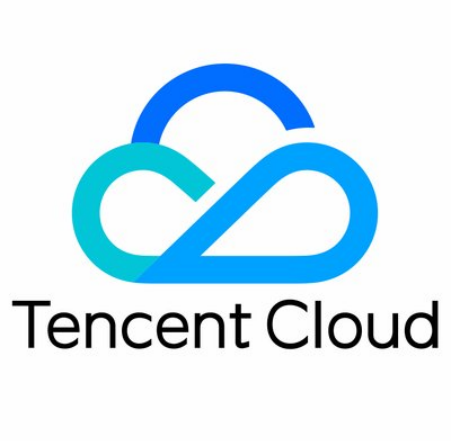
Cloud Object Storage

SDK Documentation

Product Introduction



Contents

SDK Documentation
Android SDK
Getting Started

Last updated：2018-10-16 17:20:31

Preparations for Development

Obtaining SDK

Download the XML Android SDK resources of COS service from XML Android SDK.
Download Demo from: XML Android SDK Demo.

Preparations for development

1. Android 2.2 or above is supported by the SDK.
2. Your mobile phone must be connected to a network such as GPRS, 3G or WiFi;
3. Some features may not function if there is no enough storage on the mobile phone;
4. Obtain APPID, SecretId and SecretKey from COS v5 Console.

> For more information on the definitions of SecretId, SecretKey, Bucket and other terms and how to obtain them, please see COS Glossary.

Configuring SDK

The following JAR files need to be imported into the project and stored in the libs folder:

- cos-android-sdk-V5.4.4.jar
- qcloud-foundation.1.3.0.jar
- okhttp-3.8.1.jar
- okio-1.13.0.jar

Or use Gradle to integrate the SDK into your project as follows:

- compile 'com.tencent.qcloud:cosxml:5.4.4'

The use of the SDK requires the access permissions related to network, storage, etc. You can add the following permission statement in AndroidManifest.xml (For Android 5.0 or above, dynamic permissions are also needed):

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
```

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

Getting Started

Initialization

You need to instantiate CosXmlService and CosXmlServiceConfig before performing any operation.

- CosXmlServiceConfig: configuration parameters;
- CosXmlService: The service class provided by the SDK for operating various COS services;

```
String appid = "COS service APPID";
String region = "The region where the bucket resides";

String secretId = "Cloud API key SecretId";
String secretKey ="Cloud API key SecretKey";
long keyDuration = 600; //Validity of SecretKey (in sec)

//Create a CosXmlServiceConfig object to modify the default configuration parameter as needed
CosXmlServiceConfig serviceConfig = new CosXmlServiceConfig.Builder()
.setAppidAndRegion(appid, region)
.setDebuggable(true)
.builder();

//Create a class for getting signature (see the following example for generating a signature, or refer to the Shor
tTimeCredentialProvider class provided in SDK)
LocalCredentialProvider localCredentialProvider = new LocalCredentialProvider(secretId, secretKey, keyDuration);

//Create a CosXmlService object to implement the COS operations.
Context context = getApplicationContext(); //Context of the application

CosXmlService cosXmlService = new CosXmlService(context,cosXmlServiceConfig, localCredentialProvider);
```

Simple upload of files

```
String bucket = "bucket name"; //bucket format of cos v5: xxx-appid, such as test-1253960454
String cosPath = "[object key](https://cloud.tencent.com/document/product/436/13324), which is the absolute p
ath to the storage on COS"; //for example: cosPath = "test.txt";
String srcPath = "absolute path to the local file"; //for example: srcPath = Environment.getExternalStorageDirecto
ry().getPath() + "/test.txt";
long signDuration = 600; //Validity of the signature (in sec)

PutObjectRequest putObjectRequest = new PutObjectRequest(bucket, cosPath, srcPath);

putObjectRequest.setSign(signDuration,null,null); //If it is not called, sign duration (60s) in the SDK is used by def
ault
```

