and their respective values are set to **property-value1** and **property-value2**.
- 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 Metadata**.
- When you call **ObsClient.getObject** to download an object, its customized metadata will also be downloaded.

5.7 Performing a Multipart Upload

NOTICE

If you have any questions during 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 Initiate 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 the multipart upload, listing multipart uploads, and listing uploaded parts.

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

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.initiateMultipartUpload({
       Bucket : 'bucketname',
       Key : 'objectname',
       ContentType : 'text/plain',
       Metadata : {'property' : 'property-value'}
}, (err, result) => {
       if(err){
              console.error('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
              if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                     console.log('UploadId-->' + result.InterfaceResult.UploadId);
              }
       }
});
```

NOTE

- When initiating a multipart upload, you can use the **ContentType** and **Metadata** parameters to respectively set the MIME type and custom metadata of an object.
- After the API for initiating a multipart upload is successfully called, an upload ID will be returned. This ID will be used in follow-up operations.

Uploading a Part

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. The last part uploaded ranges from 0 to 5 GB in size, and **each of the other parts ranges 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.

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using 
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.uploadPart({
       Bucket:'bucketname',
       Key:'objectname',
       // Set the part number, which ranges from 1 to 10000.
       PartNumber:1,
       // Set the upload ID.
       UploadId:'upload id from initiateMultipartUpload',
       // Set the large file to be uploaded. localfile is the path of the local file to be uploaded. You need to 
specify the file name.
       SourceFile: 'localfile',
       // Set the part size.
       PartSize: 5 * 1024 * 1024,
       // Set the start offset.
       Offset: 0
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
              if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                     console.log('ETag-->' + result.InterfaceResult.ETag);
              }
       }
});
```

NOTE

- Use the **PartNumber** parameter to specify the part number, the **UploadId** parameter to specify the globally unique ID, the **SourceFile** parameter to specify the to-be-uploaded file, the **PartSize** parameter to set the part size, and the **Offset** parameter to set the start offset of the file.
- 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.
- You can use the **ContentMD5** parameter to set the MD5 value for the object to be uploaded, which can be provided to the OBS server for data integrity verification.
- Part numbers range from 1 to 10000. If a part number exceeds this range, OBS will return **a 400 Bad Request** error.
- 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.

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.completeMultipartUpload({
       Bucket:'bucketname',
       Key:'objectname',
       // Set the upload ID.
       UploadId:'upload id from initiateMultipartUpload',
       Parts: [{'PartNumber':1,'ETag':'etag value from uploadPart'}]
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

CAUTION

- If the size of a part other than the last part is smaller than 100 KB, OBS returns **400 Bad Request**.

NOTE

- Use the **UploadId** parameter to specify the upload ID for the multipart upload and the **Parts** parameter to specify the list of part numbers and ETags. Content in the list is displayed in the ascending order by part number.
- Part numbers can be inconsecutive.

Concurrently Uploading Parts

Multipart upload is mainly used for large file upload or when the network condition is poor. The following sample code shows how to concurrently upload large files in multipart mode:

```
// Import the OBS library.
// Use npm to install the client.
const ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');
const fs = require('fs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

const Bucket = 'bucketname'
const Key = 'objectname'

// Specify the part size.
const DEFAULT_PART_SIZE = 9 * 1024 * 1024;
// Set the number of concurrent parts to 20.
const CONCURRENCY = 20

// Prepare multipart upload parameters.
const preparePartParams = (Bucket, Key, UploadId) => (sampleFile, partSize = DEFAULT_PART_SIZE) => {
    try {
        const fileSize = fs.lstatSync(sampleFile).size;
        const partCount = fileSize % partSize === 0 ? Math.floor(fileSize / partSize) : Math.floor(fileSize / partSize) + 1;

        const uploadPartParams = [];
        // Specify the concurrent upload.
        for (let i = 0; i < partCount; i++) {
            // Start position of parts in the file
            let Offset = i * partSize;
            // Part size
            let currPartSize = (i + 1 === partCount) ? fileSize - Offset : partSize;
            // Part number
            let PartNumber = i + 1;
            uploadPartParams.push({
                Bucket,
                Key,
                PartNumber,
                UploadId,
                Offset,
                SourceFile: sampleFile,
                PartSize: currPartSize,
            });
        }

        return ({ uploadPartParams, fileSize });
    } catch (error) {
        console.log(error)
    }
}


/**
 * uploadSuccessSize: Size of the parts that have been uploaded
 * uploadSuccessCount: Number of the parts that have been uploaded
 * concurrency: Current concurrency
 */
let uploadSuccessSize = 0;
let uploadSuccessCount = 0;
let concurrency = 0

const parts = [];

const uploadPart = (uploadPartParam, otherUploadPartInfo) => {
    const partCount = otherUploadPartInfo.partCount;
    const fileSize = otherUploadPartInfo.fileSize;
    concurrency++;
    return obsClient
        .uploadPart(uploadPartParam)
        .then(result => {
            const { PartNumber, PartSize } = uploadPartParam;
            if (result.CommonMsg.Status < 300) {
                uploadSuccessCount++;
                uploadSuccessSize += PartSize;
                console.log(`the current concurrent count is ${concurrency} | uploaded segment: $
{uploadSuccessCount}/${partCount}. the progress is ${((uploadSuccessSize / fileSize) * 100).toFixed(2)}% |
the partNumber ${PartNumber} upload successed.`)
                parts.push({ PartNumber, ETag: result.InterfaceResult.ETag });
            } else {
                console.log(result.CommonMsg.Code, parts)
            }
            concurrency--;
        }).catch(function (err) {
            console.log(err);
            throw err;
        })
}

const uploadFile = (sourceFile) => {
    obsClient.initiateMultipartUpload({
        Bucket,
        Key
    }).then(res => {
        const Status = res.CommonMsg.Status;
        const UploadId = res.InterfaceResult.UploadId;

        if (typeof Status === 'number' && Status > 300) {
            console.log(`initiateMultipartUpload failed! Status:${Status}`);
            return;
        }

        const partParams = preparePartParams(Bucket, Key, UploadId)(sourceFile)
        const uploadPartParams = partParams.uploadPartParams;
        const fileSize = partParams.fileSize;
        const partCount = uploadPartParams.length;
        const otherUploadPartInfo = { fileSize, partCount }

        // Call the parallel upload function.
        parallelFunc(uploadPartParams, (param) => uploadPart(param, otherUploadPartInfo),
CONCURRENCY)
            .then(() => {
                obsClient.completeMultipartUpload({
                    Bucket,
                    Key,
                    UploadId,
                    Parts: parts.sort((a, b) => a.PartNumber - b.PartNumber)
                }, (err, result) => {
                    if (err) {
                        console.log('Error-->' + err);
                    } else {
                        console.log('Status-->' + result.CommonMsg.Status);
                    }
                });
            })

    }).catch(function (err) {
        console.log(err)
    })
}


/**
 * Implement the parallel execution.
 * @param {Array} params Parameter array of the callback function
 * @param {Promise} promiseFn Callback function
 * @param {number} limit Number of parallel parts
 */
const parallelFunc = (params, promiseFn, limit) => {
    return new Promise((resolve) => {
        let concurrency = 0;
        let finished = 0;
        const count = params.length;
        const run = (param) => {
            concurrency++;
            promiseFn(param)
                .then(() => {
                    concurrency--;
                    drainQueue();
                    finished++
                    if (finished === count) {
                        resolve()
                    }
                });
        }
        const drainQueue = () => {
            while (params.length > 0 && concurrency < limit) {
                var param = params.shift();
                run(param);
            }
        }
        drainQueue();
    })

}

uploadFile('localfile');
```

NOTE

When uploading a large file in multipart mode, you need to use the **Offset** and **PartSize** parameters to set the start and end positions of each part in the file.

CAUTION

If there are too many concurrent parts, a timeout error may occur due to network instability or other causes. In such case, you need to limit the number of concurrent parts.

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 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.

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.abortMultipartUpload({
       Bucket:'bucketname',
       Key:'objectname',
       // Set the upload ID.
       UploadId:'upload id from initiateMultipartUpload',
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

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
UploadId	Upload ID, which globally identifies a multipart upload. The value is in the returned result of **ObsClient.initiateMultipartUpload**.
MaxParts	Maximum number of parts that can be listed per page.
PartNumberMarker	Part number after which listing uploaded parts begins. Only parts whose part numbers are larger than this value will be listed.

- Listing parts in simple mode

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

// List uploaded parts. uploadId is obtained from initiateMultipartUpload.
obsClient.listParts({
       Bucket : 'bucketname',
       Key: 'objectname',
       UploadId : 'upload id from initiateMultipartUpload'
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
              if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                     for(let i=0;i<result.InterfaceResult.Parts.length;i++){
                            console.log('Part['+ i +']:');
                            // Part number, specified during the upload
                            console.log('PartNumber-->' + result.InterfaceResult.Parts[i]['PartNumber']);
                            // Time when the part was last uploaded
                            console.log('LastModified-->' + result.InterfaceResult.Parts[i]['LastModified']);
                            // Part ETag
                            console.log('ETag-->' + result.InterfaceResult.Parts[i]['ETag']);
                            // Part size
                            console.log('Size-->' + result.InterfaceResult.Parts[i]['Size']);
                     }
              }
       }
});
```

NOTE

- A maximum of 1000 parts can be returned each time. If an upload of the specific upload ID contains more than 1000 parts and **InterfaceResult.IsTruncated** is **true** in the returned result, not all parts are returned. In such cases, you can use **InterfaceResult.NextPartNumberMarker** to obtain the start position for next listing.
- If you want to obtain all parts involved in a specific upload ID, you can use the paging mode for listing.

- Listing all parts

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

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

var listAll = (partNumberMarker) => {
       // List uploaded parts. uploadId is obtained from initiateMultipartUpload.
       obsClient.listParts({
              Bucket : 'bucketname',
              Key: 'objectname',
              UploadId : 'upload id from initiateMultipartUpload',
              PartNumberMarker : partNumberMarker
       }, (err, result) => {
              if(err){
                     console.log('Error-->' + err);
              }else{
                     console.log('Status-->' + result.CommonMsg.Status);
                     if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                           for(let i=0;i<result.InterfaceResult.Parts.length;i++){
                                  console.log('Part['+ i +']:');
                                  // Part number, specified during the upload
                                  console.log('PartNumber-->' + result.InterfaceResult.Parts[i]['PartNumber']);
                           // Time when the part was last uploaded
                                  console.log('LastModified-->' + result.InterfaceResult.Parts[i]['LastModified']);
                                  // Part ETag
                                  console.log('ETag-->' + result.InterfaceResult.Parts[i]['ETag']);
                                  // Part size
                                  console.log('Size-->' + result.InterfaceResult.Parts[i]['Size']);
                           }
                           if(result.InterfaceResult.IsTruncated === 'true'){
                                  listAll(result.InterfaceResult.NextPartNumberMarker);
                           }
                     }
              }
       });
};

listAll();
```

Listing Multipart Uploads

You can call **ObsClient.listMultipartUploads** to list multipart uploads. The following table describes parameters involved in **ObsClient.listMultipartUploads**.

Parameter	Description
Prefix	Prefix that the object names in the multipart uploads to be listed must contain
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** parameter 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.)
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.
KeyMarker	Object name to start with when listing multipart uploads
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.

- Listing multipart uploads in simple mode

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using 
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.listMultipartUploads({
       Bucket : 'bucketname'
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
              if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                     for(let i=0;i<result.InterfaceResult.Uploads.length;i++){
                            console.log('Uploads[' + i + ']');
                            console.log('UploadId-->' + result.InterfaceResult.Uploads[i]['UploadId']);
                            console.log('Key-->' + result.InterfaceResult.Uploads[i]['Key']);
                            console.log('Initiated-->' + result.InterfaceResult.Uploads[i]['Initiated']);
                     }
              }
       }
});
```

NOTE

- Information about a maximum of 1000 multipart uploads can be returned each time. If a bucket contains more than 1000 multipart uploads and **InterfaceResult.IsTruncated** is **true** in the returned result, not all uploads will be returned. In such cases, you can use **InterfaceResult.NextKeyMarker** and **InterfaceResult.NextUploadIdMarker** to obtain the start position for next listing.
- If you want to obtain all multipart uploads in a bucket, you can use the paging listing.

- Listing all multipart uploads

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

var listAll = (keyMarker, uploadIdMarker) => {
       obsClient.listMultipartUploads({
              Bucket : 'bucketname',
              KeyMarker : keyMarker,
              UploadIdMarker : uploadIdMarker
       }, (err, result) => {
              if(err){
                     console.log('Error-->' + err);
              }else{
                     console.log('Status-->' + result.CommonMsg.Status);
                     if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                            for(let i=0;i<result.InterfaceResult.Uploads.length;i++){
                                   console.log('Uploads[' + i + ']');
                                   console.log('UploadId-->' + result.InterfaceResult.Uploads[i]['UploadId']);
                                   console.log('Key-->' + result.InterfaceResult.Uploads[i]['Key']);
                                   console.log('Initiated-->' + result.InterfaceResult.Uploads[i]['Initiated']);
                            }

                            if(result.InterfaceResult.IsTruncated === 'true'){
                                   listAll(result.InterfaceResult.NextKeyMarker,
result.InterfaceResult.NextUploadIdMarker);
                            }
                     }
              }
       });
}

listAll();
```

5.8 Configuring Lifecycle Management

NOTICE

If you have any questions during 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 initiating a multipart upload, you can directly set the expiration time for the object. Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

// When uploading an object, set the object to expire after 30 days.
obsClient.putObject({
       Bucket : 'bucketname',
       Key : 'objectname',
       Body : 'Hello OBS',
       Expires : 30
}, (err, result) => {
       if(err){
              console.error('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});

// When initiating a multipart upload, configure the object to expire 60 days after combination.
obsClient.initiateMultipartUpload({
       Bucket : 'bucketname',
       Key : 'objectname',
       ContentType : 'text/plain',
       Expires : 60
}, (err, result) => {
       if(err){
              console.error('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
              if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                     console.log('UploadId-->' + result.InterfaceResult.UploadId);
              }
       }
});
```

NOTE

- Use the **Expires** parameter to specify expiration time for the object.
- 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.

5.9 Performing an Appendable Upload

NOTICE

If you have any questions during 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:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using 
hard coding may result in leakage.
    //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
    access_key_id: process.env.ACCESS_KEY_ID,
    secret_access_key: process.env.SECRET_ACCESS_KEY,
    server : 'https://your-endpoint'
});

// Create an appendable object. The start position must be 0.
obsClient.appendObject({
    Bucket:'bucketname',
    Key:'objectname',
    Position : 0,
    Body : 'Hello OBS'
}).then(function(result){
    console.log('Status-->' + result.CommonMsg.Status);
    if(result.CommonMsg.Status < 300 && result.InterfaceResult){
        console.log('NextPosition-->' + result.InterfaceResult.NextPosition);
    }
    // Append data to the object.
    obsClient.appendObject({
        Bucket:'bucketname',
        Key:'objectname',
        Position : result.InterfaceResult.NextPosition,
        Body : 'Hello OBS Again'
    }, function(err, result2){
        if(err){
            console.error('Error-->' + err);
        }else{
            console.log('Status-->' + result2.CommonMsg.Status);
            if(result2.CommonMsg.Status < 300 && result2.InterfaceResult){
                console.log('NextPosition-->' + result2.InterfaceResult.NextPosition);
            }
        }
    });

    // Use the API for obtaining object properties to get the start position for next appending.
    obsClient.getObjectMetadata({
        Bucket:'bucketname',
        Key:'objectname',
    }).then(function(result3){
        console.log('Status-->' + result3.CommonMsg.Status);
        if(result3.CommonMsg.Status < 300 && result3.InterfaceResult){
            console.log('RequestId-->' + result3.InterfaceResult.RequestId);
            console.log('NextPosition-->' + result3.InterfaceResult.NextPosition);
        }
    }).catch(function(err){
        console.error('err:' + err);
    });

}).catch(function(err){
    console.error('err:' + err);
});
```

NOTE

- Use the **Position** parameter to specify the start position for next appending and set it to **0** when you create an appendable object.
- 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 anymore once the object has been overwritten by a normal object.
- When you upload an object for the first time in appendable mode, an exception will be reported (HTTP status code **409**) if a common object with the same name exists.
- The ETag returned for an appendable upload is the ETag for the uploaded content, rather than that of the whole object.
- Data size in each appendable upload cannot exceed 5 GB, and 10,000 times of appendable uploads can be performed on a single object.
- After an appendable upload is complete successfully, you can use **InterfaceResult.NextPosition** obtained from the returned result or call **ObsClient.getObjectMetadata**, to get the location for next appending.

