Object Storage Service

iOS SDK Developer Guide

Issue 09

Date 2020-04-21

HUAWEI

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

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

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

Contents

1 SDK Download Links

- Latest version of OBS iOS SDK: **OBS iOS SDK**
- API documentation: **OBS_iOS_SDK_API Reference**

2 Compatibility

- Recommended iOS versions: iOS 8.0 to 12.3.1
- Recommended development tool versions: Xcode 8.0 to 10.0
- API functions: incompatible with earlier versions (since V2.1.4)

3 Quick Start

3.1 Before You Start

- Ensure that you are familiar with OBS basic concepts, such as **bucket**, **object**, and **AK and SK**.
- You can see **General Examples of OBSClient** to understand how to call OBS iOS SDK APIs in a general manner.

3.2 Setting Up an OBS Environment

Step 1 Register a cloud service account.

Ensure that you have a cloud service account before using OBS.

1. Open a browser.
2. Visit **www.huaweicloud.com/en-us/**.
3. In the upper right corner of the page, click **Register**.
4. Enter the registration information and click **Register**.

Step 2 Subscribe to OBS.

Ensure that your account balance is sufficient before using OBS.

1. Log in to OBS Console.
2. Click **Billing** in the upper right corner of the page to go to the billing center.
3. Click **Top Up**. The dialog box for top-up is displayed.
4. Top up the account as prompted.
5. After the top-up is successful, close the window and go back to the homepage of the management console.
6. Click **Object Storage Service** to subscribe to OBS and enter OBS Console.

Step 3 Create access keys.

OBS uses AKs and SKs in user accounts for signature verification to ensure that only authorized accounts can access specified OBS resources. Detailed explanations about AK and SK are as follows:

- An access key ID (AK) defines a user who accesses the OBS system. An AK belongs to only one user, but one user can have multiple AKs. The OBS system recognizes the users who access the system by their access key IDs.
- A secret access key (SK) is the key used by users to access OBS. It is the authentication information generated based on the AK and the request header. An SK matches an AK, and they group into a pair.

Access keys are classified into permanent access keys (AK/SK) and temporary access keys (AK/SK and security token). A user can create a maximum of two valid permanent access keys. Temporary access keys can be used to access OBS only within the specified validity period. After the temporary access keys expire, they need to be obtained again. For security purposes, you are advised to use temporary access keys to access OBS, or periodically update your access keys if you use permanent access keys. The following describes how to obtain access keys of these two types.

- Permanent access keys:
  a. Log in to OBS Console.
  b. In the upper right corner of the page, hover the cursor over the username and choose **My Credentials**.
  c. On the **My Credentials** page, select **Access Keys** in the navigation pane on the left.
  d. On the **Access Keys** page, click **Create Access Key**.
  e. In the **Create Access Key** dialog box that is displayed, enter the password and verification code.

     NOTE

     - If you have not bound an email address or mobile number, enter only the password.
     - If you have bound an email address and a mobile number, you can select the verification by either email or mobile phone.

  f. Click **OK**.
  g. In the **Download Access Key** dialog box that is displayed, click **OK** to save the access keys to your browser's default download path.
  h. Open the downloaded **credentials.csv** file to obtain the access keys (AK and SK).

     NOTE

     - A user can create a maximum of two valid access keys.
     - Keep the access key properly. If you click **Cancel** in the dialog box, the access keys will not be downloaded, and cannot be obtained later. You can re-create access keys if you need to use them.
- Temporary access keys:

  The temporary AK/SK and security token are temporary access tokens issued by the system to users. The validity period ranges from 15 minutes to 24 hours which can be set using APIs. After the validity period expires, users need to obtain the access keys again. The temporary AK/SK and security token shall observe the principle of least privilege. When the temporary AK/SK are used

for authentication, the temporary AK/SK and security token must be used at the same time.

For details about how to obtain temporary access keys, see **Obtaining a Temporary Access Key and Security Token**.

For details about how to use temporary access keys, see **Creating an Instance of OBSClient**.

----End

3.3 Preparing a Development Environment

Download Xcode of the latest version from the **Xcode's official website**.

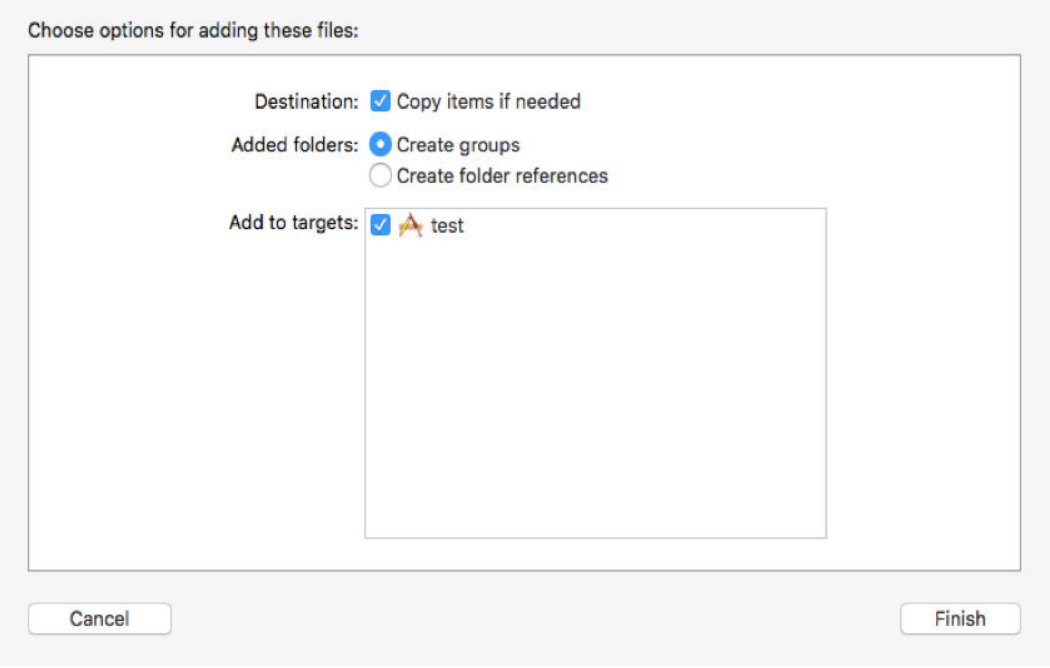
3.4 Installing the SDK

iOS Project Configuration

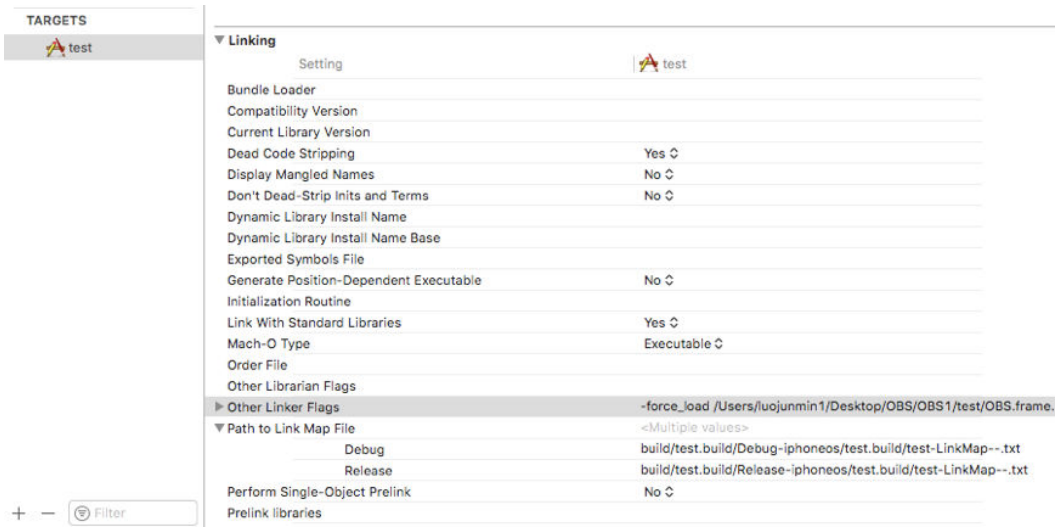
Step 1 Download the OBS iOS SDK software package by referring to **SDK Download Links**.

Step 2 On Xcode, create a project.

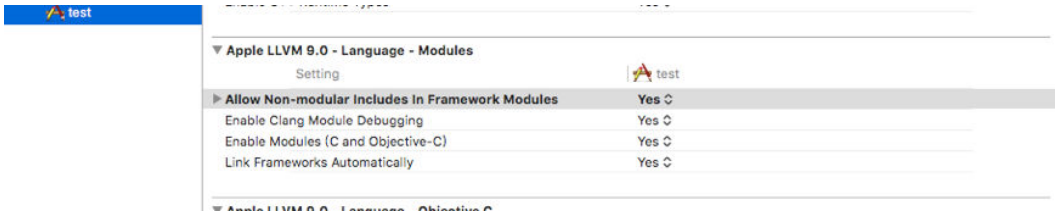
Step 3 Add **OBS.framework** to the project and select **Copy items if needed**.



Step 4 Choose **TARGETS** > **Build Settings** > **Linking** > **Other Linker Flags** and add flag **-force_load $(SRCROOT)***XXX*/**OBS.framework/OBS**. *XXX* indicates the name of the folder saving **OBS.framework** under the project folder.



Step 5 Choose **TARGETS** > **Build Settings** > **Apple LLVM9.0 – Language –Modules** and set **Allow Non-modular includes in Framework Modules** to **Yes**.



Step 6 Import the OBS header file when needed.

```
#import <OBS/OBS.h>
```

Step 7 Run **command+B** to check whether **OBS.framework** passes the compilation.

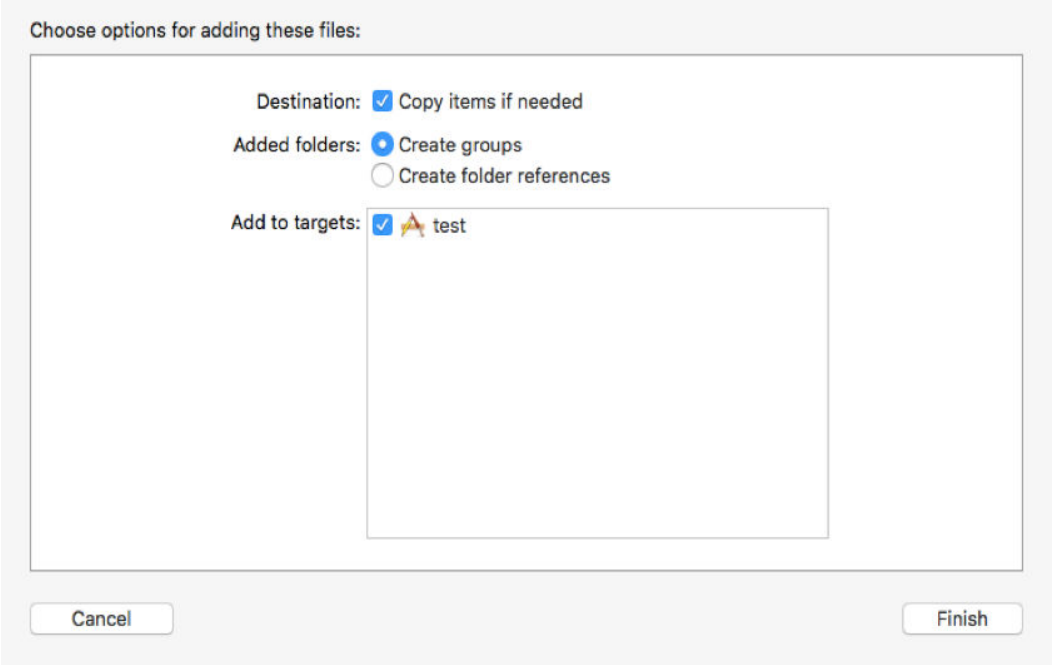
----End

Mac OSX Project Configuration

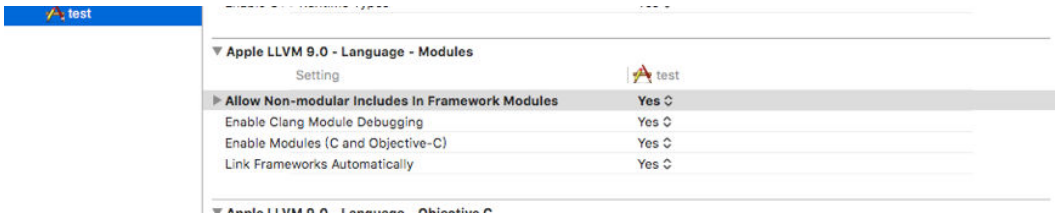
Step 1 Download the OBS iOS SDK software package by referring to **SDK Download Links**.

Step 2 On Xcode, create a project.

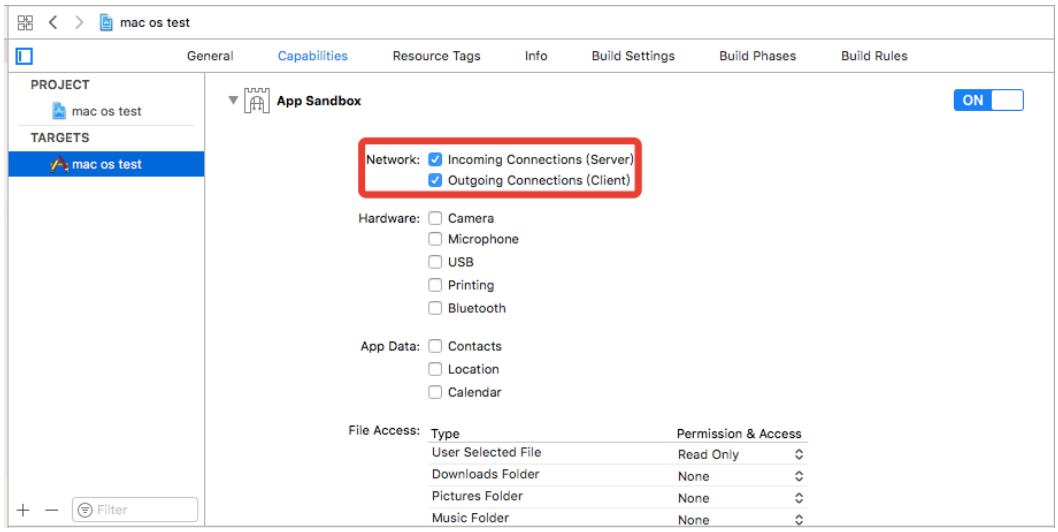
Step 3 Add **OBS.framework** to the project and select **Copy items if needed**.



Step 4 Choose **TARGETS** > **Build Settings** > **Apple LLVM9.0 – Language –Modules** and set **Allow Non-modular includes in Framework Modules** to **Yes**.



Step 5 Set network connections.



Step 6 Import the OBS header file when needed.

```
#import <OBS/OBS.h>
```

Step 7 Run **command+B** to check whether **OBS.framework** passes the compilation.

----End

3.5 Obtaining Endpoints

- You can click **here** to view the endpoints and regions enabled for OBS.

NOTICE

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

3.6 Initializing an Instance of OBSClient

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

```
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:@"*** Provide your Access Key ***" secretKey:@"*** Provide your Secret Key ***"];

OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:@"https://your-
endpoint" credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
OBSClient *client  = [[OBSClient alloc] initWithConfiguration:conf];
}
```

NOTE

For more information, see chapter "Initialization."

3.7 Creating a Bucket

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. Sample code is as follows:

```
OBSCreateBucketRequest *request = [[OBSCreateBucketRequest alloc]
initWithBucketName:@"bucketname"];
[client createBucket:request completionHandler:^(OBSCreateBucketResponse *response, NSError *error) {
    NSLog(response.location);
}];
```

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, starts with a digit or letter, and supports only lowercase letters, digits, hyphens (-), and periods (.)
  - 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**.
- For more information, see **Bucket Management**.

NOTICE

- This parameter is not required if the endpoint belongs to the default region (cn-north-1). Otherwise, set this parameter to the region to which the endpoint belongs. Click **here** to query currently valid regions.
- When creating a bucket, you can specify its region. For details, see **Creating a Bucket with Parameters Specified**.

3.8 Uploading an object

Sample code:

```
NSString *filePath = [[NSBundle mainBundle]pathForResource:@"fileName" ofType:@"Type"];
OBSPutObjectWithFileRequest *request = [[OBSPutObjectWithFileRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" uploadFilePath:filePath];
// Query the upload progress.
request.uploadProgressBlock = ^(int64_t bytesSent, int64_t totalBytesSent, int64_t
totalBytesExpectedToSend) {
    NSLog(@"%0.1f%%",(float)floor(totalBytesSent*10000/totalBytesExpectedToSend)/100);
};
// Upload a file.
[ client putObject:request completionHandler:^(OBSPutObjectResponse *response, NSError *error){
    NSLog(@"%@",response.etag);
}];
```

NOTE

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

3.9 Downloading an Object

Sample code:

```
NSString * outfilePath = [NSTemporaryDirectory() stringByAppendingString:@"filename"];
OBSGetObjectToFileRequest *request = [[OBSGetObjectToFileRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" downloadFilePath:outfilePath];
// Query the download progress.
request.downloadProgressBlock = ^(int64_t bytesWritten, int64_t totalBytesWritten, int64_t
totalBytesExpectedToWrite) {
    NSLog(@"%0.1f%%",(float)floor(totalBytesWritten*10000/totalBytesExpectedToWrite)/100);
};
```



```
[client getObject:request completionHandler:^(OBSGetObjectResponse *response, NSError *error){
    NSLog(@"%@",response.etag);
}];
```

NOTE

- When you call **getObject**, an instance of **OBSObject** will be returned. This instance contains the content and properties of the object.
- For more information, see **Object Download**.

3.10 Listing Objects

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

```
OBSListObjectsRequest *request = [[OBSListObjectsRequest alloc] initWithBucketName:@"bucketname"];
request.maxKeys = [NSNumber numberWithInt:10];
request.origin = @"www.example1.com";
[client listObjects:request completionHandler:^(OBSListObjectsResponse *response, NSError *error) {
    NSLog(@"%d",response.contentsList.count);
}];
```

NOTE

- After you call **listObjects**, an instance of **OBSListObjectsResponse** will be returned. This instance contains the response result of the **listObject** request.
- In the previous sample code, 1000 objects will be listed, by default.
- For more information, see **Listing Objects**.