```
/*Set progress display
Implement the CosXmlProgressListener.onProgress(long progress, long max) method,
progress indicates the uploaded size, and max indicates the total size of the file
*/
putObjectRequest.setProgressListener(new CosXmlProgressListener() {
@Override
public void onProgress(long progress, long max) {
float result = (float) (progress * 100.0/max);
Log.w("TEST","progress =" + (long)result + "%");
}
});

//Upload using synchronization method
try {
PutObjectResult putObjectResult = cosXmlService.putObject(putObjectRequest);

Log.w("TEST","success: " + putObjectResult.accessUrl);

} catch (CosXmlClientException e) {

//Throw an exception
Log.w("TEST","CosXmlClientException =" + e.toString());
} catch (CosXmlServiceException e) {

//Throw an exception
Log.w("TEST","CosXmlServiceException =" + e.toString());
}

//Upload using an asynchronous callback: SDK provides an asynchronous callback method for the COS services
/**

cosXmlService.putObjectAsync(putObjectRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
Log.w("TEST","success =" + result.accessUrl);
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceExcepti
on serviceException) {

String errorMsg = clientException != null ? clientException.toString() : serviceException.toString();
Log.w("TEST",errorMsg);
}
});

*/
```

Multipart upload

Multipart upload generally involves three steps: initializing multipart upload -> performing multipart upload -> completing upload.

```
String bucket = "bucket name"; //bucket format of cos v5: xxx-appid, such as test-1253960454
String cosPath = "[object key](https://cloud.tencent.com/document/product/436/13324), which is the absolute p
ath to the storage on COS";


//First, initialize the multipart upload to obtain the uploadId, which is used for subsequent multipart uploads, co
mpletion of uploads, etc.

String uploadId = null;

InitMultipartUploadRequest initMultipartUploadRequest = new InitMultipartUploadRequest(bucket, cosPath);

initMultipartUploadRequest.setSign(600,null,null);
try {
InitMultipartUploadResult initMultipartUploadResult =
cosXmlService.initMultipartUpload(initMultipartUploadRequest);

//If the initialization is successful, get the uploadId;
Log.w("TEST","success");
uploadId = initMultipartUploadResult.initMultipartUpload.uploadId;

} catch (CosXmlClientException e) {

//Throw an exception
Log.w("TEST","CosXmlClientException =" + e.toString());
} catch (CosXmlServiceException e) {

//Throw an exception
Log.w("TEST","CosXmlServiceException =" + e.toString());
}

//Second, multipart upload requires the parameters uploadId and partNumber; the corresponding eTag number i
s also needed. This example only shows the upload of the first part in a multipart upload.
//Part number: The number of current part among all parts, which starts from 1
//etag: The combination of MD5 returned when the part is uploaded successfully and the part number.

String srcPath = "Absolute path to the local file";
int partNumber = 1; //number of the part to be uploaded, starting from 1; this example shows the upload of the f
irst part

String eTag = null;

UploadPartRequest uploadPartRequest = new UploadPartRequest(bucket, cosPath, partNumber,
srcPath, uploadId);

uploadPartRequest.setSign(600,null,null);
```

```
/*Set progress display
Implement the CosXmlProgressListener.onProgress(long progress, long max) method,
progress indicates the uploaded size, and max indicates the total size of the file
*/
uploadPartRequest.setProgressListener(new CosXmlProgressListener() {
@Override
public void onProgress(long progress, long max) {
float result = (float) (progress * 100.0/max);
Log.w("TEST","progress =" + (long)result + "%");
}
});

try {
UploadPartResult uploadPartResult = cosXmlService.uploadPart(uploadPartRequest);

Log.w("TEST","success");
eTag = uploadPartResult.eTag; //Get the eTag of the part

} catch (CosXmlClientException e) {

//Throw an exception
Log.w("TEST","CosXmlClientException =" + e.toString());
} catch (CosXmlServiceException e) {