5.10 Performing a Multipart Copy

NOTICE

If you have any questions during 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 partial object in a bucket. You can call **ObsClient.copyPart** to copy parts. Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
```

```
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

var destBucketName = 'destbucketname';
var destObjectKey = 'destobjectname';
var sourceBucketName = 'sourcebucketname';
var sourceObjectKey = 'sourceobjectname';

// Initiate a multipart upload.
obsClient.initiateMultipartUpload({
       Bucket : destBucketName,
       Key : destObjectKey
}, (err, result) => {
       if(!err && result.CommonMsg.Status < 300){
              var uploadId = result.InterfaceResult.UploadId;
              console.log('\t' + uploadId + '\n');

              // Obtain information about the large object.
              obsClient.getObjectMetadata({
                     Bucket : sourceBucketName,
                     Key : sourceObjectKey
              }, (err, result) => {
                     if(!err && result.CommonMsg.Status < 300){
                           // Set the part size to 100 MB.
                           var partSize = 100 * 1024 * 1024;
                           var objectSize = Number(result.InterfaceResult.ContentLength);

                           // Calculate the number of parts to be copied.
                           var partCount = objectSize % partSize === 0 ? Math.floor(objectSize / partSize) :
Math.floor(objectSize / partSize) + 1;

                           var events = require('events');
                           var eventEmitter = new events.EventEmitter();
                           var parts = [];

                           // Concurrently copy parts.
                           for(let i=0;i<partCount;i++){
                                  let rangeStart = i * partSize;
                                  let rangeEnd = (i + 1) === partCount ? objectSize - 1 : rangeStart + partSize - 1;
                                  let partNumber = i + 1;
                                  obsClient.copyPart({
                                         Bucket: destBucketName,
                                         Key: destObjectKey,
                                         PartNumber : partNumber,
                                         UploadId : uploadId,
                                         CopySource: sourceBucketName + '/' + sourceObjectKey,
                                         CopySourceRange : 'bytes=' + rangeStart + '-' + rangeEnd
                                  }, (err, result) => {
                                         if(!err && result.CommonMsg.Status < 300){
                                                parts.push({PartNumber : partNumber, ETag : result.InterfaceResult.ETag});
                                                if(parts.length === partCount){
                                                       var _parts = parts.sort((a, b) => {
                                                              if(a.PartNumber >= b.PartNumber){
                                                                     return 1;
                                                              }
                                                              return -1;
                                                       });
                                                       eventEmitter.emit('copy part finished', _parts);
                                                }
                                         }else{
                                                throw new Error(err || result.CommonMsg.Code);
                                         }
```

```
                        });
                    }

                    // Wait until the copy is complete.
                    eventEmitter.on('copy part finished', (parts) => {
                        // Combine the parts.
                        obsClient.completeMultipartUpload({
                            Bucket: destBucketName,
                            Key: destObjectKey,
                            UploadId: uploadId,
                            Parts: parts
                        }, (err, result) => {
                            if(!err && result.CommonMsg.Status < 300){
                                console.log('Complete to upload multiparts finished.\n');
                            }
                        });
                    });
                }
            });
        }
});
```

NOTE

Use the **PartNumber** parameter to specify the part number, the **UploadId** parameter to specify the globally unique ID for the multipart upload, the **CopySource** parameter to specify the information about the source object, and the **CopySourceRange** parameter to specify the copy range.

5.11 Performing a Resumable Upload

NOTICE

If you have any questions during 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 will be returned. Otherwise, an error is returned in callback function 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.

CAUTION

- The total size of files uploaded by the resumable upload API must be larger than 100 KB.

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

Parameter	Description
Bucket	(Mandatory) Bucket name
Key	(Mandatory) Object name
UploadFile	(Mandatory) Local file to be uploaded
PartSize	Part size, in bytes. The value ranges from 100 KB to 5 GB and defaults to 5 MB.
TaskNum	Maximum number of threads that can be concurrently executed for upload. The default value is **20**.
EnableCheckpoint	Whether to enable the resumable upload mode. The default value is **false**, which indicates that this mode is disabled.
CheckpointFile	File used to record the upload progress. This parameter is effective only in the resumable upload mode. If this parameter is null, the file will be in the same directory as the local file to be uploaded.
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.

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using 
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.uploadFile({
       Bucket : 'bucketname',
       Key : 'objectname',
       // Set the large file to be uploaded. localfile is the path of the local file to be uploaded. You need to 
specify the file name.
       UploadFile : 'localfile',
       // Set the part size to 10 MB.
       PartSize : 10 * 1024 * 1024,
       // Enable the resumable upload mode.
       EnableCheckpoint : true
}, (err, result) => {
    if(err){
        console.error('Error-->' + err);
    }else{
        console.log('RequestId-->' + result.InterfaceResult.RequestId);
        console.log('Bucket-->' + result.InterfaceResult.Bucket);
        console.log('Key-->' + result.InterfaceResult.Key);
        console.log('Location-->' + result.InterfaceResult.Location);
    }
});
```

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, this API degrades to the simple encapsulation of multipart upload, and no checkpoint file will be generated.
- **CheckpointFile** and **EnableCheckSum** are effective only when **EnableCheckpoint** is **true**.

5.12 Performing a Browser-Based Upload

NOTICE

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

Performing a browser-based upload is to upload objects to a specified bucket in HTML form. The maximum size of an object is 5 GB.

You can call **ObsClient.createPostSignatureSync** to generate request parameters for a browser-based upload. You can use Node.js code to simulate a browser-based upload. For details, see **post-object-sample**. You can also perform a browser-based upload with the following steps:

Step 1 Call **ObsClient.createPostSignatureSync** to generate request parameters for authentication.

Step 2 Prepare an HTML form page.

Step 3 Enter the request parameters in the HTML page.

Step 4 Select a local file and upload it in browser-based mode.

----End

NOTE

There are two request parameters generated:

- **Policy**, which corresponds to the **policy** field in the form
- **Signature**: which corresponds to the **signature** field in the form

The following sample code shows how to generate the parameters in a browser-based upload request.

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint',
       signature : 'obs'
});

// Fill in parameters in the form.
var formParams = {
              // Set the object ACL to public-read.
              'x-obs-acl': obsClient.enums.AclPublicRead,
              // Set the MIME type for the object.
              'content-type': 'text/plain',
};

// Set the validity period for the browser-based upload request, in seconds.
var expires = 3600;

var res = obsClient.createPostSignatureSync({Expires:expires, FormParams: formParams});

// Obtain the request parameters.
console.log('\t' + res.Policy);
console.log('\t' + res.Signature);
```

Code of an HTML form example is as follows:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>

<form action="http://bucketname.your-endpoint/" method="post" enctype="multipart/form-data">
Object key
<!-- Object name -->
<input type="text" name="key" value="objectname" />
<p>
ACL
<!-- Object ACL -->
<input type="text" name="x-obs-acl" value="public-read" />
<p>
Content-Type
<!-- Object MIME type -->
<input type="text" name="content-type" value="text/plain" />
<p>
<!-- Base64 code of the policy -->
<input type="hidden" name="policy" value="*** Provide your policy ***" />
<!-- AK -->
<input type="hidden" name="AccessKeyId" value="*** Provide your access key ***"/>
<!-- Signature information -->
<input type="hidden" name="signature" value="*** Provide your signature ***"/>

<input name="file" type="file" />
<input name="submit" value="Upload" type="submit" />
</form>
```

```
</body>
</html>
```

NOTE

- Values of **policy** and **signature** in the HTML form are obtained from the value returned by **ObsClient.createPostSignatureSync**.
- You can directly download the HTML form example: **PostDemo**.

6 Object Download

6.1 Object Download Overview

NOTICE

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

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

- **Performing a Text-Based Download**
- **Performing a Streaming Download**
- **Performing a File-Based Download**
- **Performing a Partial Download**
- **Performing a Resumable Download**

You can call **ObsClient.getObject** to download an object.

6.2 Performing a Text-Based Download

NOTICE

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

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
```

```
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.getObject({
       Bucket : 'bucketname',
       Key : 'objectname'
}, (err, result) => {
       if(err){
              console.error('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
              if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                   // Obtain the object content.
                   console.log('Object Content:');
                   console.log(result.InterfaceResult.Content);
              }
       }
});
```

NOTE

In the returned result of a text-based download, **InterfaceResult.Content** is a String object.

6.3 Performing a Streaming Download

NOTICE

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

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.getObject({
       Bucket : 'bucketname',
	Key : 'objectname',
	SaveAsStream : true
}, (err, result) => {
	if(err){
		console.error('Error-->' + err);
	}else{
		console.log('Status-->' + result.CommonMsg.Status);
		if(result.CommonMsg.Status < 300 && result.InterfaceResult){
			// Read the object content.
			console.log('Object Content:\n');
			result.InterfaceResult.Content.on('data', (data) => {
				console.log(data.toString());
			});
		}
	}
});
```

NOTE

- Use the **SaveAsStream** parameter to specify the download mode to streaming download.
- **InterfaceResult.Content** is an instance of **stream.Readable** and can be used to save the object content to a local file or the memory.

6.4 Performing a File-Based Download

NOTICE

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

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
	//Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
	//Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
	access_key_id: process.env.ACCESS_KEY_ID,
	secret_access_key: process.env.SECRET_ACCESS_KEY,
	server : 'https://your-endpoint'
});

obsClient.getObject({
	Bucket : 'bucketname',
	Key : 'objectname',
	SaveAsFile : 'localfile'
}, (err, result) => {
	if(err){
		console.error('Error-->' + err);
	}else{
		console.log('Status-->' + result.CommonMsg.Status);
	}
});
```

NOTE

Use the **SaveAsFile** parameter to specify the path for saving the to-be-downloaded file.

6.5 Performing a Partial Download

NOTICE

If you have any questions during 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 0 to 1000, data at the zeroth to the thousandth bytes, 1001 bytes in total, will be returned. If the specified range is invalid, data of the whole object will be returned. Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using 
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.getObject({
       Bucket : 'bucketname',
       Key : 'objectname',
       // Set the start position and end position for downloading.
       Range : 'bytes=0-1000'
}, (err, result) => {
       if(err){
              console.error('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
              if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                   // Obtain the object content.
                   console.log('Object Content:');
                   console.log(result.InterfaceResult.Content.toString());
              }
       }
});
```

NOTE

- Use the **Range** parameter to specify the download range in the "*bytes=x-y*" format.
- If the specified range is invalid (because the start or end position is set to a negative integer or the range is larger than the object length), data of the whole object will be returned.
- This download method also can be used to concurrently download parts of a large object. For details about the sample code, see **concurrent-download-object-sample**.

6.6 Performing a Conditioned Download

NOTICE

If you have any questions during 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 error code will be returned and the download will fail.

You can set the following conditions:

Parameter	Description	Format
IfModifiedSince	Returns the object if it has been modified since the specified time; otherwise, an error is returned.	This parameter must conform to the HTTP time format specified in http://www.ietf.org/rfc/rfc2616.txt.
IfUnmodifiedSince	Returns the object if it has not been modified since the specified time; otherwise, an error is returned.	This parameter must conform to the HTTP time format specified in http://www.ietf.org/rfc/rfc2616.txt.
IfMatch	Returns the source object if its ETag is the same as the one specified by this parameter; otherwise, an error code is returned.	Character string
IfNoneMatch	Returns the source object if its ETag is different from the one specified by this parameter; otherwise, an error code is returned.	Character string

NOTE

- The ETag of the source object is the MD5 check value of the source object content.
- If a request includes **IfUnmodifiedSince** or **IfMatch** and the specified condition is not met, the object download will fail with error code **412 Precondition Failed** returned.
- If a request includes **IfModifiedSince** or **IfNoneMatch** and the specified condition is not met, the object download will fail with error code **304 Not Modified** returned.

Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.getObject({
       Bucket : 'bucketname',
       Key : 'objectname',
       IfModifiedSince : 'Wed, 04 Jul 2018 08:54:53 GMT'
}, (err, result) => {
       if(err){
              console.error('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
              if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                   // Obtain the object content.
                   console.log('Object Content:');
                   console.log(result.InterfaceResult.Content.toString());
              }
       }
});
```

6.7 Rewriting Response Headers

NOTICE

If you have any questions during 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
ResponseContentType	Rewrites **Content-Type** in HTTP/HTTPS responses.

Parameter	Description
ResponseContentLanguage	Rewrites **Content-Language** in HTTP/HTTPS responses.
ResponseExpires	Rewrites **Expires** in HTTP/HTTPS responses.
ResponseCacheControl	Rewrites **Cache-Control** in HTTP/HTTPS responses.
ResponseContentDisposition	Rewrites **Content-Disposition** in HTTP/HTTPS responses.
ResponseContentEncoding	Rewrites **Content-Encoding** in HTTP/HTTPS responses.

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');


// Create an instance of ObsClient.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.getObject({
       Bucket : 'bucketname',
       Key : 'objectname',
       ResponseContentType : 'image/jpeg'
}, (err, result) => {
       if(err){
              console.error('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
              if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                  // Obtain the rewritten response headers.
                  console.log(result.InterfaceResult.ContentType);
              }
       }
});
```

6.8 Obtaining Customized Metadata

NOTICE

If you have any questions during 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 customized data is returned. Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using 
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

// Download the object to obtain the object's customized metadata.
obsClient.getObject({
       Bucket : 'bucketname',
       Key : 'objectname',
}, (err, result) => {
       if(err){
              console.error('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
              if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                  console.log('Metadata-->' + JSON.stringify(result.InterfaceResult.Metadata['property']));
              }
       }
});
```

6.9 Downloading Object

NOTICE

If you have any questions during 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 Node.js SDK
Expedited restore	Data can be restored within 1 to 5 minutes.	ObsClient.enums.RestoreTierExpedited
Standard restore	Data can be restored within 3 to 5 hours. This is the default option.	ObsClient.enums.RestoreTierStandard

CAUTION

To prolong the validity period of the Archive data restored, you can repeatedly restore the Archive data, but you will be billed for each restore. 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 an Archive object. Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

// Restore an Archive object.
obsClient.restoreObject({
       Bucket : 'bucketname',
       Key : 'objectname',
       Days : 1,
       Tier : obsClient.enums.RestoreTierExpedited
}, (err, result) => {
       if(err){
              console.error('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);

// Wait for the object to be restored.
              setTimeout(()=>{

                     // Download the object to obtain the object content.
                     obsClient.getObject({
                           Bucket : 'bucketname',
                           Key : 'objectname'
                     }, (err, result) => {
                            if(err){
                               console.error('Error-->' + err);
                            }else{
                               console.log('Status-->' + result.CommonMsg.Status);
                               if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                                   // Obtain the object content.
                                   console.log('Object Content:');
                                   console.log(result.InterfaceResult.Content.toString());
                               }
                            }
                     });
              }, 6 * 60 * 1000);
       }
});
```

NOTE

- The object specified in **ObsClient.restoreObject** must be in the OBS Archive storage class. Otherwise, an error will be reported when you call this API.
- Use the **Days** parameter to specify the retention period (from 1 to 30 days) of the restored objects and the **Tier** parameter to specify the time spent on restoring the objects.

6.10 Performing a Resumable Download

NOTICE

If you have any questions during 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 will be returned. Otherwise, an error is returned in callback function 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 main parameters involved in this API.

Parameter	Description
Bucket	(Mandatory) Bucket name
Key	(Mandatory) Object name
DownloadFile	Full path of the local directory to which the object is downloaded. If this parameter is null, the downloaded object is saved in the directory where the program is executed.
PartSize	Part size, in bytes. The value ranges from 100 KB to 5 GB and defaults to 5 MB.
TaskNum	Maximum number of threads that can be concurrently executed for download. The default value is **20**.

Parameter	Description
EnableCheckpoint	Whether to enable the resumable download mode. The default value is **false**, which indicates that this mode is disabled.
CheckpointFile	File used to record the download progress. This parameter is effective only in the resumable download mode. If this parameter is null, the file will be in the same local directory as the downloaded object.
VersionId	Object version
IfModifiedSince	Returns the object if it has been modified since the specified time; otherwise, an error is returned.
IfUnmodifiedSince	Returns the object if it has not been modified since the specified time; otherwise, an error is returned.
IfMatch	Returns the source object if its ETag is the same as the one specified by this parameter; otherwise, an error code is returned.
IfNoneMatch	Returns the source object if its ETag is different from the one specified by this parameter; otherwise, an error code is returned.

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.downloadFile({
       Bucket : 'bucketname',
       Key : 'objectname',
      // Set the local path to which the object is downloaded.
       DownloadFile : 'localfile',
       // Set the part size to 10 MB.
       PartSize : 10 * 1024 * 1024,
       // Enable the resumable download mode.
       EnableCheckpoint : true
}, (err, result) => {
    if(err){
        console.error('Error-->' + err);
    }else{
        console.log('RequestId-->' + result.InterfaceResult.RequestId);
        console.log('LastModified-->' + result.InterfaceResult.LastModified);
        console.log('Metadata-->' + JSON.stringify(result.InterfaceResult.Metadata));
    }
});
```

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, this API degrades to the simple encapsulation of partial download, and no checkpoint file will be generated.
- **CheckpointFile** is effective only when **EnableCheckpoint** is **true**.

7 Object Management

7.1 Obtaining Object Properties

NOTICE

If you have any questions during 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 properties of an object, including the length, MIME type, customized metadata. Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.getObjectMetadata({
       Bucket : 'bucketname',
       // prefix: Optional. It indicates the name of the folder before the object.
       Key : '[prefix/]objectname'
}, (err, result) => {
       if(err){
          console.error('Error-->' + err);
       }else{
          console.log('Status-->' + result.CommonMsg.Status);
          if(result.CommonMsg.Status < 300 && result.InterfaceResult){
              console.log(result.InterfaceResult.ContentType);
              console.log(result.InterfaceResult.ContentLength);
              console.log(result.InterfaceResult.Metadata['property']);
          }
```

```
    }
});
```

7.2 Managing Object ACLs

NOTICE

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

Object ACLs, similar to bucket ACLs, support pre-defined access control policies and direct configuration. For details, see **Managing Bucket ACLs**.

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

1. Specify a pre-defined access control policy during object upload.
2. Call **ObsClient.setObjectAcl** to specify a pre-defined access control policy.
3. Call **ObsClient.setObjectAcl** to set the ACL directly.

Specifying a Pre-defined Access Control Policy During Object Upload

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
    //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
    access_key_id: process.env.ACCESS_KEY_ID,
    secret_access_key: process.env.SECRET_ACCESS_KEY,
    server : 'https://your-endpoint'
});

obsClient.putObject({
    Bucket : 'bucketname',
    Key : 'objectname',
    Body : 'Hello OBS',
    // Set the object ACL to public-read.
    ACL : obsClient.enums.AclPublicRead
}, (err, result) => {
    if(err){
       console.error('Error-->' + err);
    }else{
       console.log('Status-->' + result.CommonMsg.Status);
    }
});
```

Setting a Pre-defined Access Control Policy for an Object

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
      //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
      //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
      access_key_id: process.env.ACCESS_KEY_ID,
      secret_access_key: process.env.SECRET_ACCESS_KEY,
      server : 'https://your-endpoint'
});

obsClient.setObjectAcl({
      Bucket : 'bucketname',
      Key : 'objectname',
      // Set the object ACL to private.
      ACL : obsClient.enums.AclPrivate
}, (err, result) => {
      if(err){
         console.error('Error-->' + err);
      }else{
         console.log('Status-->' + result.CommonMsg.Status);
      }
});
```

NOTE

Use the **ACL** parameter to specify ACL for an object.

Directly Setting an Object ACL

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
      //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
      //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
      access_key_id: process.env.ACCESS_KEY_ID,
      secret_access_key: process.env.SECRET_ACCESS_KEY,
      server : 'https://your-endpoint'
});

obsClient.setObjectAcl({
      Bucket : 'bucketname',
      Key : 'objectname',
      // Set the object owner.
      Owner:{'ID':'ownerid'},
      Grants:[
           // Grant all permissions to a specified user.
          { Grantee : {Type : 'CanonicalUser',ID : 'userid'}, Permission : obsClient.enums.PermissionFullControl},
              // Grant the READ permission to all users.
          { Grantee: {Type : 'Group', URI : obsClient.enums.GroupAllUsers}, Permission :
obsClient.enums.PermissionRead},
      ]
}, (err, result) => {
      if(err){
          console.error('Error-->' + err);
      }else{
          console.log('Status-->' + result.CommonMsg.Status);
      }
});
```

NOTE

- Use the **Owner** parameter to specify the object owner and use the **Grants** parameter to grant permissions for authorized users.
- 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.
- OBS buckets support the following grantee group:
  - All users: ObsClient.enums.GroupAllUsers

Obtaining an Object ACL

You can call **ObsClient.getObjectAcl** to obtain an object ACL. Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
    //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
    access_key_id: process.env.ACCESS_KEY_ID,
    secret_access_key: process.env.SECRET_ACCESS_KEY,
    server : 'https://your-endpoint'
});

obsClient.getObjectAcl({
    Bucket : 'bucketname',
    Key : 'objectname'
}, (err, result) => {
    if(err){
        console.error('Error-->' + err);
    }else{
        console.log('Status-->' + result.CommonMsg.Status);
            if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                    console.log('Owner[ID]-->' + result.InterfaceResult.Owner.ID);
                    console.log('Owner[Name]-->' + result.InterfaceResult.Owner.Name);
                    for(let i=0;i<result.InterfaceResult.Grants.length;i++){
                        console.log('Grant[' + i + ']:');
                        console.log('Grantee[ID]-->' + result.InterfaceResult.Grants[i]['Grantee']['ID']);
                        console.log('Grantee[URI]-->' + result.InterfaceResult.Grants[i]['Grantee']['URI']);
                        console.log('Permission-->' + result.InterfaceResult.Grants[i]['Permission']);
                    }
            }
    }
});
```

7.3 Listing Objects

NOTICE

If you have any questions during 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
Prefix	Prefix that the object names to be listed must contain
Marker	Object name to start with when listing objects in a bucket. Objects are listed in the lexicographical order.
MaxKeys	Maximum number of objects to be listed in the response body. The value ranges from **1** to **1000**. If the specified value exceeds **1000**, only 1,000 objects are returned.
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** parameter 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.) 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.

Listing Objects in Simple Mode

The following sample code shows how to list objects in simple mode. A maximum of 1,000 objects can be returned.

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');
// Create an instance of ObsClient.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.listObjects({
       Bucket : 'bucketname'
}, (err, result) => {
       if(err){
          console.error('Error-->' + err);
       }else{
          console.log('Status-->' + result.CommonMsg.Status);
              if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                     for(let j=0;j<result.InterfaceResult.Contents.length;j++){
                               console.log('Contents[' + j +  ']:');
                               console.log('Key-->' + result.InterfaceResult.Contents[j]['Key']);
                               console.log('Owner[ID]-->' + result.InterfaceResult.Contents[j]['Owner']['ID']);
                        }
              }
       }
});
```

NOTE

- A maximum of 1,000 objects can be listed each time. If a bucket contains more than 1,000 objects, **InterfaceResult.IsTruncated** in the response is **true**, indicating not all objects were listed. In such case, you can use **InterfaceResult.NextMarker** to obtain the start position for next listing.
- If you want to obtain all objects in a specified bucket, you can use the paging mode for listing objects.

Listing Objects by Specifying the Number

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.listObjects({
       Bucket : 'bucketname',
// Set the number of objects to be listed to 100.
       MaxKeys : 100
}, (err, result) => {
       if(err){
          console.error('Error-->' + err);
       }else{
          console.log('Status-->' + result.CommonMsg.Status);
```

```
                if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                    for(let j=0;j<result.InterfaceResult.Contents.length;j++){
                          console.log('Contents[' + j +  ']:');
                          console.log('Key-->' + result.InterfaceResult.Contents[j]['Key']);
                          console.log('Owner[ID]-->' + result.InterfaceResult.Contents[j]['Owner']['ID']);
                    }
                }
       }
});
```

Listing Objects by Specifying a Prefix

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.listObjects({
       Bucket : 'bucketname',
       // Set the number to 100 and the prefix to prefix.
       MaxKeys : 100,
       Prefix : 'prefix'
}, (err, result) => {
       if(err){
          console.error('Error-->' + err);
       }else{
          console.log('Status-->' + result.CommonMsg.Status);
                if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                    for(let j=0;j<result.InterfaceResult.Contents.length;j++){
                          console.log('Contents[' + j +  ']:');
                          console.log('Key-->' + result.InterfaceResult.Contents[j]['Key']);
                          console.log('Owner[ID]-->' + result.InterfaceResult.Contents[j]['Owner']['ID']);
                     }
                }
       }
});
```

Listing Objects by Specifying the Start Position

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.listObjects({
       Bucket : 'bucketname',
// List 100 objects following test in lexicographic order.
       MaxKeys : 100,
       Marker : 'test'
}, (err, result) => {
       if(err){
          console.error('Error-->' + err);
       }else{
          console.log('Status-->' + result.CommonMsg.Status);
                 if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                        for(let j=0;j<result.InterfaceResult.Contents.length;j++){
                                  console.log('Contents[' + j +  ']:');
                                  console.log('Key-->' + result.InterfaceResult.Contents[j]['Key']);
                                  console.log('Owner[ID]-->' + result.InterfaceResult.Contents[j]['Owner']['ID']);
                         }
                 }
       }
});
```

Listing All Objects in Paging Mode

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

var listAll = (marker) => {
       obsClient.listObjects({
              Bucket : 'bucketname',
// Set the number of objects displayed per page to 100.
              MaxKeys : 100,
              Marker : marker
       }, (err, result) => {
              if(err){
                     console.error('Error-->' + err);
              }else{
                     console.log('Status-->' + result.CommonMsg.Status);
                     if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                            for(let j=0;j<result.InterfaceResult.Contents.length;j++){
                                  console.log('Contents[' + j +  ']:');
                                  console.log('Key-->' + result.InterfaceResult.Contents[j]['Key']);
                                  console.log('Owner[ID]-->' + result.InterfaceResult.Contents[j]['Owner']['ID']);

                            }
                            if(result.InterfaceResult.IsTruncated === 'true'){
                                  listAll(result.InterfaceResult.NextMarker);
                            }
                     }
              }
       });
};

listAll();
```

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:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

var listAll = (marker) => {
       obsClient.listObjects({
              Bucket : 'bucketname',
              MaxKeys : 1000,
              // Set the prefix of objects in the folder to dir/.
              Prefix : 'dir/'
       }, (err, result) => {
              if(err){
                     console.error('Error-->' + err);
              }else{
                     console.log('Status-->' + result.CommonMsg.Status);
                     if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                            for(let j=0;j<result.InterfaceResult.Contents.length;j++){
                                   console.log('Contents[' + j +  ']:');
                                   console.log('Key-->' + result.InterfaceResult.Contents[j]['Key']);
                                   console.log('Owner[ID]-->' + result.InterfaceResult.Contents[j]['Owner']['ID']);

                            }
                            if(result.InterfaceResult.IsTruncated === 'true'){
                                   listAll(result.InterfaceResult.NextMarker);
                            }
                     }
              }
       });
};

listAll();
```