3.11 Deleting an Object

Sample code:

```
OBSDeleteObjectRequest *request = [[OBSDeleteObjectRequest alloc] initWithBucketName:@"bucketname"
objectKey:@"objectname"];
[client deleteObject:request completionHandler:^(OBSDeleteObjectResponse *response, NSError *error) {
    NSLog(@"%@",response);
}];
```

3.12 General Examples of OBSClient

If no error is generated when you call an API in an instance of the OBSClient class, the operation is successful. Otherwise, the operation fails. Sample code is as follows:

```
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

// Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
OBSClient *client = [[OBSClient alloc] initWithConfiguration:conf];
```

```
// Create a request for listing objects.
OBSListBucketsRequest *request = [OBSListBucketsRequest new];

// List objects.
OBSBFTask *task = [client listBuckets:request completionHandler:^(OBSListBucketsResponse *response,
NSError *error) {
    if(error){
        // Rectify the fault.

    }else{
        // Obtain the result.
        for(OBSBucket *bucket in response.bucketsList){
            NSLog(@"bucketname=%@",bucket.name);
        }
    }
}];
[task waitUntilFinished];
```

NOTICE

- As an instance of OBSClient is declared as a local variable, when an OBSClient API is called, the **waitUntilFinished** operation must be performed on the request task to ensure that the instance of OBSClient is always valid during the request task execution. Otherwise, the network request may fail and the program may break down.

4 Initialization

4.1 Configuring the AK and SK

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 start initialization.

4.2 Creating an Instance of OBSClient

OBSClient functions as the iOS client for accessing OBS. It offers users 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 iOS SDK to send a request to OBS, you need to initialize an instance of OBSClient and modify the default configurations in **OBSServiceConfiguration** based on actual needs.

Sample code for creating an instance of OBSClient using permanent access keys (AK/SK):

```
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

// Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
OBSClient *client  = [[OBSClient alloc] initWithConfiguration:conf];
```

Sample code for creating an instance of OBSClient using temporary access keys (AK/SK and security token):

```
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";
```



```
// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];
// securityToken
credentialProvider.securityToken = @"*** Provide your Security Token ***";

// Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
OBSClient *client  = [[OBSClient alloc] initWithConfiguration:conf];
```

NOTE

Currently, when multiple resumable upload tasks need to be executed concurrently, an independent instance of OBSClient needs to be initialized for each upload task to process requests.

Sample code for creating an OBSClient instance using a user-defined domain name:

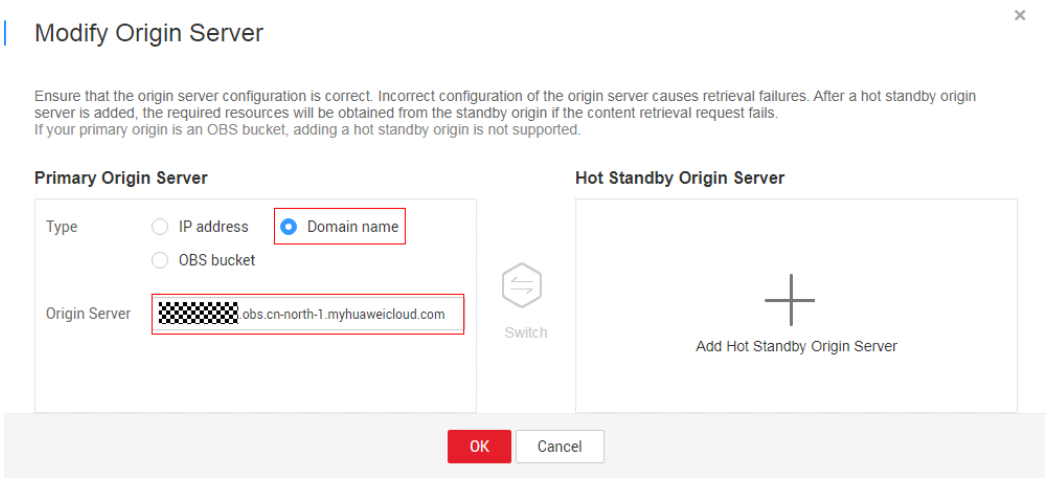
```
NSString *endPoint = @"your-custom-domain";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

// Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];
// Set to use self-defined domain name to access OBS.
conf.defaultDomainMode = OBSDomainModeCustom;

// Initialize an instance of OBSClient.
OBSClient *client  = [[OBSClient alloc] initWithConfiguration:conf];
```

You can set OBSServiceConfiguration.defaultDomainMode to OBSDomainModeCustom to access OBS using a user-defined domain name. In this case, set **endPoint** to the user-defined domain name. To access an OBS bucket using a user-defined domain name, you need to bind the user-defined domain name to the access domain name of the bucket. For details, see **User-Defined Domain Name Binding** and **Configuring User-Defined Domain Name Binding**.

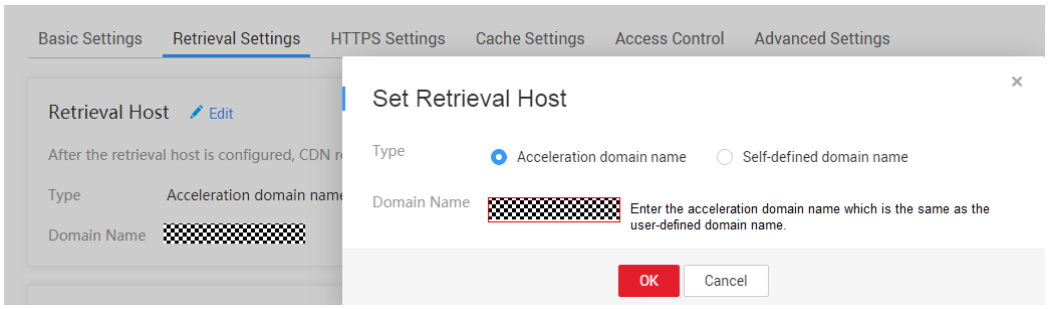
If CDN is configured for a user-defined domain name, that is, the user-defined domain name is a CDN acceleration domain name, you need to configure CDN to ensure that the user-defined domain name can be used to access OBS. The following uses HUAWEI CLOUD as an example to describe how to configure CDN.

Step 1 Log in to the HUAWEI CLOUD CDN console and select **Domain Name Management** from the navigation pane on the left. You can view all configured CDN domain names.

Step 2 Configure the origin server. Click the user-defined domain name to be used to go to the domain name configuration page. Modify the origin server. Set **Type** of **Primary Origin Server** to **Domain name**, and set **Origin Server** to the domain name of the OBS bucket to be accessed.



Step 3 Set the retrieval host. The retrieval host must be specified as the acceleration domain name, that is, the user-defined domain name to access OBS. Otherwise, the retrieval authentication may fail.



----End

4.3 Configuring an Instance of OBSClient

You can use the **OBSServiceConfiguration** configuration class to configure an instance of OBSClient, including the proxy, connection timeout period, maximum number of retry attempts, and other parameters listed in the following table.

Table 4-1 Parameters for configuring a network request

Parameter	Description	Recommended Value
OBSServiceConfiguration.credentialProvider	User credential. For details, see **Table 4-2**.	N/A
OBSServiceConfiguration.proxyConfig	Proxy configuration. The default value is null. For details, see **Table 4-3**.	N/A
OBSServiceConfiguration.trustUnsafeCert	Whether to trust insecure certificates. The default value is **NO**.	Default value

Parameter	Description	Recommended Value
OBSServiceConfiguration.maxConcurrentCommandRequestCount	Maximum number of concurrent command requests. The default value is **3**.	Default value
OBSServiceConfiguration.maxConcurrentUploadRequestCount	Maximum number of concurrent upload requests. The default value is **3**.	Default value
OBSServiceConfiguration.maxConcurrentDownloadRequestCount	Maximum number of concurrent download requests. The default value is **3**.	Default value
OBSServiceConfiguration.defaultDomainMode	Domain name access mode. You can set this parameter to **OBSDomainModeCustom** to use a user-defined domain name. By default, the non-user-defined domain name access mode is used.	Default value
OBSServiceConfiguration.commandSessionConfiguration.HTTPMaximumConnectionsPerHost	Maximum number of command request connections that can be opened. The default value in iOS is **4**.	N/A
OBSServiceConfiguration.uploadSessionConfiguration.HTTPMaximumConnectionsPerHost	Maximum number of upload request connections that can be opened. The default value in iOS is **4**.	N/A
OBSServiceConfiguration.downloadSessionConfiguration.HTTPMaximumConnectionsPerHost	Maximum number of download request connections that can be opened. The default value in iOS is **4**.	N/A
OBSServiceConfiguration.backgroundUploadSessionConfiguration.HTTPMaximumConnectionsPerHost	Maximum number of background upload request connections that can be opened. The default value in iOS is **4**.	N/A
OBSServiceConfiguration.backgroundDownloadSessionConfiguration.HTTPMaximumConnectionsPerHost	Maximum number of background download request connections that can be opened. The default value in iOS is **4**.	N/A

NOTE

Set parameters with **N/A** as the recommended value based on your needs. For security purposes, you are advised to use HTTPS for endpoints.

The following table describes parameters involved in **OBSStaticCredentialProvider**:

Table 4-2 Server identification configurations

Parameter	Description	Method
accessKey	User's AK	credentialProvider.Access Key = @"Provide your Access Key"
secretKey	User's SK	credentialProvider.Secret Key = @"Provide your Secret Key"
securityToken	Temporary token	credentialProvid-er.securityToken = token

NOTE

credentialProvider is an instance of **OBSStaticCredentialProvider**.

For details about how to obtain the **securityToken**, see **Setting Up an OBS Environment**.

The following table describes parameters involved in **OBSHTTPProxyConfiguration**:

Table 4-3 Proxy server configurations

Parameter	Description	Method
proxyType	Network access type (Possible values are enumerated).	To allow HTTP only: proxyConfig.proxyType=OBSHTTPRroxyTypeHTTP To allow HTTPS only: proxyConfig.proxyType=OBSHTTPRroxyTypeHTTPS To allow both HTTP and HTTPS: proxyConfig.proxyType=OBSHTTPRroxyTypeHTTPAndHTTPS
proxyHost	Host address of the proxy server.	proxyConfig.proxyHost = @"host"
proxyPort	Port ID of the proxy server.	proxyConfig.proxyPort = @"port"
username	Username used for connecting to the proxy server.	proxyConfig.username = @"username"
password	Password used for connecting to the proxy server.	proxyConfig.password = @"password"

NOTE

proxyConfig is an instance of **OBSHTTPProxyConfiguration**.

5 Bucket Management

5.1 Creating a Bucket

You can call **createBucket** to create a bucket.

Creating a Bucket in Simple Mode

Sample code:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

//Make a request for creating a bucket.
OBSCreateBucketRequest *request = [[OBSCreateBucketRequest alloc]
initWithBucketName:@"bucketname"];
//Create a bucket.
[client createBucket:request completionHandler:^(OBSCreateBucketResponse *response, NSError *error) {
    NSLog(@"%@",response.location);
 }];
```


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, starts with a digit or letter, and supports only lowercase letters, digits, hyphens (-), and periods (.)
  - 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.
- In this example, the bucket is of the default **ACL** (**public-read-write**), in the OBS Standard storage class, and in the default location where the global domain resides.

NOTICE

- This parameter is not required if the endpoint belongs to the default region (cn-north-1). Otherwise, set this parameter to the region to which the endpoint belongs. Click **here** to query currently valid regions.
- When creating a bucket, you can specify its region. For details, see **Creating a Bucket with Parameters Specified**.

Creating a Bucket with Parameters Specified

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

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

//Create a bucket.
OBSCreateBucketRequest *request = [[OBSCreateBucketRequest alloc]
initWithBucketName:@"bucketname"];
// Set the access control policy to public-read-write.
request.bucketACLPolicy = OBSACLPolicyPublicReadWrite;
// Set the storage class to OBS Standard.
request.defaultStorageClass = OBSStorageClassStandard;
// Set the bucket location.
request.configuration = [[OBSBucketConfiguration alloc] initWithLocationConstraint:@"bucketlocation"];

[client createBucket:request completionHandler:^(OBSCreateBucketResponse *response, NSError *error) {
```



```
    NSLog(@"%@",response.location);
}];
```

5.2 Listing Buckets

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

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// List buckets.
OBSListBucketsRequest *request = [OBSListBucketsRequest new];

[client listBuckets:request completionHandler:^(OBSListBucketsResponse *response, NSError *error) {
    for(OBSBucket *bucket in response.bucketsList){
        NSLog(@"bucketname=%@",bucket.name);
    }
}];
```

NOTE

Obtained bucket names are listed in the lexicographical order.

5.3 Deleting a Bucket

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

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

//Delete a bucket.
OBSDeleteBucketRequest *deleteRequest = [[OBSDeleteBucketRequest alloc]
initWithBucketName:@"bucketname"];
[client deleteBucket:deleteRequest completionHandler:^(OBSDeleteBucketResponse *response, NSError
*error) {
    NSLog(@"%@",response);
}];
```

NOTE

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

5.4 Obtaining Bucket Metadata

You can call **getBucketMetaData** to obtain bucket metadata. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

//Obtain the bucket metadata.
OBSGetBucketMetaDataRequest *request = [[OBSGetBucketMetaDataRequest
alloc]initWithBucketName:@"bucketname"];

[client getBucketMetaData:request completionHandler:^(OBSGetBucketMetaDataResponse *response,
NSError *error){
    NSLog(@"%@",response);
}];
```

5.5 Managing Bucket ACLs

A bucket **ACL** can be configured in three modes:

1. Specify a pre-defined access control policy during bucket creation.
2. Call **OBSSetBucketACLWithCannedACLRequest** to specify a pre-defined access control policy.
3. Call **OBSSetBucketACLWithPolicyRequest** to set the ACL directly.

The following table lists the five permissions supported by OBS.

Permission	Description	Enumeration Value in OBS iOS SDK
READ	A grantee with this permission for a bucket can obtain the list of objects in the bucket and the metadata of the bucket. A grantee with this permission for an object can obtain the object content and metadata.	OBSACLRead
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.	OBSACLWrite
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.	OBSACLRead_ACP
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.	OBSACLWrite_ACP
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.	OBSACLFull_Control

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

Policy	Description	Enumeration Value in OBS iOS 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.	OBSACLPolicyPrivate
public-read	Indicates that the owner of a bucket or object has the **FULL_CONTROL** permission for the bucket or object. Other users including anonymous users have the **READ** permission.	OBSACLPolicyPublicRead
public-read-write	Indicates that the owner of a bucket or object has the **FULL_CONTROL** permission for the bucket or object. Other users including anonymous users have the **READ** and **WRITE** permissions.	OBSACLPolicyPublicReadWrite
public-read-delivered	If this permission is set for a bucket, everyone can obtain the object list, multipart uploads, and bucket metadata in the bucket, and obtain the content and metadata of the objects in the bucket. This permission cannot be set for objects.	OBSACLPolicyPublicReadDelivered
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.	OBSACLPolicyPublicReadWriteDelivered

Specifying a Pre-defined Access Control Policy During Bucket Creation

Sample code:


```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

//Create a bucket.
OBSCreateBucketRequest *request = [[OBSCreateBucketRequest alloc]
initWithBucketName:@"bucketname"];
// Set the access control policy to public-read-write.
request.bucketACLPolicy = OBSACLPolicyPublicReadWrite;

[client createBucket:request completionHandler:^(OBSCreateBucketResponse *response, NSError *error) {
    NSLog(@"%@",response.location);
}];
```

Setting a Pre-defined Access Control Policy for the Bucket

Sample code:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

//Set the pre-defined access control policy to  public-read-write.
OBSSetBucketACLWithCannedACLRequest *request = [[OBSSetBucketACLWithCannedACLRequest
alloc]initWithBucketName:@"bucketname" cannedACL:OBSACLPolicyPublicRead];
[client setBucketACL:request completionHandler:^(OBSSetBucketACLResponse *response, NSError *error){
    NSLog(@"%@",response);
}];
```

Directly Setting the Bucket ACL

Sample code:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
```



```
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Initialize an instance of OBSUser.
OBSUser *owner = [[OBSUser alloc] initWithID:@"ownerID"];
// Set a grantee.
OBSACLGranteeUser *grantee = [[OBSACLGranteeUser alloc]initWithID:@"granteeID"];
// Grant the FULL_CONTROL permission to the grantee.
OBSACLGrant *grant = [[OBSACLGrant alloc]initWithGrantee:grantee permission:OBSACLFull_Control];

// Create a policy object.
OBSAccessControlPolicy *policy = [OBSAccessControlPolicy new];
policy.owner = owner;
[policy.accessControlList addObject:grant];

// Directly set the ACL for the bucket.
OBSSetBucketACLWithPolicyRequest *request = [[OBSSetBucketACLWithPolicyRequest
alloc]initWithBucketName:@"bucketname" accessControlPolicy:policy];
[client setBucketACL:request completionHandler:^(OBSSetBucketACLResponse *response, NSError *error){
    NSLog(@"%@",response);
}];
```

NOTE

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

Obtaining a Bucket ACL

You can call **getBucketACL** to obtain the bucket ACL. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Obtain the bucket ACL.
OBSGetBucketACLRequest *request = [[OBSGetBucketACLRequest alloc]
initWithBucketName:@"bucketname"];
[client getBucketACL:request completionHandler:^(OBSGetBucketACLResponse *response, NSError *error){
    NSLog(@"%@",response);
}];
```

5.6 Obtaining the Location of a Bucket

You can call **getBucketLocation** to obtain the location of a bucket. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";
```



```
// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Obtain a bucket location.
OBSGetBucketLocationRequest *request = [[OBSGetBucketLocationRequest
alloc]initWithBucketName:@"bucketname"];

[client getBucketLocation:request completionHandler:^(OBSGetBucketLocationResponse *response, NSError
*error){
    NSLog(response.configuration.locationConstraint);
}];
```

NOTE

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

5.7 Managing Bucket Policies

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 **setBucketPolicy** to set a bucket policy. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Set a bucket policy.
OBSSetBucketPolicyWithStringRequest *request = [[OBSGetBucketPolicyRequest alloc]
initWithBucketName:@"bucketname" policyString:@"policystring"];
[client setBucketPolicy:request completionHandler:^(OBSSetBucketPolicyResponse *response, NSError *error)
{
    NSLog(@"%@",response);
 }];
```

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 **getBucketPolicy** to obtain a bucket policy. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Obtain the bucket ACL.
OBSGetBucketPolicyRequest *request = [[OBSGetBucketPolicyRequest alloc]
initWithBucketName:g_bucketName];
[client getBucketPolicy:request completionHandler:^(OBSGetBucketPolicyResponse *response, NSError
*error){
    NSLog(@"%@",response);
 }];
```

Deleting a Bucket Policy

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

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Delete a bucket policy.
OBSDeleteBucketPolicyRequest *request = [[OBSDeleteBucketPolicyRequest alloc]
initWithBucketName:g_bucketName];
[client deleteBucketPolicy:request completionHandler:^(OBSDeleteBucketPolicyResponse *response, NSError
*error) {
     NSLog(@"%@",response);
}];
```

5.8 Obtaining Storage Information About a Bucket

The storage information about a bucket includes the number of objects in and the used capacity of the bucket. You can call **getBucketStorageInfo** to obtain the bucket storage information. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
```



```
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Obtain the storage information about a bucket.
OBSGetBucketStorageInfoRequest *request = [[OBSGetBucketStorageInfoRequest alloc]
initWithBucketName:@"bucketname"];

[client getBucketStorageInfo:request completionHandler:^(OBSGetBucketStorageInfoResponse *response,
NSError *error) {
    NSLog(@"%@",response.storageinfo);
 }];
```

5.9 Setting or Obtaining a Bucket Quota

Setting a Bucket Quota

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

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Set the bucket quota.
OBSQuota *quota = [[OBSQuota alloc] initWithQuotaNumber:[NSNumber numberWithLongLong:
1024*1024*1024]];

OBSSetBucketQuotaRequest *request = [[OBSSetBucketQuotaRequest alloc]
initWithBucketName:@"bucketname" quota:quota];
[client setBucketQuota:request completionHandler:^(OBSSetBucketQuotaResponse *response, NSError
*error) {
    NSLog(@"%@",response.statusCode);
}];
```

NOTE

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 **getBucketQuota** to obtain a bucket quota. Sample code is as follows:


```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Obtain the bucket quota.
OBSGetBucketQuotaRequest *request = [[OBSGetBucketQuotaRequest
alloc]initWithBucketName:@"bucketname"];

[client getBucketQuota:request completionHandler:^(OBSGetBucketQuotaResponse *response, NSError
*error){
    NSLog(@"%@",response.quota);
}];
```

5.10 Setting or Obtaining the Storage Class of a Bucket

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.