//Throw an exception
Log.w("TEST","CosXmlServiceException =" + e.toString());
}


//Third, when all the parts have been uploaded, complete the multipart upload by calling CompleteMultiUpload
Request.
//The parameters uploadId, partNumber, and the eTag value for each part are required

CompleteMultiUploadRequest completeMultiUploadRequest = new CompleteMultiUploadRequest(bucket, cosPa
th, uploadId, null);

completeMultiUploadRequest.setPartNumberAndEtag(partNumber, eTag); //This example only shows the upload
of one part

completeMultiUploadRequest.setSign(600,null,null);
try {
CompleteMultiUploadResult completeMultiUploadResult =
cosXmlService.completeMultiUpload(completeMultiUploadRequest);

Log.w("TEST","success: " + completeMultiUploadResult.accessUrl );

} catch (CosXmlClientException e) {

//Throw an exception
Log.w("TEST","CosXmlClientException =" + e.toString());
} catch (CosXmlServiceException e) {
```

```
//Throw an exception
Log.w("TEST","CosXmlServiceException =" + e.toString());
}
```

UploadService is recommended for multipart upload

```
//UploadService encapsulates the classes for a series of procedures for the above multipart upload request

UploadService.ResumeData resumeData = new UploadService.ResumeData();
resumeData.bucket = "bucket name";
resumeData.cosPath = "[object key](https://cloud.tencent.com/document/product/436/13324),, which is the abso
lute path to the storage on COS"; //for example: cosPath = "test.txt";
resumeData.srcPath = "absolute path to the local file"; //for example: srcPath =Environment.getExternalStorageDi
rectory().getPath() + "/test.txt";
resumeData.sliceSize = 1024 * 1024; //size of each part
resumeData.uploadId = null; //For a resumed upload, uploadId cannot be empty


UploadService uploadService = new UploadService(cosXmlService, resumeData);

/*Set progress display
Implement the CosXmlProgressListener.onProgress(long progress, long max) method,
progress indicates the uploaded size, and max indicates the total size of the file
*/
uploadService.setProgressListener(new CosXmlProgressListener() {
@Override
public void onProgress(long progress, long max) {
float result = (float) (progress * 100.0/max);
Log.w("TEST","progress =" + (long)result + "%");
}
});
try {
CosXmlResult cosXmlResult = uploadService.upload();

Log.w("TEST","success: " + cosXmlResult.accessUrl );

} catch (CosXmlClientException e) {

//Throw an exception
Log.w("TEST","CosXmlClientException =" + e.toString());
} catch (CosXmlServiceException e) {

//Throw an exception
Log.w("TEST","CosXmlServiceException =" + e.toString());
}
```

Downloading files

```
String bucket = "bucket name"; //bucket format of cos v5: xxx-appid, such as test-1253960454
String cosPath = "[object key](https://cloud.tencent.com/document/product/436/13324), which is the absolute p
ath to the storage on COS";
String savePath = "path to the file downloaded to the local machine";

GetObjectRequest getObjectRequest = GetObjectRequest(bucket, cosPath, savePath);
getObjectRequest.setSign(signDuration,null,null);

/*Set progress display
Implement the CosXmlProgressListener.onProgress(long progress, long max) method,
progress indicates the uploaded size, and max indicates the total size of the file
*/
getObjectRequest.setProgressListener(new CosXmlProgressListener() {
@Override
public void onProgress(long progress, long max) {
float result = (float) (progress * 100.0/max);
Log.w("TEST","progress =" + (long)result + "%");
}
});

//Upload using synchronization method
try {
GetObjectResult getObjectResult =cosXmlService.getObject(getObjectRequest);
Log.w("TEST","success: " + getObjectResult.xCOSStorageClass);
} catch (CosXmlClientException e) {
Log.w("TEST","CosXmlClientException =" + e.toString());
} catch (CosXmlServiceException e) {
Log.w("TEST","CosXmlServiceException =" + e.toString());
}