Listing All Objects According to Folders in a Bucket

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
```

```
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using 
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.listObjects({
       Bucket: 'bucketname',
// Set folder isolators to slashes.
       Delimiter: '/'
}, (err, result) => {
       if(!err && result.CommonMsg.Status < 300){
              console.log('Objects in the root directory:');
              for(let j=0;j<result.InterfaceResult.Contents.length;j++){
                     console.log('\tKey-->' + result.InterfaceResult.Contents[j]['Key']);
                     console.log('Owner[ID]-->' + result.InterfaceResult.Contents[j]['Owner']
['ID']);
                     console.log('Owner[Name]-->' + result.InterfaceResult.Contents[j]['Owner']['Name']);
              }
              var listObjectsByPrefix = (commonPrefixes) => {
                     for(let i=0;i<commonPrefixes.length;i++){
                            obsClient.listObjects({
                                   Bucket: 'bucketname',
                                   Delimiter: '/',
                                   Prefix: commonPrefixes[i]['Prefix']
                            }, (err, result)=>{
                                   if(!err && result.CommonMsg.Status < 300){
                                          console.log('Objects in folder [' + commonPrefixes[i]['Prefix'] + ']:');
                                          for(let j=0;j<result.InterfaceResult.Contents.length;j++){
                                                 console.log('\tKey-->' + result.InterfaceResult.Contents[j]['Key']);
                                                 console.log('Owner[ID]-->' + result.InterfaceResult.Contents[j]['Owner']
['ID']);
                                          }
                                          console.log('\n');
                                          if(result.InterfaceResult.CommonPrefixes &&
result.InterfaceResult.CommonPrefixes.length > 0){
                                                 listObjectsByPrefix(result.InterfaceResult.CommonPrefixes);
                                          }
                                   }
                            });
                     }
              };
              listObjectsByPrefix(result.InterfaceResult.CommonPrefixes);
       }
});
```

NOTE

- The sample code does not apply to scenarios where the number of objects in a folder exceeds 1000.
- Because objects and sub-folders in a folder are to be listed and all the objects end with a slash (/), **Delimiter** is always a slash (/).
- In the returned result of each recursion, **InterfaceResult.Contents** includes the objects in the folder and **InterfaceResult.CommonPrefixes** includes the sub-folders in the folder.

7.4 Deleting Objects

NOTICE

If you have any questions during 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:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.deleteObject({
       Bucket: 'bucketname',
       Key : 'objectname'
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

Deleting Objects in a Batch

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:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
      //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
      //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
      access_key_id: process.env.ACCESS_KEY_ID,
      secret_access_key: process.env.SECRET_ACCESS_KEY,
      server : 'https://your-endpoint'
});

obsClient.deleteObjects({
      Bucket: 'bucketname',
     // Set the response mode to verbose.
      Quiet : false,
      Objects : [{Key:'objectname1'},{Key:'objectname2'}, {Key : 'objectname3'}]
}, (err, result) => {
      if(err){
            console.log('Error-->' + err);
      }else{
            console.log('Status-->' + result.CommonMsg.Status);
            if(result.CommonMsg.Status < 300 && result.InterfaceResult){
              // Obtain the list of successfully deleted objects.
              console.log('Deleteds:');
              for(let i=0;i<result.InterfaceResult.Deleteds.length;i++){
                    console.log('Deleted[' + i + ']:');
                    console.log('Key-->'+result.InterfaceResult.Deleteds[i]['Key']);
                    console.log('VersionId-->' + result.InterfaceResult.Deleteds[i]['VersionId']);
              }
              // Obtain information about objects that were not deleted.
              console.log('Errors:');
              for(let i=0;i<result.InterfaceResult.Errors.length;i++){
                    console.log('Error[' + i + ']:');
                    console.log('Key-->' + result.InterfaceResult.Errors[i]['Key']);
                    console.log('VersionId-->' + result.InterfaceResult.Errors[i]['VersionId']);
              }
            }
      }
});
```

NOTE

Use the **Quiet** parameter to specify the response mode and the **Objects** parameter to specify the to-be-deleted objects.

7.5 Copying an Object

NOTICE

If you have any questions during 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 rewrite properties and ACL for it, as well as set restriction conditions.

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:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.copyObject({
       Bucket: 'destbucketname',
       Key : 'destobjectname',
       CopySource:'sourcebucketname/sourceobjectname'
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

NOTE

Use the **CopySource** parameter to specify the information about the source object.

Rewriting Object Properties

The following sample code shows how to rewrite object properties.

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});
```

```
obsClient.copyObject({
    Bucket: 'destbucketname',
    Key : 'destobjectname',
    CopySource:'sourcebucketname/sourceobjectname',
    ContentType : 'image/jpeg',
    StorageClass : obsClient.enums.StorageClassWarm,
    Metadata:{'property':'property-value'},
    MetadataDirective : ObsClient.enums.ReplaceMetadata
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

NOTE

Use the **Metadata** parameter to specify the object's customized metadata to be rewritten and the **MetadataDirective** parameter to specify the rewrite mode, which can be **ObsClient.enums.ReplaceMetadata** (rewrite) or **ObsClient.enums.CopyMetadata** (copy from the source object).

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 error code will be returned and the copy will fail.

You can set the following conditions:

Parameter	Description	Format
CopySourceIfModified-Since	Copies the source object if it has been modified since the specified time; otherwise, an exception is thrown.	This parameter must conform to the HTTP time format specified in http://www.ietf.org/rfc/rfc2616.txt.
CopySourceIfUnmodi-fiedSince	Copies the source object if it has not been modified since the specified time; otherwise, an exception is thrown.	This parameter must conform to the HTTP time format specified in http://www.ietf.org/rfc/rfc2616.txt.
CopySourceIfMatch	Copies the source object if its ETag is the same as the one specified by this parameter; otherwise, an error code is returned.	Character string
CopySourceIfNoneMatch	Copies the source object if its ETag is different from the one specified by this parameter; otherwise, an error code is returned.	Character string

NOTE

- The ETag of the source object is the MD5 check value of the source object.
- If the object copy request includes **CopySourceIfUnmodifiedSince**, **CopySourceIfMatch**, **CopySourceIfModifiedSince**, or **CopySourceIfNoneMatch**, and the specified condition is not met, the object copy will fail with error code **412 Precondition Failed** returned.
- **CopySourceIfModifiedSince** and **CopySourceIfNoneMatch** can be used together. So do **CopySourceIfUnmodifiedSince** and **CopySourceIfMatch**.

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.copyObject({
    Bucket: 'destbucketname',
    Key : 'destobjectname',
    CopySource:'sourcebucketname/sourceobjectname',
    CopySourceIfModifiedSince : 'Thu, 31 Dec 2015 16:00:00 GMT',
    CopySourceIfNoneMatch : 'none-match-etag'
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

Rewriting an Object ACL

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.copyObject({
    Bucket: 'destbucketname',
    Key : 'destobjectname',
    CopySource:'sourcebucketname/sourceobjectname',
    // Rewrite the Object ACL to public-read.
    ACL : obsClient.enums.AclPublicRead
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

NOTE

Use the **ACL** parameter to modify the object ACL.

8 Temporarily Authorized Access

8.1 Using a Temporary URL for Authorized Access

NOTICE

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

NOTE

Use the **Method** parameter to specify the HTTP request method, the **Expires** parameter to specify the validity period (in seconds) of the URL, the **Headers** parameter to specify the request headers, the **SpecialParam** parameter to specify the special operators, and the **QueryParams** parameter to specify the query parameters.

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	Special Operator (Sub-resource)	Bucket Name Required	Object Name Required
PUT Bucket	PUT	N/A	Yes	No

Operation	HTTP Request Method	Special Operator (Sub-resource)	Bucket Name Required	Object Name Required
GET Buckets	GET	N/A	No	No
DELETE Bucket	DELETE	N/A	Yes	No
GET Objects	GET	N/A	Yes	No
GET Object versions	GET	versions	Yes	No
List Multipart uploads	GET	uploads	Yes	No
Obtain Bucket Metadata	HEAD	N/A	Yes	No
GET Bucket location	GET	location	Yes	No
GET Bucket storageinfo	GET	storageinfo	Yes	No
PUT Bucket quota	PUT	quota	Yes	No
GET Bucket quota	GET	quota	Yes	No
Set Bucket storagePolicy	PUT	storagePolicy	Yes	No
GET Bucket storagePolicy	GET	storagePolicy	Yes	No
PUT Bucket acl	PUT	acl	Yes	No
GET Bucket acl	GET	acl	Yes	No
PUT Bucket logging	PUT	logging	Yes	No
GET Bucket logging	GET	logging	Yes	No
PUT Bucket policy	PUT	policy	Yes	No
GET Bucket policy	GET	policy	Yes	No
DELETE Bucket policy	DELETE	policy	Yes	No
PUT Bucket lifecycle	PUT	lifecycle	Yes	No
GET Bucket lifecycle	GET	lifecycle	Yes	No

Operation	HTTP Request Method	Special Operator (Sub-resource)	Bucket Name Required	Object Name Required
DELETE Bucket lifecycle	DELETE	lifecycle	Yes	No
PUT Bucket website	PUT	website	Yes	No
GET Bucket website	GET	website	Yes	No
DELETE Bucket website	DELETE	website	Yes	No
PUT Bucket versioning	PUT	versioning	Yes	No
GET Bucket versioning	GET	versioning	Yes	No
PUT Bucket cors	PUT	cors	Yes	No
GET Bucket cors	GET	cors	Yes	No
DELETE Bucket cors	DELETE	cors	Yes	No
PUT Bucket tagging	PUT	tagging	Yes	No
GET Bucket tagging	GET	tagging	Yes	No
DELETE Bucket tagging	DELETE	tagging	Yes	No
PUT Object	PUT	N/A	Yes	Yes
Append Object	POST	append	Yes	Yes
GET Object	GET	N/A	Yes	Yes
PUT Object - Copy	PUT	N/A	Yes	Yes
DELETE Object	DELETE	N/A	Yes	Yes
DELETE Objects	POST	delete	Yes	Yes
Obtain Object Metadata	HEAD	N/A	Yes	Yes
PUT Object acl	PUT	acl	Yes	Yes
GET Object acl	GET	acl	Yes	Yes

Operation	HTTP Request Method	Special Operator (Sub-resource)	Bucket Name Required	Object Name Required
Initiate Multipart Upload	POST	uploads	Yes	Yes
PUT Part	PUT	N/A	Yes	Yes
PUT Part - Copy	PUT	N/A	Yes	Yes
List Parts	GET	N/A	Yes	Yes
Complete Multipart Upload	POST	N/A	Yes	Yes
DELETE Multipart upload	DELETE	N/A	Yes	Yes
POST Object restore	POST	restore	Yes	Yes

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

Step 1 Call **ObsClient.createSignedUrlSync** 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 set incorrectly 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

```
// Import the OBS library.
// Use npm to install the client.
```

```
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');
var https = require('https');
var urlLib = require('url');
var crypto = require('crypto');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

let bucketName = 'bucketname';
let method = 'PUT';
let res = obsClient.createSignedUrlSync({Method : method, Bucket : bucketName});
let location = 'your-location';
let content  = `<CreateBucketConfiguration><LocationConstraint>${location}</LocationConstraint></
CreateBucketConfiguration>`;

// Make a PUT request to create a bucket.
var url = urlLib.parse(res.SignedUrl);
var req = https.request({
       method : method,
       host : url.hostname,
       port : url.port,
       path : url.path,
       rejectUnauthorized : false,
       headers : res.ActualSignedRequestHeaders || {}
});


console.log('Creating bucket using url:' + res.SignedUrl);

req.on('response',   (serverback) => {
       var buffers = [];
       serverback.on('data', (data) => {
              buffers.push(data);
       }).on('end', () => {

              if(serverback.statusCode < 300){
                     console.log('Creating bucket using temporary signature succeed.');
              }else{
                     console.log('Creating bucket using temporary signature failed!');
                     console.log('status:' + serverback.statusCode);
                     console.log('\n');
              }
              buffers = Buffer.concat(buffers);
              if(buffers.length > 0){
                     console.log(buffers.toString());
              }
              console.log('\n');
       });
}).on('error',(err) => {
       console.log('Creating bucket using temporary signature failed!');
       console.log(err);
       console.log('\n');
});

if(content){
       req.write(content);
}
req.end();
```

Uploading an Object

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');
var https = require('https');
var urlLib = require('url');
var crypto = require('crypto');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using 
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

let bucketName = 'bucketname';
let objectKey = 'objectname';
let method = 'PUT';
let res = obsClient.createSignedUrlSync({Method : method, Bucket : bucketName, Key: objectKey, Expires: 
3600});
let content  = 'Hello OBS';

//Make a PUT request to upload an object.
var url = urlLib.parse(res.SignedUrl);
var req = https.request({
       method : method,
       host : url.hostname,
       port : url.port,
       path : url.path,
       rejectUnauthorized : false,
       headers : res.ActualSignedRequestHeaders || {}
});


console.log('Creating object using url:' + res.SignedUrl);

req.on('response',   (serverback) => {
       var buffers = [];
       serverback.on('data', (data) => {
              buffers.push(data);
       }).on('end', () => {

              if(serverback.statusCode < 300){
                     console.log('Creating object using temporary signature succeed.');
              }else{
                     console.log('Creating object using temporary signature failed!');
                     console.log('status:' + serverback.statusCode);
                     console.log('\n');
              }
              buffers = Buffer.concat(buffers);
              if(buffers.length > 0){
                     console.log(buffers.toString());
              }
              console.log('\n');
       });
}).on('error',(err) => {
       console.log('Creating object using temporary signature failed!');
       console.log(err);
       console.log('\n');
});

if(content){
       req.write(content);
}
req.end();
```

Downloading an Object

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');
var https = require('https');
var urlLib = require('url');
var crypto = require('crypto');

// Create an ObsClient instance.
var obsClient = new ObsClient({
      //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using 
hard coding may result in leakage.
      //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
      access_key_id: process.env.ACCESS_KEY_ID,
      secret_access_key: process.env.SECRET_ACCESS_KEY,
      server : 'https://your-endpoint'
});

let bucketName = 'bucketname';
let objectKey = 'objectname';
let method = 'GET';
let res = obsClient.createSignedUrlSync({Method : method, Bucket : bucketName, Key: objectKey, Expires: 
3600});

//Make a GET request to download an object.
var url = urlLib.parse(res.SignedUrl);
var req = https.request({
      method : method,
      host : url.hostname,
      port : url.port,
      path : url.path,
      rejectUnauthorized : false,
      headers : res.ActualSignedRequestHeaders || {}
});


console.log('Creating object using url:' + res.SignedUrl);

req.on('response',   (serverback) => {
      var buffers = [];
      serverback.on('data', (data) => {
            buffers.push(data);
      }).on('end', () => {

            if(serverback.statusCode < 300){
                  console.log('Getting object using temporary signature succeed.');
            }else{
                  console.log('Getting object using temporary signature failed!');
                  console.log('status:' + serverback.statusCode);
                  console.log('\n');
            }
            buffers = Buffer.concat(buffers);
            if(buffers.length > 0){
                  console.log(buffers.toString());
            }
            console.log('\n');
      });
}).on('error',(err) => {
      console.log('Getting object using temporary signature failed!');
      console.log(err);
      console.log('\n');
});
```

```
req.end();
```