Storage Class	Description	Enumeration Value in OBS iOS 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.	OBSStorageClassStandard
OBS Infrequent Access	Is applicable to storing semi-frequently accessed (less than 12 times a year) data requiring quick response.	OBSStorageClassStandardIA
OBS Archive	Is applicable to archiving rarely-accessed (once a year) data.	OBSStorageClassGlacier

For more information, see **Bucket Storage Classes**.

Setting the Storage Class for a Bucket

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

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Set the storage class to OBS Standard.
OBSStoragePolicy* policy = [[OBSStoragePolicy alloc]initWithStorageClass:OBSStorageClassStandard];

// Set the storage class for a bucket.
OBSSetBucketStoragePolicyRequest *request = [[OBSSetBucketStoragePolicyRequest
alloc]initWithBucketName:@"bucketname" storagePolicy:policy];

[client setBucketStoragePolicy:request completionHandler:^(OBSSetBucketStoragePolicyResponse *response,
NSError *error){
    NSLog(@"%@",response);
}];
```

Obtaining the Storage Class of a Bucket

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

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Obtain the storage class of a bucket.
OBSGetBucketStoragePolicyRequest *request = [[OBSGetBucketStoragePolicyRequest
alloc]initWithBucketName:@"bucketname"];

[client getBucketStoragePolicy:request completionHandler:^(OBSGetBucketStoragePolicyResponse
*response, NSError *error){
    NSLog(@"%@",response.storagePolicy);
}];
```


6 Object Upload

6.1 Object Upload Overview

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

- **Performing a Streaming Upload**
- **Performing a File-Based Upload**
- **Performing a Multipart Upload**
- **Appendable Upload**
- **Performing a Resumable Upload**

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

If the uploaded object can be read by anonymous users. After the upload succeeds, anonymous users can access the object data through the object URL. 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.

6.2 Performing a Streaming Upload

Streaming upload uses **OBSPutObjectWithDataRequest** to obtain the data source. You can call **putObject** to upload the data streams to OBS. Sample code is as follows:

Uploading a Character String

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";
```



```
// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Upload a character string in streaming mode.
OBSPutObjectWithDataRequest *request = [[OBSPutObjectWithDataRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" uploadData:[@"hello"
dataUsingEncoding:NSUTF8StringEncoding]];

// Query the upload process.
request.uploadProgressBlock = ^(int64_t bytesSent, int64_t totalBytesSent, int64_t
totalBytesExpectedToSend) {
    NSLog(@"%0.1f%%",(float)floor(totalBytesSent*10000/totalBytesExpectedToSend)/100);
};

[client putObject:request completionHandler:^(OBSPutObjectResponse *response, NSError *error){
    NSLog(@"%@",response);
}];
```

Uploading a Network Stream

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Upload a network stream in streaming mode.
OBSPutObjectWithDataRequest *request = [[OBSPutObjectWithDataRequest alloc]
initWithBucketName:@"bucketname" objectKey:@"objectname" uploadDataURL:[NSURL
URLWithString:@"DateURL"]];

// Query the upload process.
request.uploadProgressBlock = ^(int64_t bytesSent, int64_t totalBytesSent, int64_t
totalBytesExpectedToSend) {
    NSLog(@"%0.1f%%",(float)floor(totalBytesSent*10000/totalBytesExpectedToSend)/100);
};

[client putObject:request completionHandler:^(OBSPutObjectResponse *response, NSError *error){
    NSLog(@"%@",response);
}];
```

Uploading a File Stream

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";


// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
```



```
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

NSString *filePath = [[NSBundle mainBundle]pathForResource:@"fileName" ofType:@"Type"];
// Upload a file stream.
NSData *uploadData = [NSData dataWithContentsOfFile:filePath];
OBSPutObjectWithDataRequest *request = [[OBSPutObjectWithDataRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"test/image1" uploadData:uploadData];

// Query the upload process.
request.uploadProgressBlock = ^(int64_t bytesSent, int64_t totalBytesSent, int64_t
totalBytesExpectedToSend) {
   NSLog(@"%0.1f%%",(float)floor(totalBytesSent*10000/totalBytesExpectedToSend)/100);
};

[client putObject:request completionHandler:^(OBSPutObjectResponse *response, NSError *error){
    NSLog(@"%@",response);
}];
```

NOTE

- To upload local files, **file-based upload** is recommended.
- To upload large files, **multipart upload** is recommended.
- The content to be uploaded cannot exceed 5 GB.

6.3 Performing a File-Based Upload

File-based upload uses local files as the data source of objects. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];
// Specify the name and type of the file to be uploaded.
NSString *filePath = [[NSBundle mainBundle]pathForResource:@"fileName" ofType:@"Type"];
// Upload a file.
OBSPutObjectWithFileRequest *request = [[OBSPutObjectWithFileRequest
alloc]initWithBucketName:@"bucket-ios-test02" objectKey:@"imageWithFile" uploadFilePath:filePath];
// Enable background upload. When an application is switched to the background, the ongoing upload
continues.
request.background = YES;

// Query the upload progress.
request.uploadProgressBlock = ^(int64_t bytesSent, int64_t totalBytesSent, int64_t
totalBytesExpectedToSend) {
    NSLog(@"%0.1f%%",(float)floor(totalBytesSent*10000/totalBytesExpectedToSend)/100);
};
```



```
[client putObject:request completionHandler:^(OBSPutObjectResponse *response, NSError *error){
   // Determine the error status.
   if(error){
      // Print error information.
      NSLog(@"Failed to upload the file.");
      NSLog(@"%@",error);
   }
// If the file is uploaded successfully, the response status code 200 is returned and printed.
   if([response.statusCode isEqualToString:@"200"]){
      NSLog(@"The file is successfully uploaded.");
      NSLog(@"%@",response);
      NSLog(@"%@",response.etag);
   }
}];
```

NOTE

The content to be uploaded cannot exceed 5 GB.

When **background** is set to **YES**, background upload is enabled.

6.4 Creating a Folder

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.

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];
//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];
// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// // Create an empty string.
OBSPutObjectWithDataRequest *request = [[OBSPutObjectWithDataRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"file/" uploadData:[@""
dataUsingEncoding:NSUTF8StringEncoding]];

// Query the upload process.
request.uploadProgressBlock = ^(int64_t bytesSent, int64_t totalBytesSent, int64_t
totalBytesExpectedToSend) {
   NSLog(@"%0.1f%%",(float)floor(totalBytesSent*10000/totalBytesExpectedToSend)/100);
};
// Create a folder named file.
[client putObject:request completionHandler:^(OBSPutObjectResponse *response, NSError *error){
   NSLog(@"%@",response);
}];
```

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.

6.5 Setting Object Properties

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

The following table describes object properties.

Property Name	Description	Default Value
contentMD5	Indicates the base64-encoded digest of the object data. It is provided to the OBS server to verify data integrity.	N/A
storageClass	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.	OBS Standard
metaDataDict	Indicates the user-defined description of properties of the object uploaded to the bucket. It is used to facilitate the customized management on the object.	N/A
contentType	Indicates the MIME type of the object specified during upload, 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 stream
customContentType	Indicates the MIME type of the object specified during upload, which defines the type and network code of the object. Different from the **contentType**, this parameter allows the input of any characters to specify the MIME type of the object to be uploaded.	N/A

Setting the MD5 Value for an Object

You can set **contentMD5** to specify the MD5 value for an object. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];
```



```
//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

NSString *filePath = [[NSBundle mainBundle]pathForResource:@"fileName" ofType:@"Type"];
//Upload a file.
OBSPutObjectWithFileRequest *request = [[OBSPutObjectWithFileRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" uploadFilePath:filePath];

// Set the MD5 value for an object.
request.contentMD5 = @"your md5 which should be encoded by base64"

[client putObject:request completionHandler:^(OBSPutObjectResponse *response, NSError *error){
    NSLog(@"%@",response.etag);
}];
```

NOTE

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

Setting the Storage Class for an Object

You can set **storageClass** to specify the storage class for an object. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];
NSString *filePath = [[NSBundle mainBundle]pathForResource:@"fileName" ofType:@"Type"];
//Upload a file.
OBSPutObjectWithFileRequest *request = [[OBSPutObjectWithFileRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" uploadFilePath:filePath];

// Set the storage class for an object.
request.storageClass = OBSStorageClassStandard;

[client putObject:request completionHandler:^(OBSPutObjectResponse *response, NSError *error){
    NSLog(@"%@",response.etag);
}];
```


NOTE

- The storage class of the objects in a bucket is the same as that of the 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. For details, see **Downloading an Archive Object**.

Customizing Metadata for an Object

You can set **metaDataDict** to customize metadata for an object. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Upload a character string in streaming mode.
OBSPutObjectWithDataRequest *request = [[OBSPutObjectWithDataRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" uploadData:[@""
dataUsingEncoding:NSUTF8StringEncoding]];

// Set the bucket metadata.
request.metaDataDict = @{@"meta1":@"value1",@"meta2":@"value2"};

[client putObject:request completionHandler:^(OBSPutObjectResponse *response, NSError *error){
    NSLog(@"%@",response);
}];
```

NOTE

- In the preceding code, two pieces of metadata named **meta1** and **meta2** are customized and their respective values are set to **value1** and **value2**.
- An object can have multiple pieces of metadata whose total size cannot exceed 8 KB.
- The customized object metadata can be obtained by using **OBSGetObjectMetaDataRequest**. For details, see **Obtaining Object Properties**.
- When you use **getObject** to download an object, its customized metadata will also be downloaded.

Setting the Type for an Object to Be Uploaded

You can set **contentType** to specify the type for an object. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
```



```
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];
NSString *filePath = [[NSBundle mainBundle]pathForResource:@"fileName" ofType:@"mp4"];
//Upload a file.
OBSPutObjectWithFileRequest *request = [[OBSPutObjectWithFileRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" uploadFilePath:filePath];

// Set the type for the object.
request.contentType = OBSContentTypeMP4;

[client putObject:request completionHandler:^(OBSPutObjectResponse *response, NSError *error){
    NSLog(@"%@",response.etag);
}];
```

Type	Enumeration Value in OBS iOS SDK
video/mp4	OBSContentTypeMP4
text/html	OBSContentTypeHTML
image/png	OBSContentTypePNG
image/jpeg	OBSContentTypeJPEG
image/gif	OBSContentTypeGIF
application/pdf	OBSContentTypePDF
audio/mp3	OBSContentTypeMP3
audio/wav	OBSContentTypeWAV
binary/octet-stream	OBSContentTypeBinary
video/quicktime	OBSContentTypeMOV
application/vnd.apple.mpegurl	OBSContentTypeM3U8

You can also set the **customContentType** field to specify the type for an object to be uploaded. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];
NSString *filePath = [[NSBundle mainBundle]pathForResource:@"fileName" ofType:@"mp4"];
//Upload a file.
```



```
OBSPutObjectWithFileRequest *request = [[OBSPutObjectWithFileRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" uploadFilePath:filePath];

// Set the type for the object.
request.customContentType = @"video/mp4";

[client putObject:request completionHandler:^(OBSPutObjectResponse *response, NSError *error){
    NSLog(@"%@",response.etag);
}];
```

NOTE

- The **contentType** parameter supports 11 object types listed in the preceding table. If no type is specified, the default type **binary/octet-stream** is used.
- Different from **contentType**, **customContentType** allows you to use character strings to specify any type of objects to be uploaded. If this parameter is not set, the object type is determined by the value of **contentType** upon upload.
- If both **customContentType** and **contentType** are specified, the type of the object to be uploaded is determined by **customContentType**.

6.6 Performing a Multipart Upload

Multipart upload is applicable to many scenarios, including:

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

Multipart upload consists of three phases:

Step 1 Initialize a multipart upload (**initiateMultipartUpload**).

Step 2 Upload parts one by one or concurrently (**uploadPart**).

Step 3 Combine parts (**completeMultipartUpload**) or abort the multipart upload (**abortMultipartUpload**).

----End

Initializing a Multipart Upload

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

You can call **initiateMultipartUpload** to initialize a multipart upload.

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
```



```
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Initialize a multipart upload.
OBSInitiateMultipartUploadRequest *request = [[OBSInitiateMultipartUploadRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname"];

[client initiateMultipartUpload:request completionHandler:^(OBSInitiateMultipartUploadResponse
*response, NSError *error) {
    NSLog(@"%@",response);
}];
```

NOTE

- The upload ID of the multipart upload returned by **response.uploadID** will be used in follow-up operations.
- In **OBSInitiateMultipartUploadRequest**, you can specify the MIME type, storage class, and customized metadata for the object. In addition, you can set the type of objects to be upload by setting **customContentType**.
- When **background** is set to **YES**, background upload is enabled.

Uploading Parts

After initializing a multipart upload, you can specify the object name and upload ID to upload parts. Each 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 comparative locations in the object. If you use the same part number to upload two parts, the latter one being uploaded will overwrite the former. Except for the last uploaded part whose size ranges from 0 to 5 GB, sizes of the other parts range from 100 KB to 5 GB. Parts are uploaded in random order and can be uploaded through different processes or machines. OBS will combine them into the object based on their part numbers.

You can call **uploadPart** to upload parts.

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];
NSString *filePath = [[NSBundle mainBundle]pathForResource:@"fileName" ofType:@"Type"];
// First part
OBSUploadPartWithFileRequest* fileRequest_first =
[[OBSUploadPartWithFileRequest alloc]initWithBucketName:@"bucket-ios-test03"
                                              objectkey:@"MultiPart"
                                             partNumber:[NSNumber numberWithInt:1]
                                               uploadID:@"uploadID"
                                         uploadFilePath:filePath];
// Enable background upload. When an application is switched to the background, the ongoing upload
continues.
fileRequest_first.background = YES;
```



```
[client uploadPart:fileRequest_first completionHandler:^(OBSUploadPartResponse *response, NSError *error)
{
    NSLog(@"Part one");
}];

// Second part
OBSUploadPartWithFileRequest* fileRequest_sec =
[[OBSUploadPartWithFileRequest alloc]initWithBucketName:@"bucket-ios-test03"
                                              objectkey:@"MultiPart"
                                             partNumber:[NSNumber numberWithInt:2]
                                               uploadID:@"uploadID"
                                         uploadFilePath:filePath];

[client uploadPart:fileRequest_sec completionHandler:^(OBSUploadPartResponse *response, NSError *error)
{
    NSLog(@"Part two");
}];
```

NOTE

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

Combining Parts

After all parts are uploaded, call the API for combining parts to form 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 **completeMultipartUpload** to combine parts.

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// List parts.
__block NSMutableArray *partsList;
OBSListPartsRequest* listRequest = [[OBSListPartsRequest alloc]initWithBucketName:@"bucketname"
objectKey:@"objectname" uploadID:@"uploadID"];
OBSBFTask *listTask = [client listParts:listRequest completionHandler:^(OBSListPartsResponse *response,
NSError *error) {
    partsList = [response.partsList mutableCopy];
```



```
    // Combine parts.
   OBSCompleteMultipartUploadRequest* comRequest = [[OBSCompleteMultipartUploadRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" uploadID:@"uploadID"];
    comRequest.partsList = partsList;
  [client completeMultipartUpload:comRequest
completionHandler:^(OBSCompleteMultipartUploadResponse *response, NSError *error) {
        NSLog(@"%@",response);
    }];

}];
```

Aborting a Multipart Upload

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

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

You can call **abortMultipartUpload** to abort a multipart upload.

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Abort a multipart upload.
OBSAbortMultipartUploadRequest *abortRequest = [[OBSAbortMultipartUploadRequest alloc]
initWithBucketName:@"bucketname" objectKey:@"objectname" uploadID:@"uploadID"];

[client abortMultipartUpload:abortRequest completionHandler:^(OBSAbortMultipartUploadResponse
*response, NSError *error) {
    NSLog(@"%@",response);
}];
```

Listing Uploaded Parts

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

The following table describes the parameters involved in this API.