//**Use an asynchronous callback request**
/**

cosXmlService.getObjectAsync(getObjectRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult cosXmlResult) {
Log.w("TEST","success");
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceExcepti
on serviceException) {
String errorMsg = clientException != null ? clientException.toString() : serviceException.toString();
Log.w("TEST",errorMsg);
}
});

*/
```

Generating Signature

For more information on how to generate a signature, please see Request Signature.
The SDK you are using has provided the class for getting signature. You simply need to inherit the BasicLifecycleCredentialProvider class and override the fetchNewCredentials() method to obtain the SecretId, SecretKey, and SecretKey Duration. To send the request using the temporary key, you need to obtain the tempSecretKey, tempSecrekId, sessionToken and expiredTime. For more information on how to obtain a temporary key via CAM, please see Quick Setup of Mobile Application Transfer Service.

Example

```
/**
Method 1: Signing using a permanent key
*/
public class LocalCredentialProvider extends BasicLifecycleCredentialProvider{
private String secretKey;
private long keyDuration;
private String secretId;

public LocalCredentialProvider(String secretId, String secretKey, long keyDuration) {
this.secretId = secretId;
this.secretKey = secretKey;
this.keyDuration = keyDuration;
}

/**
BasicQCloudCredentials is returned
*/
@Override
public QCloudLifecycleCredentials fetchNewCredentials() throws CosXmlClientException {
long current = System.currentTimeMillis() / 1000L;
long expired = current + duration;
String keyTime = current+";"+expired;
return new BasicQCloudCredentials(secretId, secretKeyToSignKey(secretKey, keyTime), keyTime);
}

private String secretKeyToSignKey(String secretKey, String keyTime) {
String signKey = null;
try {
if (secretKey == null) {
throw new IllegalArgumentException("secretKey is null");
}
if (keyTime == null) {
throw new IllegalArgumentException("qKeyTime is null");
}
} catch (IllegalArgumentException e) {
e.printStackTrace();
}
try {
```

```
byte[] byteKey = secretKey.getBytes("utf-8");
SecretKey hmacKey = new SecretKeySpec(byteKey, "HmacSHA1");
Mac mac = Mac.getInstance("HmacSHA1");
mac.init(hmacKey);
signKey = StringUtils.toHexString(mac.doFinal(keyTime.getBytes("utf-8")));
} catch (UnsupportedEncodingException e) {
e.printStackTrace();
} catch (NoSuchAlgorithmException e) {
e.printStackTrace();
} catch (InvalidKeyException e) {
e.printStackTrace();
}
return signKey;
}
}

/**
Method 2: Signing using a temporary key (recommended). This assumes that you have obtained a temporary key
tempSecretKey, tempSecrekId,
sessionToken and expiredTime.
*/
public class LocalSessionCredentialProvider extends BasicLifecycleCredentialProvider{
private String tempSecretId;
private String tempSecretKey;
private String sessionToken;
private long expiredTime;

public LocalCredentialProvider(String tempSecretId, String tempSecretKey, String sessionToken, long expiredTim
e) {
this.tempSecretId = tempSecretId;
this.tempSecretKey = tempSecretKey;
this.sessionToken = sessionToken;
this.expiredTime = keyDuration;
}