Listing Objects

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');
var https = require('https');
var urlLib = require('url');
var crypto = require('crypto');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

let bucketName = 'bucketname';
let method = 'GET';
let res = obsClient.createSignedUrlSync({Method : method, Bucket : bucketName, Expires: 3600});

// Make a GET request to obtain the object list.
var url = urlLib.parse(res.SignedUrl);
var req = https.request({
       method : method,
       host : url.hostname,
       port : url.port,
       path : url.path,
       rejectUnauthorized : false,
       headers : res.ActualSignedRequestHeaders || {}
});


console.log('Listing object using url:' + res.SignedUrl);

req.on('response',   (serverback) => {
       var buffers = [];
       serverback.on('data', (data) => {
              buffers.push(data);
       }).on('end', () => {

              if(serverback.statusCode < 300){
                     console.log('Listing object using temporary signature succeed.');
              }else{
                     console.log('Listing object using temporary signature failed!');
                     console.log('status:' + serverback.statusCode);
                     console.log('\n');
              }
              buffers = Buffer.concat(buffers);
              if(buffers.length > 0){
                     console.log(buffers.toString());
              }
              console.log('\n');
       });
}).on('error',(err) => {
       console.log('Listing object using temporary signature failed!');
       console.log(err);
       console.log('\n');
});

req.end();
```

Deleting an Object

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');
var https = require('https');
var urlLib = require('url');
var crypto = require('crypto');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

let bucketName = 'bucketname';
let objectKey = 'objectname';
let method = 'DELETE';
let res = obsClient.createSignedUrlSync({Method : method, Bucket : bucketName, Key: objectKey, Expires: 3600});

// Make a DELETE request to delete the object.
var url = urlLib.parse(res.SignedUrl);
var req = https.request({
       method : method,
       host : url.hostname,
       port : url.port,
       path : url.path,
       rejectUnauthorized : false,
       headers : res.ActualSignedRequestHeaders || {}
});


console.log('Deleting object using url:' + res.SignedUrl);

req.on('response',   (serverback) => {
       var buffers = [];
       serverback.on('data', (data) => {
              buffers.push(data);
       }).on('end', () => {

              if(serverback.statusCode < 300){
                     console.log('Deleting object using temporary signature succeed.');
              }else{
                     console.log('Deleting object using temporary signature failed!');
                     console.log('status:' + serverback.statusCode);
                     console.log('\n');
              }
              buffers = Buffer.concat(buffers);
              if(buffers.length > 0){
                     console.log(buffers.toString());
              }
              console.log('\n');
       });
}).on('error',(err) => {
       console.log('Deleting object using temporary signature failed!');
       console.log(err);
       console.log('\n');
});

req.end();
```

9 Versioning Management

9.1 Versioning Overview

NOTICE

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

OBS can store multiple versions of an object. You can quickly search for and restore different versions as well as restore data in the event of misoperations or application faults.

For more information, see **Versioning**.

9.2 Setting Versioning Status for a Bucket

NOTICE

If you have any questions during 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 on the OBS Server
Enabled	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.	Enabled
Suspended	1. Noncurrent object versions are not affected. 2. OBS creates version ID **null** to an uploaded object and the object will be overwritten after a namesake one is uploaded 3. Objects can be downloaded by specifying the version ID. By default, the latest object is downloaded if no version ID is specified. 4. Objects can be deleted by specifying version IDs. If an object is deleted with no version ID specified, the object is only attached with a delete marker whose version ID is **null**. Objects with version ID **null** are physically deleted. 5. Except for delete markers, storage space occupied by objects with all version IDs is billed.	Suspended

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
    //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
    access_key_id: process.env.ACCESS_KEY_ID,
    secret_access_key: process.env.SECRET_ACCESS_KEY,
    server : 'https://your-endpoint'
});

// Enabling versioning.
obsClient.setBucketVersioning({
    Bucket : 'bucketname',
    VersionStatus : 'Enabled'
}, (err, result) => {
    if(err){
        console.log('Error-->' + err);
    }else{
        console.log('Status-->' + result.CommonMsg.Status);
    }
});

// Suspend versioning.
obsClient.setBucketVersioning({
    Bucket : 'bucketname',
    VersionStatus : 'Suspended'
}, (err, result) => {
    if(err){
        console.log('Error-->' + err);
    }else{
        console.log('Status-->' + result.CommonMsg.Status);
    }
});
```

NOTE

Use the **VersionStatus** parameter to specify the versioning status of a bucket.

9.3 Viewing Versioning Status of a Bucket

NOTICE

If you have any questions during 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. Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
    //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
    access_key_id: process.env.ACCESS_KEY_ID,
    secret_access_key: process.env.SECRET_ACCESS_KEY,
    server : 'https://your-endpoint'
});

// Enable versioning.
obsClient.getBucketVersioning({
       Bucket : 'bucketname'
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
              if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                  console.log('VersionStatus-->' + result.InterfaceResult.VersionStatus);
              }
       }
});
```

9.4 Obtaining a Versioning Object

NOTICE

If you have any questions during 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** and specify the **VersionId** parameter to obtain a versioning object. Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

// Specify versionId to obtain versioning objects.
obsClient.getObject({
       Bucket : 'bucketname',
       Key : 'objectname',
       VersionId : 'versionid'
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
              if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                  console.log('Content-->' + result.InterfaceResult.Content.toString());
              }
       }
});
```

NOTE

If **VersionId** is null, the object of the latest version will be downloaded, by default.

9.5 Copying a Versioning Object

NOTICE

If you have any questions during 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** and set **versionId** in the **CopySource** parameter to copy a versioning object. Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.copyObject({
       Bucket : 'destbucketname',
       Key : 'destobjectname',
       // Set the version ID of the object to be copied.
       CopySource : 'sourcebucket/sourceobjectname?versionId=versionid'
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

9.6 Restoring a Specific Archive Object Version

NOTICE

If you have any questions during 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 a version of an Archive object by specifying **VersionId**. Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
```

```
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using 
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.restoreObject({
       Bucket : 'bucketname',
       Key : 'objectname',
       VersionId : 'versionid',
       Days : 1,
       // Restore a versioned object at an expedited speed.
       Tier : obsClient.enums.RestoreTierExpedited
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

CAUTION

To prolong the validity period of the Archive data restored, you can repeatedly restore the Archive data, but you will be billed for each restore. 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.

9.7 Listing Versioning Objects

NOTICE

If you have any questions during 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
Prefix	Prefix that the object names to be listed must contain.
KeyMarker	Versioning object name to start with when listing versioning objects in a bucket. All versioning objects are listed in the lexicographical order.

Parameter	Description
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.
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** parameter 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.)
VersionIdMarker	All versioning objects are listed in the lexicographical order by object name and version ID. This parameter must be used together with **KeyMarker**.

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 1,000 versioning objects can be returned.

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.listVersions({
       Bucket : 'bucketname'
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
              if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                     // Obtain versioning objects.
                     for(let j=0;j<result.InterfaceResult.Versions.length;j++){
                            console.log('Version[' + j +  ']:');
                            console.log('Key-->' + result.InterfaceResult.Versions[j]['Key']);
                            console.log('VersionId-->' + result.InterfaceResult.Versions[j]['VersionId']);
                            console.log('Owner[ID]-->' + result.InterfaceResult.Versions[j]['Owner']['ID']);
```

```
            }
            // Obtain delete markers.
            for(let i=0;i<result.InterfaceResult.DeleteMarkers.length;i++){
                console.log('DeleteMarker[' + i +  ']:');
                console.log('Key-->' + result.InterfaceResult.DeleteMarkers[i]['Key']);
                console.log('VersionId-->' + result.InterfaceResult.DeleteMarkers[i]['VersionId']);
                console.log('Owner[ID]-->' + result.InterfaceResult.DeleteMarkers[i]['Owner']['ID']);
            }
        }
    }
});
```

NOTE

- A maximum of 1,000 object versions can be listed each time. If a bucket contains more than 1,000 object versions, **InterfaceResult.IsTruncated** in the response is **true**, indicating not all object versions were listed. In such case, you can use **InterfaceResult.NextKeyMarker** and **InterfaceResult.NextVersionIdMarker** to obtain the start position for the next listing.
- If you want to obtain all versioning objects in a specified bucket, you can use the paging mode for listing objects.

Listing Versioning Objects by Specifying the Number

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using 
hard coding may result in leakage.
    //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
    access_key_id: process.env.ACCESS_KEY_ID,
    secret_access_key: process.env.SECRET_ACCESS_KEY,
    server : 'https://your-endpoint'
});

obsClient.listVersions({
    Bucket : 'bucketname',
    MaxKeys : 100
}, (err, result) => {
    if(err){
        console.log('Error-->' + err);
    }else{
        console.log('Status-->' + result.CommonMsg.Status);
        if(result.CommonMsg.Status < 300 && result.InterfaceResult){
            // Obtain versioning objects.
            for(let j=0;j<result.InterfaceResult.Versions.length;j++){
                console.log('Version[' + j +  ']:');
                console.log('Key-->' + result.InterfaceResult.Versions[j]['Key']);
                console.log('VersionId-->' + result.InterfaceResult.Versions[j]['VersionId']);
                console.log('Owner[ID]-->' + result.InterfaceResult.Versions[j]['Owner']['ID']);
            }
            // Obtain delete markers.
            for(let i=0;i<result.InterfaceResult.DeleteMarkers.length;i++){
                console.log('DeleteMarker[' + i +  ']:');
                console.log('Key-->' + result.InterfaceResult.DeleteMarkers[i]['Key']);
                console.log('VersionId-->' + result.InterfaceResult.DeleteMarkers[i]['VersionId']);
                console.log('Owner[ID]-->' + result.InterfaceResult.DeleteMarkers[i]['Owner']['ID']);
            }
        }
    }
});
```

Listing Versioning Objects by Specifying a Prefix

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
    //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
    access_key_id: process.env.ACCESS_KEY_ID,
    secret_access_key: process.env.SECRET_ACCESS_KEY,
    server : 'https://your-endpoint'
});
;

// Set the prefix to prefix and the number to 100.
obsClient.listVersions({
    Bucket : 'bucketname',
    MaxKeys : 100,
    Prefix : 'prefix'
}, (err, result) => {
    if(err){
        console.log('Error-->' + err);
    }else{
        console.log('Status-->' + result.CommonMsg.Status);
        if(result.CommonMsg.Status < 300 && result.InterfaceResult){
            // Obtain versioning objects.
            for(let j=0;j<result.InterfaceResult.Versions.length;j++){
                console.log('Version[' + j +  ']:');
                console.log('Key-->' + result.InterfaceResult.Versions[j]['Key']);
                console.log('VersionId-->' + result.InterfaceResult.Versions[j]['VersionId']);
                console.log('Owner[ID]-->' + result.InterfaceResult.Versions[j]['Owner']['ID']);
            }
            // Obtain delete markers.
            for(let i=0;i<result.InterfaceResult.DeleteMarkers.length;i++){
                console.log('DeleteMarker[' + i +  ']:');
                console.log('Key-->' + result.InterfaceResult.DeleteMarkers[i]['Key']);
                console.log('VersionId-->' + result.InterfaceResult.DeleteMarkers[i]['VersionId']);
                console.log('Owner[ID]-->' + result.InterfaceResult.DeleteMarkers[i]['Owner']['ID']);
            }
        }
    }
});
```

Listing Versioning Objects by Specifying the Start Position

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
```

```
	//Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
	access_key_id: process.env.ACCESS_KEY_ID,
	secret_access_key: process.env.SECRET_ACCESS_KEY,
	server : 'https://your-endpoint'
});

// List 100 versioning objects whose names following test in lexicographic order.
obsClient.listVersions({
	Bucket : 'bucketname',
	MaxKeys : 100,
	KeyMarker : 'test'
}, (err, result) => {
	if(err){
		console.log('Error-->' + err);
	}else{
		console.log('Status-->' + result.CommonMsg.Status);
		if(result.CommonMsg.Status < 300 && result.InterfaceResult){
			// Obtain versioning objects.
			for(let j=0;j<result.InterfaceResult.Versions.length;j++){
				console.log('Version[' + j +  ']:');
				console.log('Key-->' + result.InterfaceResult.Versions[j]['Key']);
				console.log('VersionId-->' + result.InterfaceResult.Versions[j]['VersionId']);
				console.log('Owner[ID]-->' + result.InterfaceResult.Versions[j]['Owner']['ID']);
			}
// Obtain delete markers.
			for(let i=0;i<result.InterfaceResult.DeleteMarkers.length;i++){
				console.log('DeleteMarker[' + i +  ']:');
				console.log('Key-->' + result.InterfaceResult.DeleteMarkers[i]['Key']);
				console.log('VersionId-->' + result.InterfaceResult.DeleteMarkers[i]['VersionId']);
				console.log('Owner[ID]-->' + result.InterfaceResult.DeleteMarkers[i]['Owner']['ID']);
			}
		}
	}
});
```

Listing All Versioning Objects in Paging Mode

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
	//Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
	//Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
	access_key_id: process.env.ACCESS_KEY_ID,
	secret_access_key: process.env.SECRET_ACCESS_KEY,
	server : 'https://your-endpoint'
});

var listAll = (keyMarker, versionIdMarker) => {
	obsClient.listVersions({
		Bucket : 'bucketname',
		MaxKeys : 100,
		KeyMarker : keyMarker,
		VersionIdMarker : versionIdMarker
	}, (err, result) => {
		if(err){
			console.log('Error-->' + err);
		}else{
			console.log('Status-->' + result.CommonMsg.Status);
			if(result.CommonMsg.Status < 300 && result.InterfaceResult){
```

```
                // Obtain versioning objects.
                for(let j=0;j<result.InterfaceResult.Versions.length;j++){
                    console.log('Version[' + j +  ']:');
                    console.log('Key-->' + result.InterfaceResult.Versions[j]['Key']);
                    console.log('VersionId-->' + result.InterfaceResult.Versions[j]['VersionId']);
                    console.log('Owner[ID]-->' + result.InterfaceResult.Versions[j]['Owner']['ID']);
                }
                // Obtain delete markers.
                for(let i=0;i<result.InterfaceResult.DeleteMarkers.length;i++){
                    console.log('DeleteMarker[' + i +  ']:');
                    console.log('Key-->' + result.InterfaceResult.DeleteMarkers[i]['Key']);
                    console.log('VersionId-->' + result.InterfaceResult.DeleteMarkers[i]['VersionId']);
                    console.log('Owner[ID]-->' + result.InterfaceResult.DeleteMarkers[i]['Owner']['ID']);
                }
                if(result.InterfaceResult.IsTruncated === 'true'){
                    listAll(result.InterfaceResult.NextKeyMarker,
result.InterfaceResult.NextVersionIdMarker);
                }
            }
        }
    });
};

listAll();
```