Parameter	Description	Method in OBS iOS SDK
bucketName	Bucket name	request.bucketName
objectKey	Object name	request.objectKey

Parameter	Description	Method in OBS iOS SDK
uploadID	Upload ID, which globally identifies a multipart upload. The value is in the returned result of **initiateMultipartUpload**.	request.uploadID
maxParts	Maximum number of parts that can be listed per page.	request.maxParts
partNumberMarker	Part number after which listing parts begins. Only parts whose part numbers are larger than this value will be listed.	request.partNumberMarker

- Listing parts in simple mode

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";
NSString *uploadID = @"upload id from OBSInitateMultipartUpload";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];


OBSListPartsRequest* listRequest = [[OBSListPartsRequest alloc]initWithBucketName:@"bucketname"
objectKey:@"objectname" uploadID:uploadID];
[client listParts:listRequest completionHandler:^(OBSListPartsResponse *response, NSError *error) {
    NSLog(@"%@",response);
}];
```

NOTE

- Information about a maximum of 1000 parts can be returned each time. If an upload of the specific upload ID contains more than 1000 parts and **response.isTruncated** is **YES** in the returned result, not all parts will be returned. In such cases, you can use **response.getNextPartNumberMarker** 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

If the number of parts of a multipart upload is larger than 1000, you can use the following sample code to list all parts.

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";
NSString *uploadID = @"upload id from OBSInitateMultipartUpload";
```



```
// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];


__block OBSListPartsResponse *result;

OBSListPartsRequest* listRequest = [[OBSListPartsRequest alloc]initWithBucketName:@"bucketname"
objectKey:@"objectname" uploadID:uploadID];

// List all uploaded parts.
do {
    dispatch_semaphore_t sema = dispatch_semaphore_create(0);
    [client listParts:listRequest completionHandler:^(OBSListPartsResponse *response, NSError *error) {
        result = response;
        NSLog(@"%@",result);
        listRequest.partNumberMarker = result.nextPartNumberMarker;
        dispatch_semaphore_signal(sema);
    }];
    dispatch_semaphore_wait(sema, DISPATCH_TIME_FOREVER);

} while (result.isTruncated);
```

- Listing all parts in paging mode

The previously described listing (1000 parts per page) is a special paging listing mode. The following sample code shows how to specify the number of parts displayed per page when listing.

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";
NSString *uploadID = @"upload id from OBSInitateMultipartUpload";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

 __block OBSListPartsResponse *result;

OBSListPartsRequest* listRequest = [[OBSListPartsRequest alloc]initWithBucketName:@"bucketname"
objectKey:@"objectname" uploadID:uploadID];
listRequest.maxUploads = [NSNumber numberWithInt:100];
// List all uploaded parts.
do {
    dispatch_semaphore_t sema = dispatch_semaphore_create(0);
    [client listParts:listRequest completionHandler:^(OBSListPartsResponse *response, NSError *error) {
       result = response;
        NSLog(@"%@",result);
        listRequest.partNumberMarker = result.nextPartNumberMarker;
        dispatch_semaphore_signal(sema);
    }];
    dispatch_semaphore_wait(sema, DISPATCH_TIME_FOREVER);
```



```
} while (result.isTruncated);
```

Listing Multipart Uploads

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

Parameter	Description	Method in OBS iOS SDK
bucketName	Bucket name	request.bucketName = @"bucketname"
delimiter	Character used to group object names involved in multipart uploads. If the object name contains the **delimiter** parameter, the character string from the first character to the first delimiter in the object name is grouped under a single result element, **commonPrefix**. (If a prefix is specified in the request, the prefix must be removed from the object name.)	request.delimiter = @"delimiter"
prefix	Prefix that the object names in the multipart uploads to be listed must contain	request.prefix = @"prefix"
maxUploads	Maximum number of listed multipart uploads. The value ranges from 1 to 1000. If the value is not in this range, 1000 parts are listed by default.	request.maxUploads = [NSNumber numberWithInt:Int]
keyMarker	Object name to start with when listing multipart uploads	request.keyMarker = @"keymarker"
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.	request.uploadIDMarker = @"ifmarker"

- Listing multipart uploads in simple mode

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
```



```
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// List multipart uploads.
OBSListMultipartUploadsRequest *request = [[OBSListMultipartUploadsRequest
alloc]initWithBucketName:@"bucketname"];

[client listMultipartUploads:request completionHandler:^(OBSListMultipartUploadsResponse *response,
NSError *error) {
    NSLog(@"%@",response);
}];
```

NOTE

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

- Listing all multipart uploads

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];
__block OBSListMultipartUploadsResponse *result;
OBSListMultipartUploadsRequest *request = [[OBSListMultipartUploadsRequest
alloc]initWithBucketName:@"bucket-ios-test03"];

//List all multipart uploads.
do {
    dispatch_semaphore_t sema = dispatch_semaphore_create(0);
    [client listMultipartUploads:request completionHandler:^(OBSListMultipartUploadsResponse *response,
NSError *error) {
        result = response;
        NSLog(@"%@",result);
        request.keyMarker = result.nextKeyMarker;
        request.uploadIDMarker = result.nextUploadIDMarker;
        dispatch_semaphore_signal(sema);
        }];
    dispatch_semaphore_wait(sema, DISPATCH_TIME_FOREVER);
} while (result.isTruncated);
```

- Listing all multipart uploads in paging mode

The previous sample code (listing 1000 uploads per page) shows a special paging listing mode. The following sample code shows how to specify the number of uploads displayed per page when listing.


```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];


__block OBSListMultipartUploadsResponse *result;


OBSListMultipartUploadsRequest *request = [[OBSListMultipartUploadsRequest
alloc]initWithBucketName:@"bucketname"];

// Set the number of multipart uploads displayed per page to 100.
request.maxUploads = [NSNumber numberWithInt:100];

//List all multipart uploads.
do {
    dispatch_semaphore_t sema = dispatch_semaphore_create(0);
    [client listMultipartUploads:request completionHandler:^(OBSListMultipartUploadsResponse *response,
NSError *error) {
        result = response;
        NSLog(@"%@",result);
        request.keyMarker = result.nextKeyMarker;
        request.uploadIDMarker = result.nextUploadIDMarker;
        dispatch_semaphore_signal(sema);

    }];
    dispatch_semaphore_wait(sema, DISPATCH_TIME_FOREVER);

} while (result.isTruncated);
```

6.7 Configuring Lifecycle Management

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];
NSString *filePath = [[NSBundle mainBundle]pathForResource:@"fileName" ofType:@"Type"];
// Upload a file.
OBSPutObjectWithFileRequest *request = [[OBSPutObjectWithFileRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" uploadFilePath:filePath];

// Specify that the file will expire 30 days after creation.
request.expires = [NSNumber numberWithInteger:30];
```



```
[client putObject:request completionHandler:^(OBSPutObjectResponse *response, NSError *error){
   NSLog(@"%@",response.etag);
}];
```

NOTE

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

6.8 Appendable Upload

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

```
   NSString *filePath = [[NSBundle mainBundle] pathForResource:@"FileName" ofType:@"FileSuffix"];
   NSFileManager *manager = [NSFileManager defaultManager];
   NSDictionary *fileDic = [manager attributesOfItemAtPath:filePath error:nil];
   unsigned long long size = [[fileDic objectForKey:NSFileSize] longLongValue];
   int filesize = size;
   //Create an object in appendable mode.
   OBSAppendObjectWithFileRequest *request = [[OBSAppendObjectWithFileRequest alloc]
initWithBucketName:@"bucketName" objectKey:@"objectname" uploadFilePath:filePath];
   request.position = [NSNumber numberWithFloat:0];

   request.uploadProgressBlock =  ^(int64_t bytesSent, int64_t totalBytesSent, int64_t
totalBytesExpectedToSend) {
       NSLog(@"%0.1f%%",(float)floor(totalBytesSent*10000/totalBytesExpectedToSend)/100);

   };
   [client appendObject:request completionHandler:^(OBSAppendObjectResponse *response, NSError
*error) {
       NSLog(@"%@",response);
       //Start position for next appending
      Int position = response.nextPosition;
   }];

   //Append data to the object.
   OBSAppendObjectWithFileRequest *request = [[OBSAppendObjectWithFileRequest alloc]
initWithBucketName:@"bucketName" objectKey:@"objectname" uploadFilePath:filePath];
   request.position = [NSNumber numberWithFloat:size];

   request.uploadProgressBlock =  ^(int64_t bytesSent, int64_t totalBytesSent, int64_t
totalBytesExpectedToSend) {
       NSLog(@"%0.1f%%",(float)floor(totalBytesSent*10000/totalBytesExpectedToSend)/100);

   };
   [client appendObject:request completionHandler:^(OBSAppendObjectResponse *response, NSError
*error) {
       NSLog(@"%@",response);
       //Start position for next appending
       Int position = response.nextPosition;
   }];
```


NOTE

- Objects uploaded using **putObject**, referred to as normal objects, can overwrite objects uploaded using **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 thrown (status code **409**) if a normal 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 appended each time can be up to 5 GB, and 10000 times of appendable uploads can be performed on a single object.
- After an appendable upload is successful, you can use **response.nextPosition** or call **getObjectMetadata** to get the start position for next appending.

6.9 Performing a Multipart Copy

As a special case of multipart upload, multipart copy implements multipart upload by copying the whole or part of an object in a bucket. You can call **copyPart** to copy parts. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Copy parts.
OBSCopyPartRequest* copyRequest =
[[OBSCopyPartRequest alloc]initWithSrcBucketName:@"buckeName"
                                    srcObjectKey:@"MultiPart"
                                uploadBucketName:@"bucketName"
                                 uploadObjectKey:@"MultiPart"
                                uploadPartNumber:[NSNumber numberWithInt:3]
                                        uploadID:@"uploadID"];

[client copyPart:copyRequest completionHandler:^(OBSCopyPartResponse *response, NSError *error) {
    NSLog(@"%@",response);
}];
```

6.10 Performing a Resumable Upload

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

indicating a successful upload is returned. Otherwise, an exception is thrown to remind you of calling the API again for re-uploading. Based on the upload status of each part recorded in the checkpoint file, the re-uploading will upload the parts failed to be uploaded previously, instead of uploading all parts. By virtue of this, resources are saved and efficiency is improved.

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

Parameter	Description	Method in OBS iOS SDK
bucketName	(Mandatory) Bucket name	request.bucketName
objectKey	(Mandatory) Object name	request.objectKey
objectACLPolicy	Object access control policy	request.objectACLPolicy
storageClass	Object storage class	request.storageClass
metaDataDict	Object metadata	request.metaDataDict
websiteRedirectLocation	Redirection location	request.websiteRedirect-Location
encryption	Encryption mode	request.encryption
enableCheckpoint	Whether to enable the resumable upload mode. The default value is **NO**, which indicates that this mode is disabled.	request.enableCheckpoint
enableMD5Check	Whether to enable MD5 verification. The default value is **NO**, which indicates that MD5 verification is disabled.	request.enableMD5Check
checkpointFilePath	File used to record the upload progress. This parameter is effective only in the resumable upload mode. If the value is null, the file is in the same directory as the local file to be uploaded. The file name extension can be set to **obsuploadcheckpoint**.	request.checkpointFilePath

Parameter	Description	Method in OBS iOS SDK
partSize	Part size, in bytes. The value ranges from **100 KB** to **5 GB** and defaults to **5 MB**.	request.partSize

Sample code:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Set the maximum number of files that can be uploaded concurrently in multipart mode.
client.configuration.maxConcurrentUploadRequestCount = 5;
// Maximum number of connections for multipart upload requests.
client.configuration.uploadSessionConfiguration.HTTPMaximumConnectionsPerHost = 10;
NSString *filePath = [[NSBundle mainBundle]pathForResource:@"fileName" ofType:@"Type"];
OBSUploadFileRequest *request = [[OBSUploadFileRequest alloc]initWithBucketName:@"bucketname"
objectKey:@"objectname" uploadFilePath:filePath];
// Set the part size to 5 MB.
request.partSize = [NSNumber numberWithInteger: 5 * 1024*1024];
// Enable resumable upload.
request.enableCheckpoint = YES;
// Specify the checkpoint file path.
request.checkpointFilePath = @"Your CheckPoint File"

// Upload a file.
request.uploadProgressBlock =  ^(int64_t bytesSent, int64_t totalBytesSent, int64_t
totalBytesExpectedToSend) {
    NSLog(@"%0.1f%%",(float)floor(totalBytesSent*10000/totalBytesExpectedToSend)/100);
};

OBSBFTask  *task = [client uploadFile:request completionHandler:^(OBSUploadFileResponse *response,
NSError *error) {
     NSLog(@"%@",response);
}];

[task waitUntilFinished];

if(task.error){
    // Continue to upload the file.
}
```


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 **NO**, which indicates that the resumable upload mode is disabled. In such cases, the API for resumable upload degrades to the simple encapsulation of multipart upload, and no checkpoint file will be generated.
- **checkpointFile** is effective only when **enableCheckpoint** is **YES**.
- Currently, when multiple resumable upload tasks need to be executed concurrently, an independent instance of OBSClient needs to be initialized for each upload task to process requests.

7 Object Download

7.1 Object Download Overview

OBS iOS SDK provides APIs for downloading objects in the following modes:

- **Streaming download**
- **Partial download**
- **Resumable download**

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

7.2 Performing a Streaming Download

Sample code:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Streaming download
OBSGetObjectToDataRequest *request = [[OBSGetObjectToDataRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname"];

//Query the download progress.
request.downloadProgressBlock = ^(int64_t bytesWritten, int64_t totalBytesWritten, int64_t
totalBytesExpectedToWrite) {
    NSLog(@"%0.1f%%",(float)(totalBytesWritten)*100/(float)totalBytesExpectedToWrite);
};

// Receive downloaded data.
__block NSMutableData *objectData = [NSMutableData new];
```



```
request.onReceiveDataBlock = ^(NSData *data) {
    [objectData appendData:data];
};

// Download result
[client getObject:request completionHandler:^(OBSGetObjectResponse *response, NSError *error){
    NSLog(@"%@",response);
}] ;
}
```

7.3 Performing a Partial Download

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 0th to the 1000th bytes, 1001 bytes in total, will be returned. If the specified range is invalid, data of the whole object will be returned.

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Streaming download
OBSGetObjectToDataRequest *request = [[OBSGetObjectToDataRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname"];

// Set the start and end positions of the download range to 0 and 10000 respectively.
request.range = @"bytes=0-1000";

//Query the download progress.
request.downloadProgressBlock = ^(int64_t bytesWritten, int64_t totalBytesWritten, int64_t
totalBytesExpectedToWrite) {
    NSLog(@"%0.1f%%",(float)(totalBytesWritten)*100/(float)totalBytesExpectedToWrite);
};

// Receive downloaded data.
__block NSMutableData *objectData = [NSMutableData new];
request.onReceiveDataBlock = ^(NSData *data) {
    [objectData appendData:data];
};

// Download result
[client getObject:request completionHandler:^(OBSGetObjectResponse *response, NSError *error){
    NSLog(@"%@",response);
}] ;
```

NOTE

- If the specified range is invalid (because the start or end position is set to a negative integer or the range is longer than the object length), data of the whole object will be returned.

7.4 Performing a Conditioned Download

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

You can set the following conditions:

Parameter	Description	Method in OBS iOS SDK
ifModifiedSince	Returns the object if it is modified after the time specified by this parameter; otherwise, an exception is thrown.	request.ifModifiedSince
ifUnmodifiedSince	Returns the object if it remains unchanged since the time specified by this parameter; otherwise, an exception is thrown.	request.ifUnmodifiedSince
ifETagMatch	Returns the source object if its ETag is the same as the one specified by this parameter; otherwise, an exception is thrown.	request.ifETagMatch
ifETagNoneMatch	Returns the source object if its ETag is different from the one specified by this parameter; otherwise, an exception is thrown.	request.ifETagNoneMatch

NOTE

- The ETag of an object is the MD5 check value of the object.
- If the specified condition is not met, error **Precondition Failed** will be returned.

Sample code:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
```



```
client = [[OBSClient alloc] initWithConfiguration:conf];

// Streaming download
OBSGetObjectToDataRequest *request = [[OBSGetObjectToDataRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname"];

// Set conditions.
request.ifModifiedSince = [[OBSUtils getDateFormatterRFC1123]dateFromString:@"Mon, 18 Dec 2017
03:50:49 GMT"];
// Check whether the ETags are the same.
request.ifETagMatch = @"123223";


//Query the download progress.
request.downloadProgressBlock = ^(int64_t bytesWritten, int64_t totalBytesWritten, int64_t
totalBytesExpectedToWrite) {
    NSLog(@"%0.1f%%",(float)(totalBytesWritten)*100/(float)totalBytesExpectedToWrite);
};

// Receive downloaded data.
__block NSMutableData *objectData = [NSMutableData new];
request.onReceiveDataBlock = ^(NSData *data) {
    [objectData appendData:data];
};

// Download result
[client getObject:request completionHandler:^(OBSGetObjectResponse *response, NSError *error){
    NSLog(@"%@",response);
}] ;
```

NOTE

When calling **OBSGetObjectToFileRequest**, you can set **background** to **YES** to enable background download.

7.5 Rewriting Response Headers

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

Parameter	Description	Method in OBS iOS SDK
responseContentType	Rewrites **Content-Type** in HTTP/HTTPS responses.	request.responseContentType
responseContentLanguage	Rewrites **Content-Language** in HTTP/HTTPS responses.	request.responseContentLanguage
responseExpires	Rewrites **Expires** in HTTP/HTTPS responses.	request.responseExpires
responseCacheControl	Rewrites **Cache-Control** in HTTP/HTTPS responses.	request.responseCacheControl
responseContentDisposition	Rewrites **Content-Disposition** in HTTP/HTTPS responses.	request.responseContentDisposition

Parameter	Description	Method in OBS iOS SDK
responseContentEncoding	Rewrites **Content-Encoding** in HTTP/HTTPS responses.	request.responseContentEncoding

Sample code:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Streaming download
OBSGetObjectToDataRequest *request = [[OBSGetObjectToDataRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname"];

//Rewrite ContentType.
request.responseContentType = @"image/jpeg";


//Query the download progress.
request.downloadProgressBlock = ^(int64_t bytesWritten, int64_t totalBytesWritten, int64_t
totalBytesExpectedToWrite) {
    NSLog(@"%0.1f%%",(float)(totalBytesWritten)*100/(float)totalBytesExpectedToWrite);
};

// Receive downloaded data.
__block NSMutableData *objectData = [NSMutableData new];
request.onReceiveDataBlock = ^(NSData *data) {
    [objectData appendData:data];
};

// Download result
[client getObject:request completionHandler:^(OBSGetObjectResponse *response, NSError *error){
    NSLog(@"%@",response);
}] ;
```

7.6 Obtaining Customized Metadata

After an object is successfully downloaded, you can obtain its customized data (**mataDataDict**) from the response.