/**
SessionQCloudCredential is returned
*/
@Override
public QCloudLifecycleCredentials fetchNewCredentials() throws CosXmlClientException {
return new SessionQCloudCredentials(tmpSecretId, tmpSecretKey, sessionToken, expiredTime);
}
}
```

API Documentation

Last updated：2018-09-30 10:53:16

Description on SDK Exceptions

In the SDK, if an API call fails to operate on a COS object, a CosXmlClientException or CosXmlServiceException will be thrown.

CosXmlClientException

Client exceptions refer to server interaction failures caused by unexpected client issues such as failure to connect to the server, failure to parse the data returned by the server, and the occurrence of I/O exception when reading a local file. Inherited from Exception, CosXmlClientException has no custom member variables, and is used in the same way as Exception.

CosXmlServiceException

CosXmlServiceException refers to scenarios in which interaction is completed but the operation failed. For example, the client accesses a bucket that does not exist, delete a file that does not exist, or does not have the permission to perform an operation, or the server failed. CosXmlServiceException contains the status code, requestid, and error details returned by the server. After an exception is captured, it is recommended to print the entire exception. The exception contains the necessary troubleshooting factors. Member variables of exception are described as follows:

request Member	Description	Type
requestId	Request ID to specify a request. It is very important for troubleshooting.	String
statusCode	Status code of the response. 4xx represents the request failure caused by the client, and 5xx represents the failure caused by the server exception. Please see COS Error Messages	String
errorCode	Error Code returned in the body when request fails. Please see COS Error Messages	String
errorMessage	Error Message returned in the body when request fails. Please see COS Error Messages	String

Initialization

You need to instantiate CosXmlService and CosXmlServiceConfig before performing operations.

For more information on the definitions of SecretId, SecretKey, Bucket and other terms and how to obtain them, please see COS Glossary.

Instantiating CosXmlServiceConfig

Call CosXmlServiceConfig.Builder().builder() to instantiate the CosXmlServiceConfig object.

Parameters

Parameter Name	Description	Type	Required
appid	COS service APPID String	Yes	
region	The region where the bucket resides String	Yes	

Other configuration methods

Method	Description
setAppidAndRegion(String, String)	Sets the region to which the "appid" and the "bucket" belongs
isHttps(boolean)	true: https request; false: http request (default)
setDebuggable(boolean)	debug log mode on or off

Example

```
String appid = "COS service APPID";
String region = "The region where the bucket resides"; //Region: You can view the created bucket via COS consol
e
CosXmlServiceConfig serviceConfig = new CosXmlServiceConfig.Builder()
.isHttps(true)
.setAppidAndRegion(appid, region)
.setDebuggable(true)
.builder();
```

Instantiating CosXmlService

Call the `CosXmlService(Context context, CosXmlServiceConfig serviceConfig, QCloudCredentialProvider cloudCredentialProvider)` construction method to instantiate the CosXmlService object.

Parameters

Parameter Name	Description	Type	Required
context	Context of application	Context	Yes
serviceConfig	Class for configuring SDK	CosXmlServiceConfig	Yes
basicLifecycleCredentialProvider	Class for getting signature of service request	BasicLifecycleCredentialProvider	Yes

Example

```
String appid = "COS service APPID";
String region = "The region where the bucket is located";

//Create a CosXmlServiceConfig object to modify the default configuration parameter as needed
CosXmlServiceConfig serviceConfig = new CosXmlServiceConfig.Builder()
.isHttps(true)
.setAppidAndRegion(appid, region)
.setDebuggable(true)
.builder();

/**
*
* Create an object of the class for getting signature ShortTimeCredentialProvider to calculate the signature when using COS.
* Implement your own signature method (extends BasicLifecycleCredentialProvider and the * * fetchNewCredentials() method) by applying the signature format provided in SDK.
* The default signature algorithm provided in the SDK is used here.
*
*/
String secretId = "Cloud API key secretId";
String secretKey ="Cloud API key secretKey";
long keyDuration = 600; //Validity of SecretKey (in sec)
ShortTimeCredentialProvider localCredentialProvider = new ShortTimeCredentialProvider(secretId, secretKey, keyDuration);

//Create a CosXmlService object to implement the COS operations.
Context context = getApplicationContext(); //Context of the application
CosXmlService cosXmlService = new CosXmlService(context,cosXmlServiceConfig, localCredentialProvider);
```

Generating Signature

For more information on how to generate and use a signature, please see Request Signature. The SDK has provided the class for getting signature. You simply need to inherit the BasicLifecycleCredentialProvider class and override the fetchNewCredentials() method. For more information on how to obtain a temporary key, please see Quick Setup of Mobile Application Transfer Service.

Example

```
/**
Method 1: Signing using a permanent key
*/
public class LocalCredentialProvider extends BasicLifecycleCredentialProvider{
private String secretKey;
private long keyDuration;
private String secretId;

public LocalCredentialProvider(String secretId, String secretKey, long keyDuration) {
```