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:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
    //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
    access_key_id: process.env.ACCESS_KEY_ID,
    secret_access_key: process.env.SECRET_ACCESS_KEY,
    server : 'https://your-endpoint'
});

var listAll = (keyMarker, versionIdMarker) => {
    obsClient.listVersions({
        Bucket : 'bucketname',
        MaxKeys : 100,
        KeyMarker : keyMarker,
        VersionIdMarker : versionIdMarker,
        // Set the prefix of objects in the folder to dir/.
        Prefix : 'dir/'
    }, (err, result) => {
        if(err){
            console.log('Error-->' + err);
        }else{
            console.log('Status-->' + result.CommonMsg.Status);
            if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                // Obtain versioning objects.
                for(let j=0;j<result.InterfaceResult.Versions.length;j++){
                    console.log('Version[' + j +  ']:');
                    console.log('Key-->' + result.InterfaceResult.Versions[j]['Key']);
                    console.log('VersionId-->' + result.InterfaceResult.Versions[j]['VersionId']);
```

```
                                console.log('Owner[ID]-->' + result.InterfaceResult.Versions[j]['Owner']['ID']);
                            }
// Obtain delete markers.
                            for(let i=0;i<result.InterfaceResult.DeleteMarkers.length;i++){
                                console.log('DeleteMarker[' + i +  ']:');
                                console.log('Key-->' + result.InterfaceResult.DeleteMarkers[i]['Key']);
                                console.log('VersionId-->' + result.InterfaceResult.DeleteMarkers[i]['VersionId']);
                                console.log('Owner[ID]-->' + result.InterfaceResult.DeleteMarkers[i]['Owner']['ID']);
                            }
                            if(result.InterfaceResult.IsTruncated === 'true'){
                                listAll(result.InterfaceResult.NextKeyMarker, 
result.InterfaceResult.NextVersionIdMarker);
                            }
                        }
                    }
        });
};

listAll();
```

Listing All Versioning Objects According to Folders in a Bucket

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using 
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.listVersions({
       Bucket : 'bucketname',
       // Set the folder isolators to slashes (/).
       Delimiter : '/'
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Objects in the root directory:');
              if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                  // Obtain versioning objects.
                  for(let j=0;j<result.InterfaceResult.Versions.length;j++){
                         console.log('Version[' + j +  ']:');
                         console.log('Key-->' + result.InterfaceResult.Versions[j]['Key']);
                         console.log('VersionId-->' + result.InterfaceResult.Versions[j]['VersionId']);
                         console.log('Owner[ID]-->' + result.InterfaceResult.Versions[j]['Owner']['ID']);
                  }
                  // Obtain delete markers.
                  for(let i=0;i<result.InterfaceResult.DeleteMarkers.length;i++){
                         console.log('DeleteMarker[' + i +  ']:');
                         console.log('Key-->' + result.InterfaceResult.DeleteMarkers[i]['Key']);
                         console.log('VersionId-->'+ result.InterfaceResult.DeleteMarkers[i]['VersionId']);
                         console.log('Owner[ID]-->' + result.InterfaceResult.DeleteMarkers[i]['Owner']['ID']);
                  }
              }
              var listVersionsByPrefix = (commonPrefixes) => {
                     for(let n=0;n<commonPrefixes.length;n++){
                           obsClient.listVersions({
```

```
                         Bucket : 'bucketname',
                         Delimiter : '/',
                         Prefix: commonPrefixes[n]['Prefix']
                    }, (err, result) => {
                         console.log('Objects in folder [' + commonPrefixes[n]['Prefix'] + ']:');
                         if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                           // Obtain versioning objects.
                           for(let j=0;j<result.InterfaceResult.Versions.length;j++){
                                console.log('Version[' + j +  ']:');
                                console.log('Key-->' + result.InterfaceResult.Versions[j]['Key']);
                                console.log('VersionId-->' + result.InterfaceResult.Versions[j]['VersionId']);
                                console.log('Owner[ID]-->' + result.InterfaceResult.Versions[j]['Owner']['ID']);
                           }
                           // Obtain delete markers.
                           for(let i=0;i<result.InterfaceResult.DeleteMarkers.length;i++){
                                console.log('DeleteMarker[' + i +  ']:');
                                console.log('Key-->' + result.InterfaceResult.DeleteMarkers[i]['Key']);
                                console.log('VersionId-->' + result.InterfaceResult.DeleteMarkers[i]
['VersionId']);
                                console.log('Owner[ID]-->' + result.InterfaceResult.DeleteMarkers[i]['Owner']
['ID']);
                           }
                           console.log('\n');
                           if(result.InterfaceResult.CommonPrefixes &&
result.InterfaceResult.CommonPrefixes.length > 0){
                                listVersionsByPrefix(result.InterfaceResult.CommonPrefixes);
                           }
                         }
                    });
               }
          };
          listVersionsByPrefix(result.InterfaceResult.CommonPrefixes);
     }
});
```

NOTE

- The previous sample code does not include scenarios where the number of objects in a folder exceeds 1000.
- Because objects and sub-folders in a folder are to be listed and all the objects end with a slash (/), **Delimiter** is always a slash (/).
- In the returned result of each recursion, **InterfaceResult.Versions** includes the versioning objects in the folder, **InterfaceResult.DeleteMarkers** includes the delete markers involved in the folder, and **InterfaceResult.CommonPrefixes** includes the sub-folders in the folder.

9.8 Setting or Obtaining a Versioning Object ACL

NOTICE

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

Directly Setting a Versioning Object ACL

You can call **ObsClient.setObjectAcl** to set the ACL for a versioning object by specifying the version ID (**VersionId**). Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
```

```
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.setObjectAcl({
       Bucket : 'bucketname',
       Key : 'objectname',
       VersionId : 'versionid',
       // Set the versioning object ACL to public-read by specifying the pre-defined access control policy.
       ACL : obsClient.enums.AclPublicRead
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});

obsClient.setObjectAcl({
       Bucket : 'bucketname',
       Key : 'objectname',
       VersionId : 'versionid',
       // Set the object owner.
       Owner:{'ID':'ownerid'},
       Grants:{
         Grant:[
                // Grant the READ permission to all users.
               { Grantee:{Type : 'Group', URI : obsClient.enums.GroupAllUsers}, Permission :
obsClient.enums.PermissionRead},
                // Grant the WRITE_ACP permission to all users.
               { Grantee:{Type : 'Group', URI : obsClient.enums.GroupAllUsers}, Permission :
obsClient.enums.PermissionWriteAcp}
         ]
       }
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

NOTE

- Use the **Owner** parameter to specify the object owner and use the **Grants** parameter to grant permissions for authorized users.
- 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.
- OBS buckets support the following grantee group:
  - All users: ObsClient.enums.GroupAllUsers

Obtaining a Versioning Object ACL

You can call **ObsClient.getObjectAcl** to obtain the ACL of a versioning object by specifying the version ID (**VersionId**). Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
    //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
    access_key_id: process.env.ACCESS_KEY_ID,
    secret_access_key: process.env.SECRET_ACCESS_KEY,
    server : 'https://your-endpoint'
});

obsClient.getObjectAcl({
    Bucket : 'bucketname',
    Key : 'objectname',
    VersionId : 'versionid'
}, (err, result) => {
    if(err){
        console.log('Error-->' + err);
    }else{
        console.log('Status-->' + result.CommonMsg.Status);
        if(result.CommonMsg.Status < 300 && result.InterfaceResult){
            console.log('Owner[ID]-->' + result.InterfaceResult.Owner.ID);
            for(let i=0;i<result.InterfaceResult.Grants.Grant.length;i++){
                console.log('Grant[' + i + ']:');
                console.log('Grantee[ID]-->' + result.InterfaceResult.Grants.Grant[i]['Grantee']['ID']);
                console.log('Grantee[URI]-->' + result.InterfaceResult.Grants.Grant[i]['Grantee']['URI']);
                console.log('Permission-->'+ result.InterfaceResult.Grants.Grant[i]['Permission']);
            }
        }
    }
});
```

9.9 Deleting Versioning Objects

NOTICE

If you have any questions during 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 delete a versioning object by specifying the version ID (**VersionId**). Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
    //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
    access_key_id: process.env.ACCESS_KEY_ID,
    secret_access_key: process.env.SECRET_ACCESS_KEY,
    server : 'https://your-endpoint'
});

obsClient.deleteObject({
    Bucket : 'bucketname',
    Key : 'objectname',
    VersionId : 'versionid'
}, (err, result) => {
    if(err){
            console.log('Error-->' + err);
    }else{
            console.log('Status-->' + result.CommonMsg.Status);
    }
});
```

Deleting Versioning Objects in a Batch

You can call **ObsClient.deleteObjects** to pass the version ID (**VersionId**) of each to-be-deleted object to delete them. Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
        //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using 
hard coding may result in leakage.
        //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
        access_key_id: process.env.ACCESS_KEY_ID,
        secret_access_key: process.env.SECRET_ACCESS_KEY,
        server : 'https://your-endpoint'
});

obsClient.deleteObjects({
        Bucket: 'bucketname',
        // Set the response mode to verbose.
        Quiet : false,
        Objects : [{Key:'objectname1', VersionId : 'version1'},{Key:'objectname2', VersionId : 'version2'}, {Key : 
'objectname3', VersionId : 'version3'}]
}, (err, result) => {
        if(err){
                console.log('Error-->' + err);
        }else{
                console.log('Status-->' + result.CommonMsg.Status);
                if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                        // Obtain information about successfully deleted objects.
                        console.log('Deleteds:');
                        for(let i=0;i<result.InterfaceResult.Deleteds.length;i++){
                                console.log('Deleted[' + i + ']:');
                                console.log('Key-->'+result.InterfaceResult.Deleteds[i]['Key']);
                                console.log('VersionId-->' + result.InterfaceResult.Deleteds[i]['VersionId']);
                        }
                        // Obtain information about objects that were not deleted.
                        console.log('Errors:');
                        for(let j=0;j<result.InterfaceResult.Errors.length;j++){
                                console.log('Error[' + j + ']:');
                                console.log('Key-->' + result.InterfaceResult.Errors[j]['Key']);
                                console.log('VersionId-->' + result.InterfaceResult.Errors[j]['VersionId']);
                        }
                }
```

```
    }
});
```

10 Lifecycle Management

10.1 Lifecycle Management Overview

NOTICE

If you have any questions during 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 based on the prefix. A lifecycle rule must contain:

- Rule ID, which uniquely identifies the rule
- Prefix of objects 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 an object of the latest version, which can be specified in either mode:
  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 expiration time and transition policy for a non-current object version will take effect only after versioning is enabled for buckets.

10.2 Setting Lifecycle Rules

NOTICE

If you have any questions during 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 for a bucket.

Setting an Object Transition Policy

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
      //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
      //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
      access_key_id: process.env.ACCESS_KEY_ID,
      secret_access_key: process.env.SECRET_ACCESS_KEY,
      server : 'https://your-endpoint'
});

obsClient.setBucketLifecycle({
      Bucket: 'bucketname',
      Rules:[
            {
                  ID:'rule1',Prefix:'prefix1',Status:'Enabled',
                  // Specify that objects whose names contain the specified prefix will be transitioned to OBS
Infrequent Access 30 days after creation.

                  Transitions:[{StorageClass: obsClient.enums.StorageClassWarm, Days:30}],
                 // Specify that objects whose names contain the specified prefix will be transitioned to OBS
Archive after being noncurrent for 30 days.

                  NoncurrentVersionTransitions:[{StorageClass: obsClient.enums.StorageClassCold,
NoncurrentDays : 30}]
            },
            {
                  ID:'rule2',Prefix:'prefix2',Status:'Enabled',
                  // Specify when objects whose names contain the specified prefix will be transitioned to OBS
Infrequent Access.

                    Transitions:[{StorageClass: obsClient.enums.StorageClassWarm, Date:
'2018-10-31T00:00:00Z'}],
             }
       ]
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

Setting an Object Expiration Time

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.setBucketLifecycle({
       Bucket: 'bucketname',
       Rules:[
              {
                     ID:'rule1',Prefix:'prefix1',Status:'Enabled',
                      // Specify that objects whose names contain the specified prefix will expire 60 days after
creation.
                     Expiration:{Days:60},
                      // Specify that objects whose names contain the specified prefix will expire after changing
into noncurrent versions for 60 days.
                     NoncurrentVersionExpiration:{NoncurrentDays : 60}
              },
              {
                     ID:'rule2',Prefix:'prefix2',Status:'Enabled',
                     // Specify when the objects whose names contain the specified prefix will expire. The value
must conform to the ISO8601 standards and must be at 00:00 (UTC time).
                     Expiration:{Date: '2018-12-31T00:00:00Z'},
              }
       ]
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

NOTE

Use the **Rules** parameter to specify the lifecycle rules for a bucket.

10.3 Viewing Lifecycle Rules

NOTICE

If you have any questions during 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 of a bucket. Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.getBucketLifecycle({
       Bucket: 'bucketname'
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
              if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                     for(let i=0;i<result.InterfaceResult.Rules.length;i++){
                            console.log('Rule[' + i + ']:');
                            console.log('ID-->' + result.InterfaceResult.Rules[i]['ID']);
                            console.log('Prefix-->' + result.InterfaceResult.Rules[i]['Prefix']);
                            console.log('Status-->' + result.InterfaceResult.Rules[i]['Status']);
                            for(let j=0;j<result.InterfaceResult.Rules[i]['Transitions'].length;j++){
                                   console.log('Transition[' + j + ']:');
                                   console.log('Transition[StorageClass]-->' + result.InterfaceResult.Rules[i]['Transitions']
[j]['StorageClass']);
                                   console.log('Transition[Date]-->' + result.InterfaceResult.Rules[i]['Transitions'][j]
['Date']);
                                   console.log('Transition[Days]-->' + result.InterfaceResult.Rules[i]['Transitions'][j]
['Days']);                          }
                            console.log('Expiration[Date]-->' + result.InterfaceResult.Rules[i]['Expiration']['Date']);
                            console.log('Expiration[Days]-->' + result.InterfaceResult.Rules[i]['Expiration']['Days']);
                            for(let k=0;k<result.InterfaceResult.Rules[i]['NoncurrentVersionTransitions'].length;k++){
                                   console.log('NoncurrentVersionTransition[' + k + ']:');
                                   console.log('NoncurrentVersionTransition[StorageClass]-->' +
result.InterfaceResult.Rules[i]['NoncurrentVersionTransitions'][k]['StorageClass']);
                                   console.log('NoncurrentVersionTransition[NoncurrentDays]-->' +
result.InterfaceResult.Rules[i]['NoncurrentVersionTransitions'][k]['NoncurrentDays']);                     }
                            console.log('NoncurrentVersionExpiration[NoncurrentDays]-->' +
result.InterfaceResult.Rules[i]['NoncurrentVersionExpiration']['NoncurrentDays']);
                     }
              }
       }
});
```

10.4 Deleting Lifecycle Rules

NOTICE

If you have any questions during 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 of a bucket. Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.deleteBucketLifecycle({
       Bucket: 'bucketname'
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

11 CORS

11.1 CORS Overview

NOTICE

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

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

For more information, see **CORS**.