7.7 Downloading an Archive Object

If you want to download an Archive object, you need to restore the object first. Three restore options are supported, as described in the following table.

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

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

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Restore an object.
OBSRestoreObjectRequest *request = [[OBSRestoreObjectRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" storeDays:[NSNumber numberWithInt:
30]]; //1 to 30
request.restoreTier = OBSRestoreTierExpedited;

OBSBFTask *task = [ self.client restoreObject:request completionHandler:^(OBSRestoreObjectResponse
*response, NSError *error){
    NSLog(@"%@",response);
}];

// Wait until the object is restored.
sleep(6*60);

// Download an object.
NSString * outfilePath = [NSTemporaryDirectory() stringByAppendingString:@"filename"];
OBSGetObjectToFileRequest *request1 = [[OBSGetObjectToFileRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" downloadFilePath:outfilePath];
request1.downloadProgressBlock = ^(int64_t bytesWritten, int64_t totalBytesWritten, int64_t
totalBytesExpectedToWrite) {
    NSLog(@"%0.1f%%",(float)floor(totalBytesWritten*10000/totalBytesExpectedToWrite)/100);
};

[client getObject:request1 completionHandler:^(OBSGetObjectResponse *response, NSError *error){
    NSLog(@"%@",response.etag);
}];
```

7.8 Performing a Resumable Download

Downloading large files often fails due to poor network conditions or program breakdowns. It is a waste of resources to restart the download process upon a download failure, and the restarted download process may still suffer from the unstable network. To resolve such issues, you can use the API for resumable

download, whose working principle is to divide the to-be-downloaded file into multiple parts and download them separately. The download result of each part is recorded in a checkpoint file in real time. Only when all parts are successfully downloaded, the result indicating a successful download is returned. Otherwise, an exception is thrown to remind you of calling the API again for re-downloading. Based on the download status of each part recorded in the checkpoint file, the re-downloading will download the parts failed to be downloaded previously, instead of downloading all parts. By virtue of this, resources are saved and efficiency is improved.

You can use **downloadFile** to perform a resumable download. The following table describes the parameters involved in this API.

Parameter	Description	Method in OBS iOS SDK
bucketName	(Mandatory) Bucket name	request.bucketName
objectKey	(Mandatory) Object name	request.objectKey
downloadFilePath	Full path of the local directory to which the object is downloaded	request.downloadFilePath
versionID	Object version ID	request.versionID
enableCheckpoint	Whether to enable the resumable upload mode. The default value is **NO**, which indicates that this mode is disabled.	request.enableCheckpoint
enableMD5Check	Whether to enable MD5 verification	request.enableMD5Check
enableForceOverwrite	Whether to enable forcible overwriting	request.enableForceOverwrite
checkpointFilePath	File used to record the download progress. This parameter is effective only in the resumable download mode. If the value is null, the file is in the same local directory as the downloaded object.	request.checkpointFilePath
partSize	Part size, in bytes. The value ranges from 5 MB to 5 GB.	request.partSize

Parameter	Description	Method in OBS iOS SDK
ifModifiedSince	Returns the object if it is modified after the time specified by this parameter; otherwise, an exception is thrown.	request.ifModifiedSince
ifUnmodifiedSince	Returns the object if it remains unchanged since the time specified by this parameter; otherwise, an exception is thrown.	request.ifUnmodifiedSince
ifETagMatch	Returns the source object if its ETag is the same as the one specified by this parameter; otherwise, an exception is thrown.	request.ifETagMatch
ifETagNoneMatch	Returns the source object if its ETag is different from the one specified by this parameter; otherwise, an exception is thrown.	request.ifETagNoneMatch

Sample code:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";


// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Storage path
NSString * outfilePath = [NSTemporaryDirectory() stringByAppendingString:@"filename"];
// Maximum number of resumable downloads that can be currently performed
self.client.configuration.maxConcurrentDownloadRequestCount = 5;
// Resumable download
OBSDownloadFileRequest *request = [[OBSDownloadFileRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" downloadFilePath:outfilePath];

// Whether to enable forcible overwriting
request.enableForceOverwrite = YES;
// Part size
request.partSize = [NSNumber numberWithInteger:5*1024*1024];
// Whether to enable resumable download
request.enableCheckpoint = YES;
```



```
request.downloadProgressBlock = ^(int64_t bytesWritten, int64_t totalBytesWritten, int64_t
totalBytesExpectedToWrite) {
    NSLog(@"%0.1f%%",(float)floor(totalBytesWritten*10000/totalBytesExpectedToWrite)/100);

};
OBSBFTask  *task = [client downloadFile:request completionHandler:^(OBSDownloadFileResponse
*response, NSError *error) {
    NSLog(@"%@",response);
}];

[task waitUntilFinished];

if(task.error){
    // Perform the download again.
}
```


8 Object Management

8.1 Obtaining Object Properties

You can call **getObjectMetaData** to obtain properties of an object, including the length, MIME type, and customized metadata. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Obtain object properties.
OBSGetObjectMetaDataRequest *request = [[OBSGetObjectMetaDataRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname"];


[client getObjectMetaData:request completionHandler:^(OBSGetObjectMetaDataResponse *response,
NSError *error){
    NSLog(@"meta:%@\n storageClass:%@\n websiteRedirectlocation:%@\n size:
%@",response.metaDataDict,response.storageClass,response.websiteRedirectLocation,response.size);
}];
```

8.2 Managing Object ACLs

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 two modes:

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

Specifying a Pre-defined Access Control Policy During Object Upload

Sample code:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Upload a file.
OBSPutObjectWithFileRequest *request = [[OBSPutObjectWithFileRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" uploadFilePath:_imagePath];

//Set the access control policy of the object to public-read-write.
request.objectACLPolicy = OBSACLPolicyPublicRead;


request.uploadProgressBlock = ^(int64_t bytesSent, int64_t totalBytesSent, int64_t
totalBytesExpectedToSend) {
    NSLog(@"%0.1f%%",(float)floor(totalBytesSent*10000/totalBytesExpectedToSend)/100);
};

[client putObject:request completionHandler:^(OBSPutObjectResponse *response, NSError *error){
    NSLog(@"%@",response.etag);
}];
```

Directly Setting the Object ACL

Sample code:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
 OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

//Set the object ACL directly.
// Initialize an instance of OBSUser.
OBSUser *owner = [[OBSUser alloc] initWithID:@"owner id"];
// Grantees
OBSACLGranteeUser *grantee = [[OBSACLGranteeUser alloc]initWithID:@"grantee id"];
//Object ACL
OBSACLGrant *grant = [[OBSACLGrant alloc]initWithGrantee:grantee permission:OBSACLFull_Control];
OBSAccessControlPolicy *policy = [OBSAccessControlPolicy new];
policy.owner = owner;
[policy.accessControlList addObject:grant];
for(int i=0;i<=20;i++){
```



```
    [policy.accessControlList addObject:grant];
}

// Set the object ACL.
OBSSetObjectACLRequest *request = [[OBSSetObjectACLRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" accessControlPolicy:policy];
[client setObjectACL:request completionHandler:^(OBSSetObjectACLResponse *response, NSError *error){
    NSLog(@"%@",response);
}];
```

NOTE

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

Obtaining the Object ACL

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

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Obtain the ACL of an object.
OBSGetObjectACLRequest *request = [[OBSGetObjectACLRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname"];
[client getObjectACL:request completionHandler:^(OBSGetObjectACLResponse *response, NSError *error){
    NSLog(@"%@",response);
}];
```

8.3 Listing Objects

You can call **listObjects** to list objects in a bucket.

The following table describes the parameters involved in this API.

Parameter	Description	Method in OBS iOS SDK
bucketName	Bucket name	request.bucketName
prefix	Name prefix that the objects to be listed must contain	request.prefix
marker	Object name to start with when listing objects in a bucket. All objects are listed in the lexicographical order.	request.marker

Parameter	Description	Method in OBS iOS SDK
maxKeys	Maximum number of objects listed in the response. The value ranges from 1 to 1000. If the value is not in this range, 1000 objects are listed by default.	request.maxKeys
delimiter	Character used to group object names. If the object name contains the **delimiter** parameter, the character string from the first character to the first delimiter in the object name is grouped under a single result element, **commonPrefix**. (If a prefix is specified in the request, the prefix must be removed from the object name.)	request.delimiter

Listing Parts in Simple Mode

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

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";


// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// List objects.
OBSListObjectsRequest *request = [[OBSListObjectsRequest alloc] initWithBucketName:@"bucketname"];

[client listObjects:request completionHandler:^(OBSListObjectsResponse *response, NSError *error) {
    for (int i =0; i<response.contentsList.count; i++) {
        NSLog(@"%@ \n",response.contentsList[i].key);
    }
}];
```

NOTE

- Information about a maximum of 1000 objects can be listed each time. If a bucket contains more than 1000 objects and **response.isTruncated** is **YES** in the returned result, not all objects are listed. In such cases, you can use **response.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 Versioning Objects by Specifying the Number

Sample code:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// List objects.
OBSListObjectsRequest *request = [[OBSListObjectsRequest alloc] initWithBucketName:@"bucketname"];
// Set the number of objects to be listed to 1.
request.maxKeys = [NSNumber numberWithInt:1];

[client listObjects:request completionHandler:^(OBSListObjectsResponse *response, NSError *error) {
    for (int i =0; i<response.contentsList.count; i++) {
       NSLog(@"%@ \n",response.contentsList[i].key);
  }
}];
```

Listing Versioning Objects by Specifying a Prefix

Sample code:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// List objects.
OBSListObjectsRequest *request = [[OBSListObjectsRequest alloc] initWithBucketName:@"bucketname"];
// Set the prefix to testdir2.
request.prefix = @"/testdir2/";

[client listObjects:request completionHandler:^(OBSListObjectsResponse *response, NSError *error) {
    for (int i =0; i<response.contentsList.count; i++) {
        NSLog(@"%@ \n",response.contentsList[i].key);
    }
}];
```

Listing Versioning Objects by Specifying the Start Position

Sample code:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
```



```
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// List objects.
OBSListObjectsRequest *request = [[OBSListObjectsRequest alloc] initWithBucketName:@"bucketname"];

//Configure that objects whose names are following test in lexicographical order will be listed.
request.marker = @"test";
[client listObjects:request completionHandler:^(OBSListObjectsResponse *response, NSError *error) {
    for (int i =0; i<response.contentsList.count; i++) {
        NSLog(@"%@ \n",response.contentsList[i].key);
    }
}];
```

Listing All Objects in Paging Mode

Sample code:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

__block OBSListObjectsResponse *result;

// List objects.
OBSListObjectsRequest *request = [[OBSListObjectsRequest alloc] initWithBucketName:@"bucketname"];

request.maxKeys = [NSNumber numberWithInt:1];
//List all objects.
do {
    dispatch_semaphore_t sema = dispatch_semaphore_create(0);
    [client listObjects:request completionHandler:^(OBSListObjectsResponse *response, NSError *error) {
        result = response;
     for (int i =0; i<response.contentsList.count; i++) {
        NSLog(@"%@ \n",response.contentsList[i].key);
     }
        request.marker = result.nextMarker;
        dispatch_semaphore_signal(sema);
   }];
   dispatch_semaphore_wait(sema, DISPATCH_TIME_FOREVER);
} while (result.isTruncated);
```


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:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// List objects.
OBSListObjectsRequest *request = [[OBSListObjectsRequest alloc] initWithBucketName:@"bucketname"];

request.prefix = @"file/";

__block OBSListObjectsResponse *result;
do {
    dispatch_semaphore_t sema = dispatch_semaphore_create(0);
    [client listObjects:request completionHandler:^(OBSListObjectsResponse *response, NSError *error) {
        result = response;for (int i =0; i<response.contentsList.count; i++) {
            NSLog(@"%@ \n",response.contentsList[i].key);
        }
        request.marker = result.nextMarker;
        dispatch_semaphore_signal(sema);
    }];
    dispatch_semaphore_wait(sema, DISPATCH_TIME_FOREVER);} while (result.isTruncated);
```

Listing All Folders in the root Directory

Sample code:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// List objects.
OBSListObjectsRequest *request = [[OBSListObjectsRequest alloc] initWithBucketName:@"bucketname"];

request.delimiter = @"/";
// List folders in the root directory.
[client listObjects:request completionHandler:^(OBSListObjectsResponse *response, NSError *error) {
    for (int i =0; i<response.commonPrefixesList.count; i++) {
```



```
        NSLog(@"%@ \n",response.commonPrefixesList[i].prefix);
    }

}];
```

Listing All Objects According to Folders in a Bucket

Sample code:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// List objects.
OBSListObjectsRequest *request = [[OBSListObjectsRequest alloc] initWithBucketName:@"bucketname"];

request.delimiter = @"/";
// Objects in the root directory
[client listObjects:request completionHandler:^(OBSListObjectsResponse *response, NSError *error) {
    [self listObjectsByPrefix:client request:request result:response];
    for (int i =0; i<response.contentsList.count; i++) {
        NSLog(@"%@ \n",response.contentsList[i].key);
    }

}];
```

listObjectsByPrefix function:

```
-(void) listObjectsByPrefix:(OBSClient*) client request:(OBSListObjectsRequest *) request result:
(OBSListObjectsResponse*) result{

    for (OBSCommonPrefix *prefix in result.commonPrefixesList){
        request.prefix = prefix.prefix;

        [client listObjects:request completionHandler:^(OBSListObjectsResponse *response, NSError *error) {
            // Change asynchronization to synchronization.
            dispatch_semaphore_t sema = dispatch_semaphore_create(0);
            NSLog(@"Objects in folder [%@]:",prefix.prefix);
            for (int i=0; i<response.contentsList.count; i++) {
                NSLog(@"%@ \n",response.contentsList[i].key);
            }
            dispatch_semaphore_wait(sema, DISPATCH_TIME_FOREVER);
            [self listObjectsByPrefix:client request:request result:response];
        }];
    }

}
```


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 object names end with a slash (/), **delimiter** is always a slash (/).
- **result.commonPrefixesList** contains the sub-folders of the requested folder.
- The **listObjectsByPrefix** function requires **OBSListObjectsModel.h** be imported.

8.4 Deleting an Object

Deleting a Single Object

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

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Delete an object.

OBSDeleteObjectRequest *request = [[OBSDeleteObjectRequest alloc] initWithBucketName:@"bucketname"
objectKey:@"objectname"];

[client deleteObject:request completionHandler:^(OBSDeleteObjectResponse *response, NSError *error) {
    NSLog(@"%@",response);
}];
```

Deleting Objects in a Batch

You can call **deleteObjects** to delete multiple 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:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";
```



```
// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Delete objects in a batch.
OBSDeleteObjectsRequest *deleteRequest = [[OBSDeleteObjectsRequest alloc]
initWithBucketName:@"bucketname"];

// List of objects to be deleted
OBSObjectToDelete *object1 = [[OBSObjectToDelete alloc] initWithObjectKey:@"objectname1"];
OBSObjectToDelete *object2 = [[OBSObjectToDelete alloc] initWithObjectKey:@"objectname2"];

deleteRequest.objectList = @[object1,object2];

[client deleteObjects:deleteRequest completionHandler:^(OBSDeleteObjectsResponse *response, NSError
*error) {
   for(int i=0;i<response.deletedList.count;i++){
      NSLog(@"%@\n",response.deletedList[i].key);
   }
}];
```

8.5 Copying an Object

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

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

Copying an Object in Simple Mode

Sample code:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

OBSCopyObjectRequest *request = [[OBSCopyObjectRequest alloc]initWithSrcBucketName:@"source-
bucketname" srcObjectKey:@"objectname1" dstBucketName:@"destination-bucketname"
dstObjectKey:@"objectname2"];

[client copyObject:request completionHandler:^(OBSCopyObjectResponse *response, NSError *error){
    NSLog(@"%@",response);
}] ;
```


Rewriting Object Properties

The following sample code shows how to rewrite object properties.

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

OBSCopyObjectRequest *request = [[OBSCopyObjectRequest alloc]initWithSrcBucketName:@"source-
bucketname" srcObjectKey:@"objectname1" dstBucketName:@"destination-bucketname"
dstObjectKey:@"objectname2"];

// Rewrite properties.
request.dstObjectMetaDirective = OBSMetaDirectiveCopy;

request.dstObjectStorageClass = OBSStorageClassStandard;

request.dstObjectWebsiteRedirectLocation = @"URL";

request.customContentType = @"video/mp4";


[client copyObject:request completionHandler:^(OBSCopyObjectResponse *response, NSError *error){
    NSLog(@"%@",response);
}] ;
```

Copying an Object by Specifying Conditions

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

You can set the following conditions:

Parameter	Description	Enumeration Value in OBS iOS SDK
Copy-Source-if-Match	Copies the source object if its ETag is the same as the one specified by this parameter; otherwise, an exception is thrown.	cpSrcIfETagMatch
Copy-Source-if-None-Match	Copies the source object if its ETag is different from the one specified by this parameter; otherwise, an exception is thrown.	cpSrcIfETagNoneMatch

Parameter	Description	Enumeration Value in OBS iOS SDK
Copy-Source-if-Modified-Since	Copies the source object if it is changed after the time specified by this parameter; otherwise, an exception is thrown.	cpSrcIfModifiedSince
Copy-Source-if-Unmodified-Since	Copies the source object if it is changed before the time specified by this parameter; otherwise, an exception is thrown.	cpSrcIfUnmodifiedSince

NOTE

- The ETag of the source object is the MD5 check value of the source object.
- If **Copy-Source-if-Unmodified-Since**, **Copy-Source-if-Match**, **Copy-Source-if-Modified-Since**, or **Copy-Source-if-None-Match** is included and its specified condition is not met, an exception will be thrown.
- **Copy-Source-if-Modified-Since** and **Copy-Source-if-None-Match** can be used together, and so do **Copy-Source-if-Unmodified-Since** and **Copy-Source-if-Match**.