```
this.secretId = secretId;
this.secretKey = secretKey;
this.keyDuration = keyDuration;
}

/**
BasicQCloudCredentials is returned
*/
@Override
public QCloudLifecycleCredentials fetchNewCredentials() throws CosXmlClientException {
long current = System.currentTimeMillis() / 1000L;
long expired = current + duration;
String keyTime = current+";"+expired;
return new BasicQCloudCredentials(secretId, secretKeyToSignKey(secretKey, keyTime), keyTime);
}

private String secretKeyToSignKey(String secretKey, String keyTime) {
String signKey = null;
try {
if (secretKey == null) {
throw new IllegalArgumentException("secretKey is null");
}
if (keyTime == null) {
throw new IllegalArgumentException("qKeyTime is null");
}
} catch (IllegalArgumentException e) {
e.printStackTrace();
}
try {
byte[] byteKey = secretKey.getBytes("utf-8");
SecretKey hmacKey = new SecretKeySpec(byteKey, "HmacSHA1");
Mac mac = Mac.getInstance("HmacSHA1");
mac.init(hmacKey);
signKey = StringUtils.toHexString(mac.doFinal(keyTime.getBytes("utf-8")));
} catch (UnsupportedEncodingException e) {
e.printStackTrace();
} catch (NoSuchAlgorithmException e) {
e.printStackTrace();
} catch (InvalidKeyException e) {
e.printStackTrace();
}
return signKey;
}
}

/**
Method 2: Signing using a temporary key (recommended). This assumes that you have obtained a temporary key
tempSecretKey, tempSecrekId,
sessionToken and expiredTime.
*/
public class LocalSessionCredentialProvider extends BasicLifecycleCredentialProvider{
```

```
private String tempSecretId;
private String tempSecretKey;
private String sessionToken;
private long expiredTime;

public LocalSessionCredentialProvider(String tempSecretId, String tempSecretKey, String sessionToken, long exp
iredTime) {
this.tempSecretId = tempSecretId;
this.tempSecretKey = tempSecretKey;
this.sessionToken = sessionToken;
this.expiredTime = keyDuration;
}

/**
SessionQCloudCredential is returned
*/
@Override
public QCloudLifecycleCredentials fetchNewCredentials() throws CosXmlClientException {
return new SessionQCloudCredentials(tmpSecretId, tmpSecretKey, sessionToken, expiredTime);
}
}
```

Simple upload of files

This API can be used to upload local files to the specified Bucket. The steps are as follows:

1. Call the `PutObjectRequest(String, String, String)` construction method to instantiate the PutObjectRequest object.
2. Call putObject method of CosXmlService, input PutObjectRequest, and get the returned PutObjectResult object. (Alternatively, call the putObjectAsync method, and input PutObjectRequest and CosXmlResultListener for asynchronous callback).

Parameters

Parameter Name	Description	Type	Required
bucket	bucket name (bucket format of cos v5: xxx-appid, such as test-1253960454)	String	Yes
cosPath	Object key, which is the absolute path to the storage on COS	String	Yes
srcPath	Absolute path to the local file	String	Yes
signDuration	Validity of the signature (in sec)	Long	Yes
checkHeaderListForSign	Request header in signature for verification	Set<String>	No

Parameter Name	Description	Type	Required
checkParameterListForSing	Request parameters in signature for verification	Set<String>	No
qCloudProgressListener	Callback for upload progress	CosXmlProgressListener	No
cosXmlResultListener	Callback for upload result	CosXmlResultListener	No

Returned result

Request result is returned through member variables of PutObjectResult object.

Member Variable Name	Description	Type
accessUrl	Return URL for accessing file when the request is successful	String
httpCode	Request is successful when it is in [200, 300), otherwise request failed	Int

If an exception CosClientException or CosServiceException is thrown, please see **Description on SDK Exceptions** at the beginning.

Example