11.2 Setting CORS Rules

NOTICE

If you have any questions during 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. Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
      //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
      //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
      access_key_id: process.env.ACCESS_KEY_ID,
      secret_access_key: process.env.SECRET_ACCESS_KEY,
      server : 'https://your-endpoint'
});

obsClient.setBucketCors({
      Bucket:'bucketname',
      CorsRules:[
          {
                  // Specify the request method, which can be GET, PUT, DELETE, POST, or HEAD.
                  AllowedMethod: ['GET','HEAD','PUT'],
                  // Specify the origin of the cross-domain request.
                  AllowedOrigin: ['http://www.a.com','http://www.b.com'],
// Specify whether headers specified in Access-Control-Request-Headers in the OPTIONS request can be
used.
                  AllowedHeader: ['x-obs-header'],
                  // Specify response headers that users can access using application programs.
                  ExposeHeader: ['x-obs-expose-header'],
                  // Specify the browser's cache time of the returned results of OPTIONS requests for specific
resources, in seconds.
                  MaxAgeSeconds: 10
            }
      ]
}, (err, result) => {
      if(err){
            console.log('Error-->' + err);
      }else{
            console.log('Status-->' + result.CommonMsg.Status);
      }
});
```

NOTE

- Use the **CorsRules** parameter to set CORS rules for a bucket.
- Both **AllowedOrigin** and **AllowedHeader** can contain up to one wildcard character (*). The wildcard character (*) indicates that all origins or headers are allowed.

11.3 Viewing CORS Rules

NOTICE

If you have any questions during 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. Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
      //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
```

```
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
       access_key_id: process.env.ACCESS_KEY_ID,
       secret_access_key: process.env.SECRET_ACCESS_KEY,
       server : 'https://your-endpoint'
});

obsClient.getBucketCors({
       Bucket:'bucketname'
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
              if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                     for(let k=0;k<result.InterfaceResult.CorsRule.length;k++){
                            console.log('CorsRule[',k,']');
                            console.log('CorsRule[MaxAgeSeconds]-->' + result.InterfaceResult.CorsRules[k]
['MaxAgeSeconds']);
                            for (let i=0;i<result.InterfaceResult.CorsRule[k]['AllowedMethod'].length;i++){
                                   console.log('CorsRule[AllowedMethod][' , i ,']-->'+result.InterfaceResult.CorsRules[k]
['AllowedMethod'][i]);
                            }
                            for(let j=0;j<result.InterfaceResult.CorsRule[k]['AllowedOrigin'].length;j++){
                                   console.log('CorsRule[AllowedOrigin][', j ,']-->'+result.InterfaceResult.CorsRules[k]
['AllowedOrigin'][j]);
                             }
                            for(let n=0;n<result.InterfaceResult.CorsRule[k]['AllowedHeader'].length;n++){
                                   console.log('CorsRule[AllowedHeader][', n ,']-->'+result.InterfaceResult.CorsRules[k]
['AllowedHeader'][n]);
                            }
                            for(let m=0;m<result.InterfaceResult.CorsRule[k]['ExposeHeader'].length;m++){
                                   console.log('CorsRule[ExposeHeader][', m ,']-->'+result.InterfaceResult.CorsRules[k]
['ExposeHeader'][m]);
                            }
                     }
              }
       }
});
```

11.4 Deleting CORS Rules

NOTICE

If you have any questions during 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. Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
var obsClient = new ObsClient({
       //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using
hard coding may result in leakage.
       //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
```

```
        access_key_id: process.env.ACCESS_KEY_ID,
        secret_access_key: process.env.SECRET_ACCESS_KEY,
        server : 'https://your-endpoint'
});

obsClient.deleteBucketCors({
        Bucket:'bucketname'
}, (err, result) => {
        if(err){
                console.log('Error-->' + err);
        }else{
                console.log('Status-->' + result.CommonMsg.Status);
        }
});
```

12 Access Logging

12.1 Logging Overview

NOTICE

If you have any questions during 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 specified buckets in OBS.

For more information, see **Logging**.

12.2 Enabling Bucket Logging

NOTICE

If you have any questions during 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 must be in the same region.
- Before configuring bucket logging, you need to create an agency for OBS on **IAM** and obtain the agency name. For details, see **Creating an IAM Agency**.

NOTE

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

Enabling Bucket Logging

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
const ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// const ObsClient = require('./lib/obs');

// Create an ObsClient instance.
const obsClient = new ObsClient({
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using hard coding may result in leakage.
    //Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
    access_key_id: process.env.ACCESS_KEY_ID,
    secret_access_key: process.env.SECRET_ACCESS_KEY,
    //EU-Dublin is used here as an example. Replace it with the one in your actual situation.
    server: 'https://obs.eu-west-101.myhuaweicloud.com'

});

// Configure logging for the bucket.
obsClient.setBucketLogging({
    Bucket:'bucketname',
    // Name of the agency created on IAM.
    Agency: 'Agency name',
    LoggingEnabled:{
        // Name of the bucket for storing the generated log file.
        TargetBucket: 'LogBucketName',
        // Specify the name prefix of the generated log file.
        TargetPrefix: 'logs/',
    }
}, (err, result) => {
    if(err){
        console.log('Error-->' + err);
    }else{
        console.log('Status-->' + result.CommonMsg.Status);
        console.log('RequestId-->' + result.CommonMsg.RequestId);
    }
});
```

NOTE

Use the **LoggingEnabled** parameter to configure logging for a bucket.

Setting ACLs for Objects to Be Logged

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
const obsClient = new ObsClient({
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using hard coding may result in leakage.
    //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
    access_key_id: process.env.ACCESS_KEY_ID,
    secret_access_key: process.env.SECRET_ACCESS_KEY,
    //EU-Dublin is used here as an example. Replace it with the one in your actual situation.
    server: 'https://obs.eu-west-101.myhuaweicloud.com'

});

// Configure logging for the bucket.
obsClient.setBucketLogging({
    Bucket:'bucketname',
    // Name of the OBS agency created by the owner of the target bucket on IAM.
    Agency: 'Agency name',
    LoggingEnabled:{
        // Name of the bucket for storing the generated log file.
        TargetBucket: 'LogBucketName',
        // Specify the name prefix of the generated log file.
        TargetPrefix: 'logs/',
        TargetGrants:[
            // Grant all users the READ permission on the logs.
           {Grantee:
{Type:'Group',URI:obsClient.enums.GroupAllUsers},Permission:obsClient.enums.PermissionRead},
            // Grant all users the WRITE permission on the logs.
           {Grantee:
{Type:'Group',URI:obsClient.enums.GroupAllUsers},Permission:obsClient.enums.PermissionWrite}
        ]
     }
}, (err, result) => {
    if(err){
        console.log('Error-->' + err);
    }else{
        console.log('Status-->' + result.CommonMsg.Status);
        console.log('RequestId-->' + result.CommonMsg.RequestId);
    }
});
```

12.3 Viewing Bucket Logging Settings

NOTICE

If you have any questions during 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 settings of a bucket. Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
const obsClient = new ObsClient({
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using hard
coding may result in leakage.
    //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
    access_key_id: process.env.ACCESS_KEY_ID,
    secret_access_key: process.env.SECRET_ACCESS_KEY,
    //EU-Dublin is used here as an example. Replace it with the one in your actual situation.
    server: 'https://obs.eu-west-101.myhuaweicloud.com'

});

obsClient.getBucketLogging({
       Bucket:'bucketname'
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
              if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                     if(result.InterfaceResult.LoggingEnabled){
                            console.log('TargetBucket-->' + result.InterfaceResult.LoggingEnabled.TargetBucket);
                            console.log('TargetPrefix-->' + result.InterfaceResult.LoggingEnabled.TargetPrefix);
                     }
                     for(let i=0;i<result.InterfaceResult.LoggingEnabled.TargetGrants.length;i++){
                            console.log('Grant[' + i + ']:');
                            console.log('Grantee[ID]-->' + result.InterfaceResult.LoggingEnabled.TargetGrants[i]
['Grantee']['ID']);
                            console.log('Grantee[URI]-->' + result.InterfaceResult.LoggingEnabled.TargetGrants[i]
['Grantee']['URI']);
                            console.log('Permission-->' + result.InterfaceResult.LoggingEnabled.TargetGrants[i]
['Permission']);
                     }
              }
       }
});
```

12.4 Disabling Bucket Logging

NOTICE

If you have any questions during 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 clear logging settings of a bucket so as to disable logging of the bucket. Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an ObsClient instance.
const obsClient = new ObsClient({
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using hard
coding may result in leakage.
    //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
    access_key_id: process.env.ACCESS_KEY_ID,
    secret_access_key: process.env.SECRET_ACCESS_KEY,
    //EU-Dublin is used here as an example. Replace it with the one in your actual situation.
    server: 'https://obs.eu-west-101.myhuaweicloud.com'

});

obsClient.setBucketLogging({
       Bucket:'bucketname',
       LoggingEnabled : {}
}, (err, result) => {
       if(err){
               console.log('Error-->' + err);
       }else{
               console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

13 Static Website Hosting

13.1 Static Website Hosting Overview

NOTICE

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

You can upload the content files of the static website to your bucket in OBS as objects and configure the **public-read** permission on the files, and then configure the static website hosting mode for your bucket to host your static websites in OBS. 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**.

13.2 Website File Hosting

NOTICE

If you have any questions during 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 ACL for the objects to **public-read**.

Step 3 Access the objects using a browser.

----End

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
const obsClient = new ObsClient({
   //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using hard
coding may result in leakage.
   //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
    access_key_id: process.env.ACCESS_KEY_ID,
    secret_access_key: process.env.SECRET_ACCESS_KEY,
    //EU-Dublin is used here as an example. Replace it with the one in your actual situation.
    server: 'https://obs.eu-west-101.myhuaweicloud.com'

});

// Upload an object.
obsClient.putObject({
       Bucket: 'bucketname',
       Key: 'test.html',
       Body: '<html><header></header><body><h1>Hello OBS</h1></body></html>',
// Set the MIME type for the object.
       ContentType: 'text/html',
// Set the object ACL to public-read.
       ACL: obsClient.enums.AclPublicRead
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

NOTE

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

13.3 Setting Website Hosting

NOTICE

If you have any questions during 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 for a bucket.

Configuring the Default Homepage and Error Pages

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
const obsClient = new ObsClient({
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using hard
coding may result in leakage.
    //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
    access_key_id: process.env.ACCESS_KEY_ID,
    secret_access_key: process.env.SECRET_ACCESS_KEY,
    //EU-Dublin is used here as an example. Replace it with the one in your actual situation.
    server: 'https://obs.eu-west-101.myhuaweicloud.com'

});

obsClient.setBucketWebsite({
       Bucket: 'bucketname',
       // Configure the default homepage.
       IndexDocument:{Suffix:'index.html'},
       // Configure the error pages.
       ErrorDocument:{Key:'error.html'}
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

Configuring the Redirection Rules

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
const obsClient = new ObsClient({
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using hard
coding may result in leakage.
    //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
    access_key_id: process.env.ACCESS_KEY_ID,
    secret_access_key: process.env.SECRET_ACCESS_KEY,
    //EU-Dublin is used here as an example. Replace it with the one in your actual situation.
    server: 'https://obs.eu-west-101.myhuaweicloud.com'

});

obsClient.setBucketWebsite({
       Bucket: 'bucketname',
       // Configure the default homepage.
       IndexDocument:{Suffix:'index.html'},
       // Configure the error pages.
       ErrorDocument:{Key:'error.html'},
        // Configure the redirection rules.
       RoutingRules:[
           {
               Condition:{HttpErrorCodeReturnedEquals:'404', KeyPrefixEquals: 'keyprefix'},
               Redirect:{Protocol:'http',ReplaceKeyWith:'replacekeyprefix', HostName: 'www.example.com',
HttpRedirectCode: '305'}
           }
    ]
}, (err, result) => {
    if(err){
        console.log('Error-->' + err);
    }else{
        console.log('Status-->' + result.CommonMsg.Status);
    }
});
```

NOTE

Use the **RoutingRules** parameter to set redirection rules for a bucket.

Configuring Redirection for All Requests

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
const obsClient = new ObsClient({
  //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using hard coding may result in leakage.
  //Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
  access_key_id: process.env.ACCESS_KEY_ID,
  secret_access_key: process.env.SECRET_ACCESS_KEY,
  //EU-Dublin is used here as an example. Replace it with the one in your actual situation.
  server: 'https://obs.eu-west-101.myhuaweicloud.com'

});

obsClient.setBucketWebsite({
    Bucket: 'bucketname',
    RedirectAllRequestsTo : {HostName : 'www.example.com', Protocol : 'http'}
}, (err, result) => {
    if(err){
        console.log('Error-->' + err);
    }else{
        console.log('Status-->' + result.CommonMsg.Status);
    }
});
```

NOTE

Use the **RedirectAllRequestsTo** parameter to set redirection rules for all requests for accessing a bucket.