Sample code:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

OBSCopyObjectRequest *request = [[OBSCopyObjectRequest alloc]initWithSrcBucketName:@"source-
bucketname" srcObjectKey:@"objectname1" dstBucketName:@"destination-bucketname"
dstObjectKey:@"objectname2"];


request.cpSrcIfETagNoneMatch = @"\"f807071206c05630b4d3c92aae4f4448\"";

request.cpSrcIfModifiedSince = @"Sunday, 06-Nov-94 08:49:37 GMT";

//request.cpSrcIfModifiedSince = [[OBSUtils      getDateFormatterRFC1123]dateFromString:@"Mon, 18 Dec
2017 03:50:49 GMT"];

[client copyObject:request completionHandler:^(OBSCopyObjectResponse *response, NSError *error){
    NSLog(@"%@",response);
}] ;
```


Modifying an Object ACL

Sample code:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

OBSCopyObjectRequest *request = [[OBSCopyObjectRequest alloc]initWithSrcBucketName:@"source-
bucketname" srcObjectKey:@"objectname1" dstBucketName:@"destination-bucketname"
dstObjectKey:@"objectname2"];

// Grant the FULL_CONTROL permission.
request.dstObjectACLPolicy = OBSACLFull_Control;
[client copyObject:request completionHandler:^(OBSCopyObjectResponse *response, NSError *error){
    NSLog(@"%@",response);
}] ;
```


9 Temporarily Authorized Access

9.1 Using a Temporary URL for Authorized Access

A temporarily authorized request is a URL temporarily authorized by specifying the AK and SK, request method, and related parameters. This URL contains authentication information and therefore you can use this URL to perform the specific operation in OBS. When the URL is being generated, you need to specify the validity period for it. All sub-classes inherited from **OBSBaseRequest** support temporary authentication.

The following table describes operations allowed for temporarily authorized requests.

Operation	Class in OBS iOS SDK
PUT Bucket	OBSCreateBucketRequest
GET Buckets	OBSListBucketsRequest
DELETE Bucket	OBSDeleteBucketRequest
GET Objects	OBSListObjectsRequest
GET Object versions	OBSListObjectsVersionsRequest
List Multipart Uploads	OBSListMultipartUploadsRequest
Obtain Bucket Metadata	OBSGetBucketMetaDataRequest
GET Bucket location	OBSGetBucketMetaDataRequest
GET Bucket storageinfo	OBSGetBucketStorageInfoRequest
PUT Bucket quota	OBSSetBucketQuotaRequest
GET Bucket quota	OBSGetBucketQuotaRequest
PUT Bucket acl	OBSSetBucketACLWithCannedACLRequest, OBSSetBucketACLWithPolicyRequest

Operation	Class in OBS iOS SDK
GET Bucket acl	OBSGetBucketACLRequest
PUT Bucket logging	OBSSetBucketLoggingRequest
GET Bucket logging	OBSGetBucketLoggingRequest
PUT Bucket policy	OBSSetBucketPolicyRequest, OBSSetBucketPolicy-WithStringRequest
GET Bucket policy	OBSGetBucketPolicyRequest
DELETE Bucket policy	OBSDeleteBucketPolicyRequest
PUT Bucket lifecycle	OBSSetBucketLifecycleRequest
GET Bucket lifecycle	OBSGetBucketLifecycleRequest
DELETE Bucket lifecycle	OBSDeleteBucketLifecycleRequest
PUT Bucket website	OBSSetBucketWebsiteRequest
GET Bucket website	OBSGetBucketWebsiteRequest
DELETE Bucket website	OBSDeleteBucketWebsiteRequest
PUT Bucket versioning	OBSSetBucketVersioningRequest
GET Bucket versioning	OBSGetBucketVersioningRequest
PUT Bucket cors	OBSSetBucketCORSRequest
GET Bucket cors	OBSGetBucketCORSRequest
DELETE Bucket cors	OBSDeleteBucketCORSRequest
PUT Bucket notification	OBSSetBucketNotificationRequest
GET Bucket notification	OBSGetBucketNotificationRequest
OPTIONS Bucket	OBSOptionsBucketRequest
PUT Bucket tagging	OBSSetBucketTaggingRequest
GET Bucket tagging	OBSGetBucketTaggingRequest
DELETE Bucket tagging	OBSDeleteBucketTaggingRequest
PUT Object	OBSPutObjectWithDataRequest, OBSPutObjectWith-FileRequest
Append Object	OBSAppendObjectWithFileRequest
GET Object	OBSGetObjectToDataRequest
PUT Object - Copy	OBSCopyObjectRequest
DELETE Object	OBSDeleteObjectRequest
DELETE Objects	OBSDeleteObjectsRequest

Operation	Class in OBS iOS SDK
Obtain Object Metadata	OBSGetObjectMetaDataRequest
PUT Object acl	OBSSetObjectACLRequest
GET Object acl	OBSGetObjectACLRequest
Initiate Multipart Upload	OBSInitiateMultipartUploadRequest
PUT Part	OBSUploadPartWithDataRequest
PUT Part - Copy	OBSCopyPartRequest
Listing Uploaded Parts	OBSListPartsRequest
Complete Multipart Upload	OBSCompleteMultipartUploadRequest
Abort Multipart Upload	OBSAbortMultipartUploadRequest
OPTIONS Object	OBSOptionsObjectRequest
POST Object restore	OBSRestoreObjectRequest

You can call **createV2PreSignedURL** to create a temporary signed URL for an authorized request. Sample code is as follows:

Listing Objects

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

    // Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];
OBSListObjectsRequest *request = [[OBSListObjectsRequest alloc] initWithBucketName:@"bucketname"];

// Create a V2 authorized access URL.
[client createV2PreSignedURL:request expireAfter:3600 completionHandler:^(NSString *urlString, NSString
*httpVerb, NSDictionary *signedHeaders) {
    NSLog(@"%@",urlString);
}]
```

Obtaining an Object

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";
```



```
// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

    // Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];
OBSGetObjectToDataRequest *request = [[OBSGetObjectToDataRequest alloc]
initWithBucketName:@"bucketname" objectKey:@"objectkey"];

    // Create a V2 authorized access URL.
[client createV2PreSignedURL:request expireAfter:3600 completionHandler:^(NSString *urlString, NSString
*httpVerb, NSDictionary *signedHeaders) {
    NSLog(@"%@",urlString);
}]
```


10 Versioning Management

10.1 Versioning Overview

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 details, see **Versioning**.

10.2 Setting Versioning Status for a Bucket

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

Versioning Status	Description	Value in OBS iOS SDK
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 **OBSListObjectsRequest** is called. You can call **OBSListObjectsVersionsRequest** 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.	OBSVersioningStatusEnabled

Versioning Status	Description	Value in OBS iOS SDK
Suspended	1. Existing objects with version IDs 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 version ID. If an object is deleted with no version ID specified, the object is only attached with a deletion mark and version ID **null**. Objects with version ID **null** are physically deleted. 5. Except for delete markers, storage space occupied by objects with all version IDs is billed.	OBSVersioningStatusSuspended

Sample code:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Enable versioning for a bucket.
OBSBucketVersioningConfiguration *conf1 = [OBSBucketVersioningConfiguration new];

conf1.status = OBSVersioningStatusEnabled;
// Configure versioning.
OBSSetBucketVersioningRequest *request = [[OBSSetBucketVersioningRequest alloc]
initWithBucketName:@"bucketname" configuration: conf];
[client setBucketVersioning:request completionHandler:^(OBSSetBucketVersioningResponse *response,
NSError *error) {
    NSLog(@"%@",response);
}];
```


10.3 Viewing Versioning Status of a Bucket

You can call **getBucketVersioning** to view the versioning status of a bucket. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// View the versioning status.
OBSGetBucketVersioningRequest *request = [[OBSGetBucketVersioningRequest alloc]
initWithBucketName:@"bucketname"];
[client getBucketVersioning:request completionHandler:^(OBSGetBucketVersioningResponse *response,
NSError *error) {
    NSLog(@"%@",response);
}];
```

10.4 Obtaining a Versioning Object

You can set **request.versionID** provided in a sub-class of **OBSAbstractGetObjectRequest** to obtain a versioning object. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Obtain versioning objects.
OBSGetObjectToDataRequest *request = [[OBSGetObjectToDataRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname"];

// Query the version ID of the versioning object.
request.versionID = @"";

//Query the download progress.
request.downloadProgressBlock = ^(int64_t bytesWritten, int64_t totalBytesWritten, int64_t
totalBytesExpectedToWrite) {
    NSLog(@"%0.1f%%",(float)(totalBytesWritten)*100/(float)totalBytesExpectedToWrite);
};
```



```
// Receive downloaded data.
__block NSMutableData *objectData = [NSMutableData new];
request.onReceiveDataBlock = ^(NSData *data) {
    [objectData appendData:data];
};

[ client getObject:request completionHandler:^(OBSGetObjectResponse *response, NSError *error){
    NSLog(@"%@",response);
}] ;
```

NOTE

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

10.5 Copying a Versioning Object

You can call **OBSCopyObjectRequest** to pass the version ID (**versionID**) to copy a versioning object. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

OBSCopyObjectRequest *request = [[OBSCopyObjectRequest alloc]initWithSrcBucketName:@"source-
bucketname" srcObjectKey:@"objectname1" dstBucketName:@"destination-bucketname"
dstObjectKey:@"objectname2"];

// Version ID of the to-be-copied versioning object
request.srcObjectVersionID = @"testVersionID";
[client copyObject:request completionHandler:^(OBSCopyObjectResponse *response, NSError *error){
    NSLog(@"%@",response);
}] ;
```

10.6 Restoring a Versioning Archive Object

You can call **OBSRestoreObjectRequest** to pass the version ID (**versionID**) to restore a versioning Archive object. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];
```



```
// Restore an Archive object.
OBSRestoreObjectRequest *request = [[OBSRestoreObjectRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" storeDays:[NSNumber numberWithInt:
30]];//1 to 30
request.restoreTier = OBSRestoreTierExpedited;

// Version ID of a versioning object
request.versionID = @"Version ID of a versioning object";
[client restoreObject:request completionHandler:^(OBSRestoreObjectResponse *response, NSError *error){
    NSLog(@"%@",response);
}];
```

10.7 Listing Versioning Objects

You can call **OBSListObjectsVersionsRequest** to list versioning objects in a bucket.

The following table describes the parameters involved in this API.

Parameter	Description
bucketName	Bucket name
prefix	Name prefix that the objects 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.
maxKeys	Maximum number of versioning object names listed in the response body. The value ranges from 1 to 1000. If the value is not in this range, 1000 versioning objects are returned by default.
delimiter	Character used to group object names. If the object name contains the **delimiter** parameter, the character string from the first character to the first delimiter in the object name is grouped under a single result element, **commonPrefix**. (If a prefix is specified in the request, the prefix must be removed from the object name.)
versionIDMarker	Object name to start with when listing versioning objects in a bucket. All versioning objects are listed in the lexicographical order by object name and version ID. This parameter must be used together with **keyMarker**.

NOTE

- If the value of **versionIDMarker** is not a version ID specified by **keyMarker**, **versionIDMarker** is invalid.
- The returned result of **OBSListObjectsVersionsRequest** includes the versioning objects and delete markers.

Listing Versioning Objects in Simple Mode

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


```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
 OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

//List versioning objects.
OBSListObjectsVersionsRequest *request = [[OBSListObjectsVersionsRequest alloc]
initWithBucketName:@"bucketname"];

[client listObjectsVersions:request completionHandler:^(OBSListObjectsVersionsResponse *response, NSError
*error) {
    for (int i =0; i<response.versionList.count; i++) {
       NSLog(@"%@ \n",response.versionList[i].key);
   }
 }];
```

NOTE

- Information about a maximum of 1000 versioning objects can be listed each time. If a bucket contains more than 1000 objects and **response.isTruncated** is **YES** in the returned result, not all versioning objects are listed. In such cases, you can use **response.nextKeyMarker** and **response.versionIdMarker** to obtain the start position for 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:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// List versioning objects.
OBSListObjectsVersionsRequest *request = [[OBSListObjectsVersionsRequest alloc]
initWithBucketName:@"bucketname"];

// Set the number of versioning objects to be listed to 100.
request.maxKeys = [NSNumber numberWithInt:100];


[client listObjectsVersions:request completionHandler:^(OBSListObjectsVersionsResponse *response, NSError
*error) {
```



```
    for (int i =0; i<response.versionList.count; i++) {
        NSLog(@"%@ \n",response.versionList[i].key);
    }
}];
```

Listing Versioning Objects by Specifying a Prefix

Sample code:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// List versioning objects.
OBSListObjectsVersionsRequest *request = [[OBSListObjectsVersionsRequest alloc]
initWithBucketName:@"bucketname"];

// Set the prefix.
request.prefix = @"/";

[client listObjectsVersions:request completionHandler:^(OBSListObjectsVersionsResponse *response, NSError
*error) {
    for (int i =0; i<response.versionList.count; i++) {
        NSLog(@"%@ \n",response.versionList[i].key);
    }

}];
```

Listing Versioning Objects by Specifying the Start Position

Sample code:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// List versioning objects.
OBSListObjectsVersionsRequest *request = [[OBSListObjectsVersionsRequest alloc]
initWithBucketName:@"bucketname"];

// Specify the start position for listing.
request.keyMarker = @"/";

[client listObjectsVersions:request completionHandler:^(OBSListObjectsVersionsResponse *response, NSError
```



```
*error) {
    for (int i =0; i<response.versionList.count; i++) {
       NSLog(@"%@ \n",response.versionList[i].key);
   }
}];
```

Listing All Versioning Objects in Paging Mode

Sample code:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// List versioning objects.

__block OBSListObjectsVersionsResponse *result;

    // List objects.
OBSListObjectsVersionsRequest *request = [[OBSListObjectsVersionsRequest alloc]
initWithBucketName:@"bucketname"];

request.maxKeys = [NSNumber numberWithInt:1];
// List all versioning objects.
do {
    dispatch_semaphore_t sema = dispatch_semaphore_create(0);
    [client listObjectsVersions:request completionHandler:^(OBSListObjectsVersionsResponse *response,
NSError *error) {
        result = response;
     for (int i =0; i<response.versionList.count; i++) {
        NSLog(@"%@ \n",response.versionList[i].key);
    }
        request.keyMarker = result.nextKeyMarker;
        dispatch_semaphore_signal(sema);
  }];
   dispatch_semaphore_wait(sema, DISPATCH_TIME_FOREVER);
} while (result.isTruncated);
```

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:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
```



```
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// List versioning objects.

__block OBSListObjectsVersionsResponse *result;

// List objects.
OBSListObjectsVersionsRequest *request = [[OBSListObjectsVersionsRequest alloc]
initWithBucketName:@"bucketname"];

request.delimiter = @"/";
// Objects in the root directory
[client listObjectsVersions:request completionHandler:^(OBSListObjectsVersionsResponse *response, NSError
*error) {

    for (int i =0; i<response.versionList.count; i++) {
        NSLog(@"%@ \n",response.versionList[i].key);
    }
    [self listVersionObjectsByPrefix:client request:request result:response];
}];
```

listVersionObjectsByPrefix function:

```
-(void) listVersionObjectsByPrefix:(OBSClient*) client request:(OBSListObjectsVersionsRequest *) request
result:            (OBSListObjectsVersionsResponse*) result{

    for (NSString * prefix in result.commonPrefixesList){
        NSLog(@"Objects in folder [%@]:",prefix);
        request.prefix = prefix;
        [client listObjectsVersions:request completionHandler:^(OBSListObjectsVersionsResponse *response,
NSError *error) {
            for (int i =0; i<response.versionList.count; i++) {
                NSLog(@"%@ \n",response.versionList[i].key);
            }
            [self listVersionObjectsByPrefix:client request:request result:response];
        }];
    }

}
```

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 object names end with a slash (/), **delimiter** is always a slash (/).
- In the returned result of each recursion, **ListVersionsResult.getVersions** includes the versioning objects in the folder and **ListVersionsResult.getCommonPrefixes** includes the sub-folders in the folder.

10.8 Setting or Obtaining a Versioning Object ACL

Directly Setting a Versioning Object ACL

You can call **OBSSetObjectACLRequest** to input the version ID (**versionID**) to set the ACL for a versioning object. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
```



```
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Set the access control policy for a versioning object.
OBSUser *owner = [[OBSUser alloc] initWithID:@"249e6c2bfbc74c928d5893895543029e"];

OBSACLGranteeUser *grantee = [[OBSACLGranteeUser alloc]initWithID:@"AKjdsjklSKLL/
DSKDSLADLADLjdsjald231124"];
// Grant the FULL_CONTROL permission to authorized users.
OBSACLGrant *grant = [[OBSACLGrant alloc]initWithGrantee:grantee permission:OBSACLFull_Control];
OBSAccessControlPolicy *policy = [OBSAccessControlPolicy new];
policy.owner = owner;
[policy.accessControlList addObject:grant];
for(int i=0;i<=20;i++){
    [policy.accessControlList addObject:grant];
 }

OBSSetObjectACLRequest *request = [[OBSSetObjectACLRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" accessControlPolicy:policy];

request.versionID = @"Version ID of a versioning object";

[client setObjectACL:request completionHandler:^(OBSSetObjectACLResponse *response, NSError *error){
    NSLog(@"%@",response);
}];
```

NOTE

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

Obtaining a Versioning Object ACL

You can call **OBSGetObjectACLRequest** to pass the version ID (**versionID**) to obtain the ACL for a versioning object. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Obtain the access control policy of a versioning object.
OBSGetObjectACLRequest *request = [[OBSGetObjectACLRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname"];
// Set the version ID for a versioning object.
request.versionID = @"Version ID of a versioning object";
```



```
[client getObjectACL:request completionHandler:^(OBSGetObjectACLResponse *response, NSError *error){
    NSLog(@"%@",response);
}];
```

10.9 Deleting Versioning Objects

Deleting a Single Versioning Object

You can call **OBSDeleteObjectRequest** to pass the version ID (**versionID**) to copy a versioning object. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Delete an object.
OBSDeleteObjectRequest *request = [[OBSDeleteObjectRequest alloc] initWithBucketName:@"bucketname"
objectKey:@"objectname"];

// Set the version ID for a versioning object.
request.versionID = @"versionID";

[client deleteObject:request completionHandler:^(OBSDeleteObjectResponse *response, NSError *error) {
    NSLog(@"%@",response);
}];
```

Batch Deleting Versioning Objects

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

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Delete objects in a batch.
OBSDeleteObjectsRequest *deleteRequest = [[OBSDeleteObjectsRequest alloc]
initWithBucketName:@"bucketname"];

// List versioning objects.
OBSObjectToDelete *object = [[OBSObjectToDelete alloc] initWithObjectKey:@"objectname"
```



```
versionID:@"versionID"];
OBSObjectToDelete *object1 = [[OBSObjectToDelete alloc] initWithObjectKey:@"objectname"
versionID:@"versionID"];

deleteRequest.objectList = @[object,object1];

[client deleteObjects:deleteRequest completionHandler:^(OBSDeleteObjectsResponse *response, NSError
*error) {
    NSLog(@"%@",response);
}];
```

11 Lifecycle Management

11.1 Lifecycle Management Overview

OBS allows you to set lifecycle rules for buckets to automatically transit the storage class of an object and delete expired objects, so as 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 with the lasted version, which can be specified in either mode:
  a. How many days after the object is created
  b. Expiration date
- Transition time for a noncurrent object version, which is 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 noncurrent object version will take effect only after versioning is enabled for buckets.