```
String bucket = "bucket";
String cosPath = "cosPath";
String srcPath = "Absolute path to the local file";

PutObjectRequest putObjectRequest = new PutObjectRequest(bucket, cosPath, srcPath);
putObjectRequest.setSign(signDuration,null,null);

putObjectRequest.setProgressListener(new CosXmlProgressListener() {
@Override
public void onProgress(long progress, long max) {
float result = (float) (progress * 100.0/max);
Log.w("TEST","progress =" + (long)result + "%");
}
});

//Upload using synchronization method
try {
PutObjectResult putObjectResult = cosXmlService.putObject(putObjectRequest);

Log.w("TEST","success: " + putObjectResult.accessUrl);

} catch (CosXmlClientException e) {
```

```
//Throw an exception
Log.w("TEST","CosXmlClientException =" + e.toString());
} catch (CosXmlServiceException e) {

//Throw an exception
Log.w("TEST","CosXmlServiceException =" + e.toString());
}

//**Upload using asynchronous callback**
/**

cosXmlService.putObjectAsync(putObjectRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult cosXmlResult) {

Log.w("TEST","success");
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceExcepti
on

serviceException) {

String errorMsg = clientException != null ? clientException.toString() : serviceException.toString();
Log.w("TEST",errorMsg);
}
});
*/
```

Multipart Upload (UploadServer is recommended)

Initializing multipart upload

This API is used to initialize multipart upload. After the execution of this request, UploadId will be returned for the subsequent Upload Part requests. The steps are as follows:

1. Call the `InitMultipartUploadRequest(String, String)` construction method to instantiate the InitMultipartUploadRequest object.
2. Call the initMultipartUpload method of CosXmlService, input InitMultipartUploadRequest, and get the returned InitMultipartUploadResult object.
   (Alternatively, call the initMultipartUploadAsync method, and input InitMultipartUploadRequest and CosXmlResultListener for asynchronous callback).

Parameters

Parameter Name	Description	Type	Required

Parameter Name	Description	Type	Required
bucket	Bucket name (bucket format of cos v5: xxx-appid, such as test-1253960454)	String	Yes
cosPath	Object key, which is the absolute path to the storage on COS	String	Yes
signDuration	Validity of the signature (in sec)	Long	Yes
checkHeaderListForSign	Request header in signature for verification	Set<String>	No
checkParameterListForSing	Request parameters in signature for verification	Set<String>	No
cosXmlResultListener	Callback for upload result	CosXmlResultListener	No

Returned result

Request result is returned through member variables of InitMultipartUploadResult object.

Member Variable Name	Description	Type
initMultipartUpload	Result returned by successful request	InitMultipartUpload
httpCode	Request is successful when it is in [200, 300), otherwise request failed	Int

If an exception CosClientException or CosServiceException is thrown, please see **Description on SDK Exceptions** at the beginning.

Example

```
String bucket = "bucket";
String cosPath = "cosPath";

InitMultipartUploadRequest initMultipartUploadRequest = new InitMultipartUploadRequest(bucket, cosPath);
initMultipartUploadRequest.setSign(signDuration,null,null);

String uploadId = null;

//Request using synchronization method
try {
InitMultipartUploadResult initMultipartUploadResult = cosXmlService.initMultipartUpload(initMultipartUploadRequest);

Log.w("TEST","success");
```

```
uploadId =initMultipartUploadResult.initMultipartUpload.uploadId;

} catch (CosXmlClientException e) {

Log.w("TEST","CosXmlClientException =" + e.toString());

} catch (CosXmlServiceException e) {

Log.w("TEST","CosXmlServiceException =" + e.toString());
}

//**Use an asynchronous callback request**
/**

cosXmlService.initMultipartUploadAsync(initMultipartUploadRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult cosXmlResult) {

Log.w("TEST","success");
uploadId = ((InitMultipartUploadResult)cosXmlResult).initMultipartUpload.uploadId;
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceExcepti
on

serviceException) {

String errorMsg = clientException != null ? clientException.toString() : serviceException.toString();
Log.w("TEST",errorMsg);
}
});