13.4 Viewing Website Hosting Settings

NOTICE

If you have any questions during 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. Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
const obsClient = new ObsClient({
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using hard
coding may result in leakage.
    //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
    access_key_id: process.env.ACCESS_KEY_ID,
    secret_access_key: process.env.SECRET_ACCESS_KEY,
    //EU-Dublin is used here as an example. Replace it with the one in your actual situation.
    server: 'https://obs.eu-west-101.myhuaweicloud.com'

});

obsClient.getBucketWebsite({
       Bucket: 'bucketname'
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
              if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                     console.log('RedirectAllRequestsTo:');
                     console.log('HostName-->' + result.InterfaceResult.RedirectAllRequestsTo['HostName']);
                     console.log('Protocol-->' + result.InterfaceResult.RedirectAllRequestsTo['Protocol']);
                     console.log('IndexDocument[Suffix]-->' + result.InterfaceResult.IndexDocument['Suffix']);
                     console.log('ErrorDocument[Key]-->' + result.InterfaceResult.ErrorDocument['Key']);
                     console.log('RoutingRules:');
                     for(let i=0;i<result.InterfaceResult.RoutingRules;i++){
                           console.log('RoutingRule[' + i + ']:');
                           let RoutingRule = result.InterfaceResult.RoutingRules[i];
                           console.log('Condition[HttpErrorCodeReturnedEquals]-->' + RoutingRule['Condition']
['HttpErrorCodeReturnedEquals']);
                            console.log('Condition[KeyPrefixEquals]-->' + RoutingRule['Condition']
['KeyPrefixEquals']);
                           console.log('Redirect[HostName]-->' + RoutingRule['Redirect']['HostName']);
                           console.log('Redirect[HttpRedirectCode]-->' + RoutingRule['Redirect']
['HttpRedirectCode']);
                           console.log('Redirect[Protocol]-->' + RoutingRule['Redirect']['Protocol']);
                           console.log('Redirect[ReplaceKeyPrefixWith]-->' + RoutingRule['Redirect']
['ReplaceKeyPrefixWith']);
                           console.log('Redirect[ReplaceKeyWith]-->' + RoutingRule['Redirect']['ReplaceKeyWith']);
                     }
              }
       }
});
```

13.5 Deleting Website Hosting Settings

NOTICE

If you have any questions during 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. Sample code is as follows:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
const obsClient = new ObsClient({
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using hard
coding may result in leakage.
    //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
    access_key_id: process.env.ACCESS_KEY_ID,
    secret_access_key: process.env.SECRET_ACCESS_KEY,
    //EU-Dublin is used here as an example. Replace it with the one in your actual situation.
    server: 'https://obs.eu-west-101.myhuaweicloud.com'

});

obsClient.deleteBucketWebsite({
       Bucket: 'bucketname'
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

14 Tag Management

14.1 Tagging Overview

NOTICE

If you have any questions during 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.

14.2 Setting Bucket Tags

NOTICE

If you have any questions during 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:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
const obsClient = new ObsClient({
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using hard coding may result in leakage.
    //Obtain an AK/SK pair on the management console. For details, see https://support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
    access_key_id: process.env.ACCESS_KEY_ID,
    secret_access_key: process.env.SECRET_ACCESS_KEY,
    //EU-Dublin is used here as an example. Replace it with the one in your actual situation.
    server: 'https://obs.eu-west-101.myhuaweicloud.com'

});

obsClient.setBucketTagging({
       Bucket: 'bucketname',
       Tags : [
           {Key:'tag1',Value:'value1'},
           {Key:'tag2',Value:'value2'}
       ]
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

NOTE

- Use the **Tags** parameter to specify tags for a bucket.
- A bucket can have up to 10 tags.
- The key and value of a tag can be composed of Unicode characters.

14.3 Viewing Bucket Tags

NOTICE

If you have any questions during 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:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
const obsClient = new ObsClient({
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using hard
coding may result in leakage.
    //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
    access_key_id: process.env.ACCESS_KEY_ID,
    secret_access_key: process.env.SECRET_ACCESS_KEY,
    //EU-Dublin is used here as an example. Replace it with the one in your actual situation.
    server: 'https://obs.eu-west-101.myhuaweicloud.com'

});

obsClient.getBucketTagging({
       Bucket: 'bucketname'
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
```

```
            if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                  result.InterfaceResult.Tags.forEach((tag) => {
                        console.log('Tag-->' + tag.Key + ':' + tag.Value);
                  });
            }
      }
});
```

14.4 Deleting Bucket Tags

NOTICE

If you have any questions during 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:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
const obsClient = new ObsClient({
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using hard
coding may result in leakage.
    //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
    access_key_id: process.env.ACCESS_KEY_ID,
    secret_access_key: process.env.SECRET_ACCESS_KEY,
    //EU-Dublin is used here as an example. Replace it with the one in your actual situation.
    server: 'https://obs.eu-west-101.myhuaweicloud.com'

});

obsClient.deleteBucketTagging({
      Bucket: 'bucketname'
}, (err, result) => {
      if(err){
            console.log('Error-->' + err);
      }else{
            console.log('Status-->' + result.CommonMsg.Status);
      }
});
```

15 Server-Side Encryption

15.1 Server-Side Encryption Overview

NOTICE

If you have any questions during 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**.

15.2 Encryption Description

NOTICE

If you have any questions during 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:

Method in OBS Node.js SDK	Description	Supported Encryption Type
ObsClient.putObject	Sets the encryption algorithm and key during object upload to enable server-side encryption.	SSE-KMS SSE-C

Method in OBS Node.js SDK	Description	Supported Encryption Type
ObsClient.appendObject	Sets the encryption algorithm and key during appendable upload to enable server-side encryption.	SSE-KMS SSE-C
ObsClient.getObject	Sets the decryption algorithm and key during object upload to decrypt the object.	SSE-C
ObsClient.copyObject	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	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

OBS Node.js SDK supports the following two types of encryption/decryption mode:

Encryption / Decryption Type	Request Parameter	Description
SSE-KMS	SseKms	Indicates that SSE-KMS mode is used. Currently, only **kms** is supported.

Encryption / Decryption Type	Request Parameter	Description
	SseKmsKey	Indicates the master key used in SSE-KMS mode. The value can be null.
SSE-C	SseC	Indicates that SSE-C mode is used. Currently, only **AES256** is supported.
	SseCKey	Indicates the key in SSE-C mode. It is calculated using the AES256 algorithm. This parameter can be used to encrypt an object to be uploaded and decrypt an object to be downloaded.
	CopySourceSseC	Indicates the source object decrypted in SSE-C mode. The value can only be AES256. This parameter is applicable to **ObsClient.copyObject** and **ObsClient.copyPart**.
	CopySourceSseCKey	Indicates the key used by a source object for decryption in SSE-C mode. It is calculated using the AES256 algorithm. This parameter is applicable to **ObsClient.copyObject** and **ObsClient.copyPart**.

15.3 Example of Encryption

NOTICE

If you have any questions during 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:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
const obsClient = new ObsClient({
   //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using hard 
coding may result in leakage.
   //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
    access_key_id: process.env.ACCESS_KEY_ID,
    secret_access_key: process.env.SECRET_ACCESS_KEY,
    //EU-Dublin is used here as an example. Replace it with the one in your actual situation.
    server: 'https://obs.eu-west-101.myhuaweicloud.com'

});

obsClient.putObject({
       Bucket: 'bucketname',
       Key: 'objectname',
       SourceFile: 'localfile',
        // Set the SSE-C encryption algorithm.
       SseC: 'AES256',
       SseCKey: 'your sse-c key generated by AES-256 algorithm'
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});

obsClient.putObject({
       Bucket: 'bucketname',
       Key: 'objectname2',
       SourceFile: 'localfile2',
       // Set the SSE-KMS encryption algorithm.
       SseKms: 'kms'
}, (err, result) => {
       if(err){
              console.log('Error-->' + err);
       }else{
              console.log('Status-->' + result.CommonMsg.Status);
       }
});
```

Decrypting a Downloaded Object

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
const obsClient = new ObsClient({
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using hard
coding may result in leakage.
    //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
    access_key_id: process.env.ACCESS_KEY_ID,
    secret_access_key: process.env.SECRET_ACCESS_KEY,
    //EU-Dublin is used here as an example. Replace it with the one in your actual situation.
    server: 'https://obs.eu-west-101.myhuaweicloud.com'

});

obsClient.getObject({
       Bucket: 'bucketname',
       Key: 'objectname',
       // Set the SSE-C decryption algorithm.
       SseC: 'AES256',
        // The key used here must be the one used for uploading the object.
       SseCKey: 'your sse-c key generated by AES-256 algorithm'
}, (err, result) => {
       if(err){
```

```
                console.log('Error-->' + err);
        }else{
                console.log('Status-->' + result.CommonMsg.Status);
                if(result.CommonMsg.Status < 300 && result.InterfaceResult){
                        console.log('Content-->' + result.InterfaceResult.Content.toString());
                }
        }
});
```

16 Troubleshooting

16.1 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 encounters a problem that prevents the operation from completing. The account may be expired or frozen.	403 Forbidden
AllAccessDisabled	You have no permission to perform the operation.	403 Forbidden
AmbiguousGrantByEmailAddress	Multiple accounts share one email address.	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 parameters.	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. Select another name and retry.	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
CustomDomainAreadyExist	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 has exceeded the upper 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 user has no permission to perform some operations due to being in arrears or insufficient funds.	403 Forbidden

Error Code	Description	HTTP Status Code
IncompleteBody	Incomplete request body.	400 Bad Request
IncorrectNumberOfFilesInPost Request	Each POST request must contain one file to be uploaded.	400 Bad Request
InlineDataTooLarge	The size of inline data has exceeded the upper limit.	400 Bad Request
InsufficientStorageSpace	Insufficient storage space.	403 Forbidden
InternalError	An internal error occurs. Retry 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	Invalid storage class.	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 failed the check.	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	This error code is returned after you send an XML file in incorrect format, stating "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	No region contained in the request and no default region defined in the system.	400 Bad Request
MissingRequestBodyError	This error code is returned after you send an empty XML file, stating "Request body is empty."	400 Bad Request
MissingRequiredHeader	Required headers missing in the request.	400 Bad Request
MissingSecurityHeader	A required header is not provided.	400 Bad Request

Error Code	**Description**	**HTTP Status Code**
NoSuchBucket	The specified 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 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 for OBS. OBS is available only after you sign up.	403 Forbidden
OperationAborted	A conflicting operation is being performed on this resource. Retry 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

Error Code	Description	HTTP Status Code
PreconditionFailed	At least one of the specified preconditions is not met.	412 Precondition Failed
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	The socket connection to the server has no read or write operations within the timeout period.	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 does not match the signature calculated by OBS. Check your AK/SK and signature calculation method.	403 Forbidden
SlowDown	Too frequent requests. 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

Error Code	Description	HTTP Status Code
TooManyBuckets	You have attempted to create more buckets than allowed.	400 Bad Request
TooManyObjectCopied	An object has been copied for too many times, exceeding 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 content.	400 Bad Request
UnresolvableGrantByEmailAd-dress	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 server-side encryption with KMS-managed keys (SSE-KMS) mode.	400 Bad Request
KMS.NotFoundException	The master key does not exist in SSE-KMS mode.	400 Bad Request
RestoreAlreadyInProgress	The objects are 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 bucket tags.	400 Bad Request
NoSuchTagSet	The specified bucket is not configured with a tag.	404 Not Found

16.2 SDK Common Result Object

After you call an API in an instance of the **ObsClient** class, a common result object will be returned if no exception is thrown. The following table lists the fields of the object:

Field		Type	Description
CommonMsg		Object	Common information generated after the API is called, including HTTP status code and error code.
	Status	Number	HTTP status code. If the value is smaller than **300**, the operation succeeds. Otherwise, the operation fails.
	Code	String	Error code returned by the OBS server. If **Status** is smaller than **300**, the value is null.
	Message	String	Error description returned by the OBS server. If **Status** is smaller than **300**, the value is null.
	HostId	String	Requested Server ID. If **Status** is smaller than **300**, the value is null.
	RequestId	String	Request ID returned by the OBS server
	Id2	String	Request ID2 returned by the OBS server
	Indicator	String	Detailed error code returned by the OBS server. If **Status** is smaller than **300**, the value is null.
InterfaceResult		Object	Result data generated after the operation is successful. If **Status** is greater than **300**, the value is null.
	RequestId	String	Request ID returned by the OBS server.
	Id2	String	Request ID2 returned by the OBS server
	Other fields	For details, see the *OBS Node.js SDK API Reference*.	

Sample code:

```
// Import the OBS library.
// Use npm to install the client.
var ObsClient = require('esdk-obs-nodejs');
// Use the source code to install the client.
// var ObsClient = require('./lib/obs');

// Create an instance of ObsClient.
const obsClient = new ObsClient({
    //Obtain an AK/SK pair using environment variables or import the AK/SK pair in other ways. Using hard
coding may result in leakage.
```

```
    //Obtain an AK/SK pair on the management console. For details, see https://
support.huaweicloud.com/eu/usermanual-ca/ca_01_0003.html.
    access_key_id: process.env.ACCESS_KEY_ID,
    secret_access_key: process.env.SECRET_ACCESS_KEY,
    //EU-Dublin is used here as an example. Replace it with the one in your actual situation.
    server: 'https://obs.eu-west-101.myhuaweicloud.com'

});

// Call APIs to perform operations, such as downloading an object.
obsClient.getObject({
       Bucket : 'bucketname',
       Key : 'objectname',
}, (err, result) => {
       if(!err){
              if(result.CommonMsg.Status < 300){
                    // Obtain RequestId.
                     console.log('RequestId-->' + result.InterfaceResult.RequestId);
                     // Obtain other parameters.
                     console.log('Content-->' + result.InterfaceResult.Content.toString());
              }else{
                     // Obtain Code and Message.
                     console.log('Code-->' + result.CommonMsg.Code);
                     console.log('Message-->' + result.CommonMsg.Message);
              }
       }
});
```

16.3 Log Analysis

Log Configuration

OBS Node.js SDK provides the logging function based on Log4js. You can call **ObsClient.initLog** to enable and configure logging. Sample code is as follows:

```
obsClient.initLog({
       name: 'test', // Log name
file_full_path:'./logs/OBS-SDK.log', //Set the path to the log file.
max_log_size:20480, //Set the size of the log file, in bytes.
backups:10, //Set the maximum number of log files that can be stored.
level:'warn', //Set the log level.
log_to_console:true //Set whether to print the log to Console.
});
```

NOTE

- The logging function is disabled by default. You need to enable it if needed.
- Use the **file_full_path** parameter to specify the path to the log file. The path can be set to an absolute path or a relative path.

Log Format

The SDK log format is: Log time|log level|invoked interface|log content. The following are examples:

```
2017/10/12 10:21:05 666|INFO |ListBuckets|enter ListBuckets...
2017/10/12 10:21:05 672|INFO |ListBuckets|prepare request parameters ok,then Send request to service start
2017/10/12 10:21:05 715|INFO |ListBuckets|2017-10-12 10:21:05|http cost 34 ms|0|
2017/10/12 10:21:05 716|INFO |ListBuckets|get response start, statusCode:200
```

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.

The following describes each log level in detail.

- **debug**: Debugging level. If this level is set, all log information will be printed.
- **info**: Information level. If this level is set, information about logs of the **warn** level and time consumed for each HTTP/HTTPS request will be printed.
- **warn**: Warning level. If this level is set, information about logs of the error level and information about partial critical events will be printed.
- **error**: Error level. If this level is set, only error information will be printed.

16.4 Lack of Modules

If an error indicating the lack of modules is displayed (such as **Cannot find module 'xml2js**) when you use OBS Node.js SDK for secondary development, check whether the dependent libraries have been properly installed. For details, see **Installing the SDK**.

16.5 Connection Timeout

If there is an error **connect ETIMEDOUT** reported when calling an API, the probable reason is that the OBS endpoint you specified is wrong or the network is disconnected. You need to check the OBS endpoint and network conditions.

16.6 Unmatched Signatures

If the HTTP status code obtained from **CommonMsg.Status** is **403** and the OBS server error code returned by **CommonMsg.Code** is **SignatureDoesNotMatch**, check whether the AK/SK is correct.

A API Reference

For details about all parameters and definitions of APIs in the OBS Node.js SDK, see the **OBS Node.js SDK API Reference**.

B Change History

Release Date	What's New
2023-03-14	This is the first official release.