11.2 Setting Lifecycle Rules

You can call **setBucketLifecycle** to set lifecycle rules for a bucket.

Setting an Object Transition Policy

Sample code:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Set lifecycle rules.
OBSLifecycleRule *rule = [[OBSLifecycleRule alloc] initWithID:@"delete-2-days" prefix:@"test/"
status:OBSLifecycleStatusEnabled];
// Specify that objects, after having been created for 30 days, will be transited to OBS Infrequent Access.
OBSLifecycleTransition* transitionStandard = [[OBSLifecycleTransition alloc]initWithDays:[NSNumber
numberWithInteger:30] storageClass:OBSStorageClassStandardIA];
// Specify that objects, after been created for 60 days, will be transited to the specified storage class.
OBSLifecycleTransition* transitionGlacier= [[OBSLifecycleTransition alloc]initWithDays:[NSNumber
numberWithInteger:60] storageClass:OBSStorageClassGlacier];
// Specify that objects, after having been changed to noncurrent versions for 30 days, will be transited to
OBS Standard.
OBSLifecycleNoncurrentVersionTransition* noncurrentTransistionStandard =
[[OBSLifecycleNoncurrentVersionTransition alloc]initWithDays:[NSNumber numberWithInteger:30]
storageClass:OBSStorageClassStandardIA];
//Specify that objects, after having been changed to noncurrent versions for 60 days, will be transited to
OBS Archive.
OBSLifecycleNoncurrentVersionTransition* noncurrentTransistionGlacier=
[[OBSLifecycleNoncurrentVersionTransition alloc]initWithDays:[NSNumber numberWithInteger:60]
storageClass:OBSStorageClassGlacier];


[rule.transitionList addObject:transitionStandard];
[rule.transitionList addObject:transitionGlacier];

[rule.noncurrentVersionTransitionList addObject:noncurrentTransistionStandard];
[rule.noncurrentVersionTransitionList addObject:noncurrentTransistionGlacier];

rule.expiration = expiration;
rule.noncurrentVersionExpiration  = noncurrentExpiration;

OBSSetBucketLifecycleRequest *request = [[OBSSetBucketLifecycleRequest
```



```
alloc]initWithBucketName:@"bucketname" ];
[request.lifecycleRuleList addObject: rule];
OBSLifecycleRule* rule2 = [rule copy];
rule2.ID = @"123";
rule2.prefix = @"test1/";
[request.lifecycleRuleList addObject: rule2];

 [client setBucketLifecycle:request completionHandler:^(OBSSetBucketLifecycleResponse *response, NSError
*error){
        NSLog(@"%@",response);
 }];
```

Setting an Object Expiration Time

Sample code:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Specify the prefix.
OBSLifecycleRule *rule = [[OBSLifecycleRule alloc] initWithID:@"delete-2-days" prefix:@"test/"
status:OBSLifecycleStatusEnabled];

// Set the new expiration time.
OBSLifecycleExpiration* expiration = [[OBSLifecycleExpiration alloc]initWithDays:[NSNumber
numberWithInteger:61]];
// Set the new expiration time for noncurrent versions.
OBSLifecycleNoncurrentVersionExpiration* noncurrentExpiration =
[[OBSLifecycleNoncurrentVersionExpiration alloc]initWithDays:[NSNumber numberWithInteger:61]];

[rule.transitionList addObject:transitionStandard];
[rule.transitionList addObject:transitionGlacier];

[rule.noncurrentVersionTransitionList addObject:noncurrentTransistionStandard];
[rule.noncurrentVersionTransitionList addObject:noncurrentTransistionGlacier];

rule.expiration = expiration;
rule.noncurrentVersionExpiration  = noncurrentExpiration;

OBSSetBucketLifecycleRequest *request = [[OBSSetBucketLifecycleRequest
alloc]initWithBucketName:@"bucketname" ];
[request.lifecycleRuleList addObject: rule];
OBSLifecycleRule* rule2 = [rule copy];
rule2.ID = @"123";
rule2.prefix = @"test1/";
[request.lifecycleRuleList addObject: rule2];

[client setBucketLifecycle:request completionHandler:^(OBSSetBucketLifecycleResponse *response, NSError
*error){
   NSLog(@"%@",response);
}];
```


11.3 Viewing Lifecycle Rules

You can call **getBucketLifecycle** to view the lifecycle rules of a bucket. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];
// View the bucket lifecycle rules.
OBSGetBucketLifecycleRequest *request = [[OBSGetBucketLifecycleRequest alloc]
initWithBucketName:@"bucketname"];
[client getBucketLifecycle:request completionHandler:^(OBSGetBucketLifecycleResponse *response, NSError
*error){
   NSLog(@"%@",response);
}];
```

11.4 Deleting Lifecycle Rules

You can call **deleteBucketLifecycle** to view the lifecycle rules of a bucket. Sample code is as follows:

```
 static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];
// Delete lifecycle rules.
OBSDeleteBucketLifecycleRequest *request = [[OBSDeleteBucketLifecycleRequest alloc]
initWithBucketName:@"bucketname"];
[client deleteBucketLifecycle:request completionHandler:^(OBSDeleteBucketLifecycleResponse *response,
NSError *error){
       NSLog(@"%@",response);
}];
```


12 CORS

12.1 CORS Overview

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**.

12.2 Setting CORS Rules

You can call **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:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Set CORS rules for a bucket.

OBSSetBucketCORSRequest *request = [[OBSSetBucketCORSRequest alloc]
initWithBucketName:@"bucketname"];
OBSCORSRule* rule = [OBSCORSRule new];
// Specify the request method, which can be GET, PUT, DELETE, POST, or HEAD.
rule.allowedMethodList =
@[OBSCORSHTTPGET,OBSCORSHTTPPUT,OBSCORSHTTPPOST,OBSCORSHTTPHEAD];
// Specify the origin of the cross-origin request.
rule.allowedOriginList = @[@"www.example1.com",@"www.example2.com"];
// Set the allowed headers.
rule.allowedHeaderList = @[@"allowedheader1",@"allowedheader2"];
// Specify response headers that users can access using application programs.
```



```
rule.exposeHeaderList = @[@"exposeheader_1",@"exposeheader_2"];
// Specify the browser's cache time of the returned results of OPTIONS requests for specific resources, in
seconds.
rule.maxAgeSeconds = [NSNumber numberWithInt:100];
[request.bucketCORSRuleList addObject:rule];
[client setBucketCORS:request completionHandler:^(OBSSetBucketCORSResponse *response, NSError *error)
{
    NSLog(@"%@",response);
 }];
```

12.3 Viewing CORS Rules

You can call **getBucketCORS** to view CORS rules of a bucket. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Obtain CORS rules of a bucket.
OBSGetBucketCORSRequest *request = [[OBSGetBucketCORSRequest alloc]
initWithBucketName:@"bucketname"];

[client getBucketCORS:request completionHandler:^(OBSGetBucketCORSResponse *response, NSError
*error) {
    NSLog(@"%@",response);
}];
```

12.4 Deleting CORS Rules

You can call **deleteBucketCORS** to delete CORS rules from a bucket. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Delete CORS rules of a bucket.
OBSDeleteBucketCORSRequest *request = [[OBSDeleteBucketCORSRequest alloc]
initWithBucketName:@"bucketname"];
```



```
[client deleteBucketCORS:request completionHandler:^(OBSDeleteBucketCORSResponse *response, NSError
*error) {
    NSLog(@"%@",response);
}];
```


13 Access Logging

13.1 Logging Overview

OBS allows you to configure access logging for buckets. After the configuration, access to buckets will be logged and the logs will be saved in specific buckets in OBS.

For more information, see **Logging**.

13.2 Enabling Bucket Logging

You can call **setBucketLogging** to enable bucket logging. by performing the following two steps.

Step 1 Set the log delivery group's access permissions on the target bucket. (Ensure that the log delivery group has the **WRITE** and **READ_ACP** permissions on the bucket.)

Step 2 Configure logging.

----End

NOTE

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

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:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];
```



```
//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

//Set the bucket ACL.
// Step 1: Set the log delivery group's access permissions.
OBSUser *owner = [[OBSUser alloc] initWithID:@"ownerID"];

OBSACLGranteeLogDelivery *grantee = [OBSACLGranteeLogDelivery new];
OBSACLGrant *grant = [[OBSACLGrant alloc]initWithGrantee:grantee permission:OBSACLFull_Control];

// Set the log delivery group's access permission to FULL_CONTROL.
OBSACLGranteeUser *userGrantee = [[OBSACLGranteeUser alloc]initWithID:@"granteeID"];
OBSACLGrant *userGrant = [[OBSACLGrant alloc]initWithGrantee:userGrantee
permission:OBSACLFull_Control];

OBSACLGranteeAllUsers *alluserGrantee = [OBSACLGranteeAllUsers new];
OBSACLGrant *alluserGrant = [[OBSACLGrant alloc]initWithGrantee:alluserGrantee
permission:OBSACLFull_Control];

OBSAccessControlPolicy *policy = [OBSAccessControlPolicy new];
policy.owner = owner;
[policy.accessControlList addObject:grant];
[policy.accessControlList addObject:userGrant];

OBSSetBucketACLWithPolicyRequest *setACLRequest = [[OBSSetBucketACLWithPolicyRequest
alloc]initWithBucketName:@"bucketname" accessControlPolicy:policy];

[client setBucketACL:setACLRequest completionHandler:^(OBSSetBucketACLResponse *response, NSError
*error){
    NSLog(@"%@",response);
}];


// Step 2: Set bucket logging.
grant = [[OBSACLGrant alloc]initWithGrantee:grantee permission:OBSACLFull_Control];

OBSSetBucketLoggingRequest *request = [[OBSSetBucketLoggingRequest
alloc]initWithBucketName:@"bucketname"];

OBSLoggingEnabled* enabledItem = [[OBSLoggingEnabled alloc]initWithTargetBucket:@"bucketname"
targetPrefix:@"access-log"];

[enabledItem.targetGrantsList addObject:userGrant];
[enabledItem.targetGrantsList addObject:alluserGrant];

[request.loggingEnabledList addObject:enabledItem];
[client setBucketLogging:request completionHandler:^(OBSSetBucketLoggingResponse *response, NSError
*error){
    NSLog(@"%@",response);
}];
```

13.3 Viewing Bucket Logging

You can call **getBucketLogging** to view the logging configuration of a bucket. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
```



```
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

//Obtain the bucket logging configuration.
OBSGetBucketLoggingRequest *request = [[OBSGetBucketLoggingRequest alloc]
initWithBucketName:@"bucketname"];
[client getBucketLogging:request completionHandler:^(OBSGetBucketLoggingResponse *response, NSError
*error){
    NSLog(@"%@",response);
}];
```

13.4 Disabling Bucket Logging

You can call **setBucketLogging** to delete all logs of a bucket so as to disable logging of the bucket. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

OBSSetBucketLoggingRequest *request = [[OBSSetBucketLoggingRequest
alloc]initWithBucketName:@"bucketname"];

[client setBucketLogging:request completionHandler:^(OBSSetBucketLoggingResponse *response, NSError
*error){
    NSLog(@"%@",response);
 }];
```


14 Static Website Hosting

14.1 Static Website Hosting Overview

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**.

14.2 Setting Website Hosting

You can call **setBucketWebsite** to set website hosting on a bucket.

Configuring the Default Homepage and Error Pages

Sample code:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];


OBSWebsiteConfCustom* redirectCustom = [OBSWebsiteConfCustom new];

// Configure the default homepage.
OBSWebsiteConfCustomIndexDocument* indexDocument = [[OBSWebsiteConfCustomIndexDocument
alloc]initWithSuffix:@"index.html"];
```



```
// Configure the error pages.
OBSWebsiteConfCustomErrorDocument* errorDocument = [[OBSWebsiteConfCustomErrorDocument
alloc]initWithKey:@"Error.html"];

OBSWebsiteConfCustomRoutingRule* redirectRule = [OBSWebsiteConfCustomRoutingRule new];

OBSWebsiteConfCustomCondition* condition = [OBSWebsiteConfCustomCondition new];
condition.keyPrefixEquals = @"docs/";
//    condition.httpErrorCodeReturnedEquals = @"404";

// Set the redirection rules.
OBSWebsiteConfCustomRedirect* redirect = [OBSWebsiteConfCustomRedirect new];
redirect.replaceKeyPrefixWith = @"documents/";
redirect.protocol = @"http";
redirect.hostName = @"URL";
//    redirect.replaceKeyWith = @"error.html";
redirect.httpRedirectCode = @"301";


redirectRule.condition = condition;
redirectRule.redirect = redirect;

[redirectCustom.indexDocumentList addObject:indexDocument];
[redirectCustom.errorDocumentList addObject:errorDocument];
[redirectCustom.routingRulesList addObject:redirectRule];

OBSSetBucketWebsiteRequest *request = [[OBSSetBucketWebsiteRequest
alloc]initWithBucketName:@"bucketname" configuration:redirectCustom];

[client setBucketWebsite:request completionHandler:^(OBSSetBucketWebsiteResponse *response, NSError
*error){
    NSLog(@"%@",response);
 }];
```

Configuring Redirection for All Requests

Sample code:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Configure that all requests will be redirected.
OBSWebsiteConfRedirectAll* redirectAll = [OBSWebsiteConfRedirectAll new];
redirectAll.hostName = @"URL";
OBSSetBucketWebsiteRequest *request = [[OBSSetBucketWebsiteRequest
alloc]initWithBucketName:@"bucketname" configuration:redirectAll];

 [client setBucketWebsite:request completionHandler:^(OBSSetBucketWebsiteResponse *response, NSError
*error){
     NSLog(@"%@",response);
 }];
```


14.3 Viewing Hosting Settings

You can call **getBucketWebsite** to view the hosting settings of a bucket. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// View hosting settings.
OBSGetBucketWebsiteRequest *request = [[OBSGetBucketWebsiteRequest alloc]
initWithBucketName:@"bucketname"];
[client getBucketWebsite:request completionHandler:^(OBSGetBucketWebsiteResponse *response, NSError
*error){
    NSLog(@"%@",response);
}];
```

14.4 Deleting Hosting Settings

You can call **deleteBucketWebsite** to delete the hosting settings of a bucket. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Delete hosting settings.
OBSDeleteBucketWebsiteRequest *request = [[OBSDeleteBucketWebsiteRequest alloc]
initWithBucketName:@"bucketname"];
[client deleteBucketWebsite:request completionHandler:^(OBSDeleteBucketWebsiteResponse *response,
NSError *error){
    NSLog(@"%@",response);
}];
```


15 Tag Management

15.1 Tagging Overview

Tags are used to identify and classify OBS buckets.

For more information, see **Tags**.

15.2 Setting Bucket Tags

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

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Set bucket tags.
OBSSetBucketTaggingRequest *request = [[OBSSetBucketTaggingRequest
alloc]initWithBucketName:@"bucketname"];
[request.tagList addObject:[[OBSBucketTag alloc]initWithKey:@"tagkey" value:@"tagvalue"]];
[request.tagList addObject:[[OBSBucketTag alloc]initWithKey:@"tagkey1" value:@"tagvalue1"]];

[client setBucketTagging:request completionHandler:^(OBSSetBucketTaggingResponse *response, NSError
*error){
    NSLog(@"%@",response);
 }];
```

NOTE

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

15.3 Viewing Bucket Tags

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

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Obtain bucket tags.
OBSGetBucketTaggingRequest *request = [[OBSGetBucketTaggingRequest alloc]
initWithBucketName:@"bucketname"];
[client getBucketTagging:request completionHandler:^(OBSGetBucketTaggingResponse *response, NSError
*error){
    NSLog(@"%@",response);
}];
```

15.4 Deleting Bucket Tags

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

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Delete bucket tags.
OBSDeleteBucketTaggingRequest *request = [[OBSDeleteBucketTaggingRequest alloc]
initWithBucketName:@"bucketname"];
[client deleteBucketTagging:request completionHandler:^(OBSDeleteBucketTaggingResponse *response,
NSError *error){
     NSLog(@"%@",response);
}];
```


16 Event Notification

16.1 Event Notification Overview

OBS provides the event notification function. With this function, specified operations on buckets will be sent to you through Simple Message Notification (SMN). Event notification involves the following configurations:

- Unique ID of each event notification. If the user does not specify an ID, OBS will automatically assign one.
- URN of the event notification topic. After detecting a specific event, OBS sends a message to the topic.
- Type of events that need to be notified.
- Filtering rules. Objects can be filtered based on the prefixes or suffixes.

For more information, see **Event Notification**.

16.2 Configuring Event Notification

You can call **setBucketNotification** to set event notification for a bucket. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Configure event notification.
OBSSetBucketNotificationRequest *request = [[OBSSetBucketNotificationRequest
alloc]initWithBucketName:@"bucketname" ];
```



```
OBSNotificationFilter *filter = [OBSNotificationFilter new];

OBSNotificationFilterRule *rule1 = [[OBSNotificationFilterRule alloc] initWithName:@"prefix"
value:@"smn/"];
OBSNotificationFilterRule *rule2 = [[OBSNotificationFilterRule alloc] initWithName:@"suffix" value:@".jpg"];
[filter.filterRuleList addObject:rule1];
[filter.filterRuleList addObject:rule2];

OBSNotificationTopicConfiguration *topicConf = [OBSNotificationTopicConfiguration new];
topicConf.topic = [NSString stringWithFormat:@"urn:smn:%@:%@:topic001",@"region",@"ownerID"] ;
topicConf.ID = @"Id001";
topicConf.filter = filter;
[topicConf.eventList addObject:@"Your Event Type"];

[request.configurationList addObject:topicConf];

[client setBucketNotification:request completionHandler:^(OBSSetBucketNotificationResponse *response,
NSError *error){
    NSLog(@"%@",response);
}];
```

16.3 Viewing Event Notification Settings

You can call **getBucketNotification** to view event notification settings of a bucket. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// View event notification settings.
OBSGetBucketNotificationRequest *request = [[OBSGetBucketNotificationRequest alloc]
initWithBucketName:@"bucketname"];
[client getBucketNotification:request completionHandler:^(OBSGetBucketNotificationResponse *response,
NSError *error){
   NSLog(@"%@",response);
}];
```

16.4 Disabling Event Notification

To disable event notification on buckets is to call **setBucketNotification** to clear all event notification settings. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];
```



```
//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Configure event notification.
OBSSetBucketNotificationRequest *request = [[OBSSetBucketNotificationRequest
alloc]initWithBucketName:@"bucketname" ];

[client setBucketNotification:request completionHandler:^(OBSSetBucketNotificationResponse *response,
NSError *error){
    NSLog(@"%@",response);
}];
```


17 Server-Side Encryption

17.1 Server-Side Encryption Overview

OBS supports server-side encryption.

For more information, see **Server-Side Encryption**.

17.2 Encryption Description

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

API Method in OBS iOS SDK	Description	Supported Encryption Type
putObject	Sets the encryption algorithm and key during object upload to enable server-side encryption.	SSE-KMS, SSE-C
getObject	Sets the decryption algorithm and key during object upload to decrypt the object.	SSE-C
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
getObjectMetadata	Sets the decryption algorithm and key when obtaining the object metadata to decrypt the object.	SSE-C

API Method in OBS iOS SDK	Description	Supported Encryption Type
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
uploadPart	Sets the encryption algorithm and key during multipart upload to enable server-side encryption for parts.	SSE-C
copyPart	1. Sets the decryption algorithm and key for decrypting the source object during object copy. 2. Sets the encryption algorithm and key during partial object copy to enable the encryption algorithm for the target object part.	SSE-C

17.3 Example of Encryption

Encrypting an Object to Be Uploaded

Sample code:

- SSE-C encryption

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";


// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Use the SSE-C algorithm to upload an object.
NSData *uploadData = [NSData dataWithContentsOfFile:_imagePath];
OBSPutObjectWithDataRequest *request = [[OBSPutObjectWithDataRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"test/image1" uploadData:uploadData];

 // Encrypt the object.
request.encryption = [[OBSEncryptionTypeCustomer alloc]initWithAlgorithm:@"AES256"
key:@"K7QkYpBkM5+hcs27fsNkUnNVaobncnLht/rCB2o/9Cw=" keyMD5:@"4XvB3tbNTN+tIEVa0/fGaQ==""];
```



```
request.uploadProgressBlock = ^(int64_t bytesSent, int64_t totalBytesSent, int64_t
totalBytesExpectedToSend) {
    NSLog(@"%0.1f%%",(float)totalBytesSent*100/(float)totalBytesExpectedToSend);
};
[client putObject:request completionHandler:^(OBSPutObjectResponse *response, NSError *error){
    NSLog(@"%@",response);
}] ;
```

NOTE

- **key**: generated through AES256.
- **keyMD5**: base64-encoded MD5 value of the key.

- SSE-KMS encryption

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Use the SSE-KMS algorithm to upload an object.
OBSPutObjectWithFileRequest *request = [[OBSPutObjectWithFileRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" uploadFilePath:_imagePath];

// SSE-KMS encryption
request.encryption = [[OBSEncryptionTypeKMS alloc]initWithKeyID:nil];

request.uploadProgressBlock = ^(int64_t bytesSent, int64_t totalBytesSent, int64_t
totalBytesExpectedToSend) {
    NSLog(@"%0.1f%%",(float)floor(totalBytesSent*10000/totalBytesExpectedToSend)/100);
};

[client putObject:request completionHandler:^(OBSPutObjectResponse *response, NSError *error){
    NSLog(@"%@",response.etag);
}];
```

Decrypting a Downloaded Object

Sample code:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Decrypt the downloaded object.
NSString * outfilePath = [NSTemporaryDirectory() stringByAppendingString:@"test.png"];
```



```
OBSGetObjectToFileRequest *request = [[OBSGetObjectToFileRequest
alloc]initWithBucketName:@"bbucketname" objectKey:@"objectname" downloadFilePath:outfilePath];

// Enter the key and keyMD5 used for encrypting the object during the object upload.
request.encryption = [[OBSEncryptionTypeCustomer alloc]initWithAlgorithm:@"AES256"
key:@"K7QkYpBkM5+hcs27fsNkUnNVaobncnLht/rCB2o/9Cw=" keyMD5:@"4XvB3tbNTN+tIEVa0/fGaQ=="];
request.downloadProgressBlock = ^(int64_t bytesWritten, int64_t totalBytesWritten, int64_t
totalBytesExpectedToWrite) {
    NSLog(@"%0.1f%%",(float)floor(totalBytesWritten*10000/totalBytesExpectedToWrite)/100);
};
[client getObject:request completionHandler:^(OBSGetObjectResponse *response, NSError *error){
    NSLog(@"%@",response.etag);
}];
```


18 Troubleshooting

18.1 OBS Server-Side Error Codes

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

Error Code	Description	HTTP Status Code
AccessDenied	Access denied.	403 Forbidden
AccessForbidden	Insufficient permission.	403 Forbidden
AccountProblem	Your account encounters a problem that prevents the operation from completing. The account may be expired or frozen.	403 Forbidden
AllAccessDisabled	The user has no permission to perform a specific operation.	403 Forbidden
AmbiguousGrantByEmai-lAddress	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. Specify a different name and retry.	409 Conflict
BucketAlreadyOwned-ByYou	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
CustomDomainAlready-Exist	The configured domain already exists.	400 Bad Request
CustomDomainNotExist	The domain to be operated does not exist.	400 Bad Request
DeregisterUserId	The user has been deregistered.	403 Forbidden
EntityTooSmall	The size of the object to be uploaded is smaller than the lower limit.	400 Bad Request
EntityTooLarge	The size of the object to be uploaded exceeds the upper limit.	400 Bad Request
FrozenUserId	The user has been frozen.	403 Forbidden
IllegalVersioningConfiguration Exception	The **Versioning** configuration specified in the request is invalid.	400 Bad Request
IllegalLocationConstraint-tException	The configured region limitation is inconsistent with the region where it resides.	400 Bad Request
InArrearOrInsufficientBalance	Insufficient permission to perform a specific operation in ACL.	403 Forbidden
IncompleteBody	Incomplete request body.	400 Bad Request

Error Code	Description	HTTP Status Code
IncorrectNumberOfFile-sInPost Request	Each POST request must contain one file to be uploaded.	400 Bad Request
InlineDataTooLarge	The size of inline data exceeds the upper limit.	400 Bad Request
nsufficientStorageSpace	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
InvalidBucketStoragePo-licy	An invalid new policy is specified during bucket policy modification.	400 Bad Request
InvalidDigest	The specified **Content-MD5** is invalid.	400 Bad Request
InvalidEncryptionAlgor-ithmError	Incorrect encryption algorithm.	400 Bad Request
InvalidLocationConstraint	The specified location constraint is invalid.	400 Bad Request
InvalidPart	One or more specified parts cannot be found. The parts may not be uploaded or the specified entity tags (ETags) do not match the parts' ETags.	400 Bad Request
InvalidPartOrder	Parts are not listed in ascending order by part number.	400 Bad Request

Error Code	Description	HTTP Status Code
InvalidPayer	Parts are not listed in ascending order by part number.	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 is invalid.	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	The provided security credentials are invalid.	403 Forbidden
InvalidStorageClass	The specified storage class is invalid.	400 Bad Request
InvalidTargetBucketFor-Logging	The delivery group has no ACL permission for the target bucket.	400 Bad Request
InvalidURI	Cannot resolve the specified uniform resource identifier (URI).	400 Bad Request
KeyTooLong	The provided key is too long.	400 Bad Request
MalformedACLError	The provided XML has bad syntax 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
MalformedPOSTRequest	The body of the POST request is in an incorrect format.	400 Bad Request

Error Code	Description	HTTP Status Code
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
MaxPostPreDataLength-Exceeded Error	The POST request fields prior to the file to be uploaded are too large.	400 Bad Request
MetadataTooLarge	The size of metadata headers exceeds the upper limit.	400 Bad Request
MethodNotAllowed	The specified method is not allowed against the requested resource.	405 Method Not Allowed
MissingContentLength	The HTTP header **Content-Length** is not provided.	411 Length Required
MissingRegion	No region in the request and no default region 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	No header field in the request.	400 Bad Request
MissingSecurityHeader	A required header is not provided.	400 Bad Request
NoSuchBucket	The specified bucket does not exist.	404 Not Found
NoSuchBucketPolicy	No bucket policy exists.	404 Not Found
NoSuchCORSConfiguration	No CORS configuration exists.	Not Found
NoSuchCustomDomain	The requested user domain does not exist.	404 Not Found

Error Code	Description	HTTP Status Code
NoSuchKey	The specified key does not exist.	404 Not Found
NoSuchLifecycleConfiguration	The requested Lifecycle does not exist.	404 Not Found
NoSuchPolicy	The specified policy name does not exist.	404 Not Found
NoSuchUpload	The specified multipart upload does not exist. The upload ID does not exist or the multipart upload job has been aborted or completed.	404 Not Found
NoSuchVersion	The specified version ID does not match any existing version.	404 Not Found
NoSuchWebsiteConfiguration	The requested website does not exist.	404 Not Found
NotImplemented	The provided header implies a function that is unavailable.	501 Not Implemented
NotSignedUp	Your account is not signed up 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 must be addressed using a specified endpoint. Send all future requests to the endpoint.	301 Moved Permanently
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

Error Code	Description	HTTP Status Code
RequestTimeTooSkewed	The socket connection to the server has no reads or writes within the timeout period.	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 and 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
TooManyBuckets	You have attempted to create more buckets than allowed.	400 Bad Request
TooManyObjectCopied	The number of copied users' objects exceeds the upper limit.	400 Bad Request
TooManyWrongSignature	The request is rejected due to high-frequency errors.	400 Bad Request

Error Code	Description	HTTP Status Code
UnexpectedContent	This request does not support content.	400 Bad Request
UnresolvableGrantByEmailAddress	The provided email address does not match any recorded accounts.	400 Bad Request
UserKeyMustBeSpecified	The user's AK is not carried in the request.	400 Bad Request
WebsiteRedirect	The website request lacks bucketName.	301 Moved Permanently
KMS.DisabledException	The master key is disabled in 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 the bucket tag.	400 Bad Request
NoSuchTagSet	The specified bucket is not configured with a tag.	404 Not Found

18.2 SDK Custom Exceptions

You can identify whether a request is successful through **error** in the response. If **error** is null, the request is successful; otherwise, the request fail and the error information will be displayed in the window. Sample code is as follows:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
```



```
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Initiate a request.
OBSListBucketsRequest *request = [OBSListBucketsRequest new];

[client listBuckets:request completionHandler:^(OBSListBucketsResponse *response, NSError *error) {
    // The request is successful and the error information is null.
    if(!error){
        //Process the request.
         NSLog(@"%@",response.headers);
    }
}];
```

18.3 SDK Common Response Headers

After you call an API related to the **OBSClient** class, an instance or a sub-class instance of the **OBSBaseResponse** class will be returned. This class contains information about HTTP/HTTPS response headers.

The following sample code shows how to process the common headers:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// Initiate a request.
OBSListBucketsRequest *request = [OBSListBucketsRequest new];

[client listBuckets:request completionHandler:^(OBSListBucketsResponse *response, NSError *error) {
    NSLog(@"%@",response.headers);
}];
```

18.4 Log Analysis

Enabling System Logging

Sample code:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
```



```
NSString *SK = @"*** Provide your Secret Key ***";
NSString *AK = @"*** Provide your Access Key ***";

// Initialize identity authentication.
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//Initialize service configuration.
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// Initialize an instance of OBSClient.
client = [[OBSClient alloc] initWithConfiguration:conf];

// *****Logging settings*******
// Specify that system logs will be printed.
[client addLogger:[OBSDDASLLogger sharedInstance]];
// Specify that the logs will be displayed in the window.
[client addLogger:[OBSDDTTYLogger sharedInstance]];

// Set log files.
OBSDDFileLogger *fileLogger = [[OBSDDFileLogger alloc] init];
// Set the log retention duration.
fileLogger.rollingFrequency = 60 * 60 * 24; // 24 hour rolling
// Set the maximum number of log files that can be retained.
fileLogger.logFileManager.maximumNumberOfLogFiles = 7;

// Customize log settings.
[client addLogger:fileLogger];

// Set the path for saving log files.
OBSDDLogFileInfo *ts =[fileLogger currentLogFileInfo];
NSLog(@"%@", ts);

// Set the log level.
[client setLogLevel:OBSDDLogLevelDebug];

// Enable logging.
[client setASLogOn];
```

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 **OBSDDLogLevelVerbose** logs and the least information in **OBSDDLogLevelError** logs. The default value is **OBSDDLogLevelInfo**.

Log Level	Description
OBSDDLogLevelOff	Disables logging.
OBSDDLogLevelError	Prints error information only.
OBSDDLogLevelWarning	Prints error and alarm information.
OBSDDLogLevelInfo	Prints error, warning, and basic running information.
OBSDDLogLevelDebug	Prints error, warning, basic running, and debugging information.
OBSDDLogLevelVerbose	Prints all logs.

19 FAQ

19.1 How Can I Perform a Multipart Upload?

For details, see **Performing a Multipart Upload**.

19.2 How Can I Create a Folder?

For details, see **Creating a Folder**.

19.3 How Can I List All Objects in a Bucket?

For details, see **Listing Objects** and **Listing Versioning Objects**.

19.4 How Can I Create a Temporarily Authorized URL?

See **Using a Temporary URL for Authorized Access**.

19.5 How Can I Identify the Endpoint and Location of OBS?

For details, see **Obtaining Endpoints**.

19.6 How Can I Obtain the AK and SK?

For details, see **Setting Up an OBS Environment**.

19.7 How Do I Obtain the Temporary AK/SK?

For details, see **Setting Up an OBS Environment**.

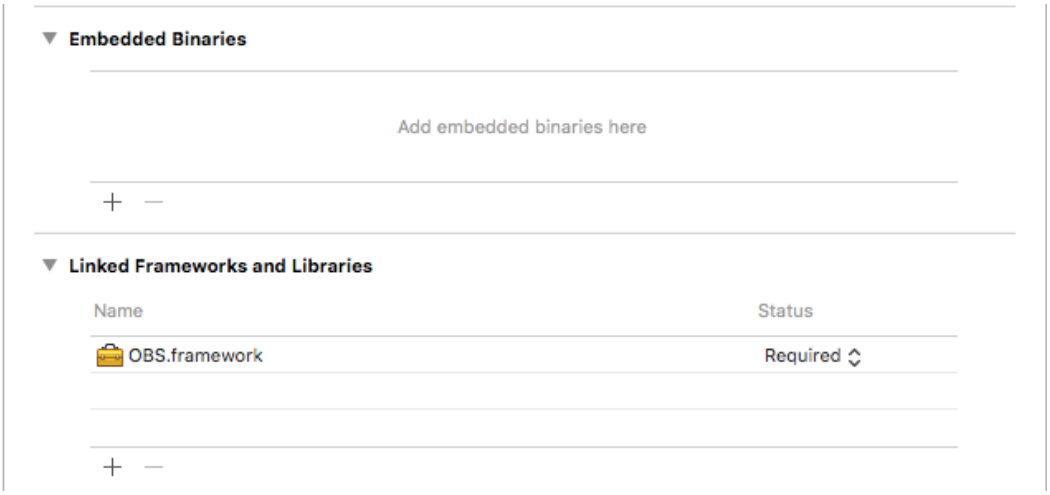
19.8 What Can I Do to Troubleshoot a Project Packing Error?

An error is reported during project packing, stating:

OBS.framework" did not contain a "archived-expanded-entitlements.xcent" resource.

Solution:

Ensure that **Embedded Binaries** does not contain **OBS.framework** that is a static library.



19.9 What Should I Do If the HTTP Status Code 405 Is Reported?

If an API calling fails and the HTTP status code returned is 405, check whether the region supports the called API.

A API Reference

For details about all parameters and definitions of APIs in the OBS iOS SDK, see the **Object Storage Service iOS SDK API Reference**.

B Change History

Release Date	Change History
2020-04-21	This is the ninth official release. Modified the following section: Added the requirements on the number of request connections to **Configuring an Instance of OBSClient**.
2020-03-13	This is the eighth official release. Modified the following sections: Added the sample code for obtaining an object in **Using a Temporary URL for Authorized Access**.
2019-11-20	This is the seventh official release. Reorganized the positions of the following sections: • Example Programs • Technical Support Channels
2019-11-1	This is the sixth official release. Modified the following sections: Added the description of creating an instance of OBSClient using local variables in **General Examples of OBSClient**. Added the description of creating an instance of OBSClient using a user-defined domain name in **Creating an Instance of OBSClient**. Added the function of setting any type of objects to be upload using character strings in **Setting Object Properties**. Added code examples for listing all folders in the root directory in **Listing Objects**. Modified the code example of property rewriting during object copy in **Copying an Object**.

Release Date	Change History
2019-08-20	This is the fifth official release. Modified the following sections: Added the usage description in the concurrent scenario in **Creating an Instance of OBSClient**. Added the error identification method in **Performing a File-Based Upload**. Added the description of setting the file type during upload in **Setting Object Properties**.
2019-05-30	This is the fourth official release. Modified the following sections: In **Setting Up an OBS Environment**, added the content related to the temporary access keys. In **Creating an Instance of OBSClient**, added code examples related to temporary access keys.
2019-03-05	This is the third official release. Added the following section: **API Reference**
2018-10-31	This is the second official release. Added the following sections: **Appendable Upload** Modified the following section: Added the enumeration values of bucket policies in **Managing Bucket ACLs**.
2018-5-31	This is the first official release.