*/
```

Multipart upload

This API is used to implement multipart upload. The allowed number of parts is limited to 10,000, and the size of part should be between 1 MB and 5 GB. The steps are as follows:

1. Call the `UploadPartRequest(String, String, int, String, String)` construction method to instantiate the UploadPartRequest object.
2. Call the uploadPart method of CosXmlService, Iinput UploadPartRequest, and get the returned UploadPartResult object.
   (Alternatively, call the uploadPartAsync method, and input UploadPartRequest and CosXmlResultListener for asynchronous callback).

Parameters

Parameter Name	Description	Type	Required
bucket	Bucket name (bucket format of cos v5: xxx-appid, such as test-1253960454)	String	Yes
cosPath	Object key, which is the absolute path to the storage on COS	String	Yes
uploadId	uploadId returned when multipart upload is initialized	String	Yes
partNumber	Part No., which starts from 1	Int	Yes
srcPath	Absolute path to the local file	String	Yes
fileOffset	Where the part starts in the file	Long	No
contentLength	Size of the part	Long	No
signDuration	Validity of the signature (in sec)	Long	No
checkHeaderListForSign	Request header in signature for verification	Set<String>	No
checkParameterListForSing	Request parameters in signature for verification	Set<String>	No
qCloudProgressListener	Callback for upload progress	CosXmlProgressListener	No
cosXmlResultListener	Callback for upload result	CosXmlResultListener	No

Returned result

Request result is returned through member variables of UploadPartResult object.

Member Variable Name	Type	Description
eTag	Return the MD5 value of the part if the request is successful. It is used to complete the sharding.	String
httpCode	Request is successful when it is in [200, 300), otherwise request failed	Int

If an exception CosClientException or CosServiceException is thrown, please see **Description on SDK Exceptions** at the beginning.

Example

```
String bucket = "bucket";
```

```
String cosPath = "cosPath";
String uploadId = " uploadId returned when multipart upload is initialized | ";
int partNumber = 1;// The number of the part to be uploaded, starting from 1
String srcPath = "Absolute path to the local file";

UploadPartRequest uploadPartRequest = new UploadPartRequest(bucket, cosPath, partNumber, srcPath, uploadI
d);
uploadPartRequest.setSign(signDuration,null,null);

uploadPartRequest.setProgressListener(new CosXmlProgressListener() {
@Override
public void onProgress(long progress, long max) {
float result = (float) (progress * 100.0/max);
Log.w("TEST","progress =" + (long)result + "%");
}
});

String eTag = null;

//Upload using synchronization method
try {
UploadPartResult uploadPartResult = cosXmlService.uploadPart(uploadPartRequest);

Log.w("TEST","success");
eTag = uploadPartResult.eTag; // Get the eTag of the part

} catch (CosXmlClientException e) {

Log.w("TEST","CosXmlClientException =" + e.toString());

} catch (CosXmlServiceException e) {

Log.w("TEST","CosXmlServiceException =" + e.toString());
}
//**Use an asynchronous callback request**
/**

cosXmlService.uploadPartAsync(uploadPartRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult cosXmlResult) {

Log.w("TEST","success");
eTag =((UploadPartResult)cosXmlResult).eTag;
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceExcepti
on

serviceException) {
```

```
String errorMsg = clientException != null ? clientException.toString() : serviceException.toString();
Log.w("TEST",errorMsg);
}
});

*/
```

Completing multipart upload

After all parts are uploaded, this API must be called to complete the entire multipart upload. The steps are as follows:

1. Call the `CompleteMultiUploadRequest(String, String, String, Map<Integer, String>)` construction method to instantiate the CompleteMultiUploadRequest object.
2. Call the completeMultiUpload method of CosXmlService, input CompleteMultiUploadRequest, and get the returned CompleteMultiUploadResult object.
   (Alternatively, call the completeMultiUploadAsync method, and input CompleteMultiUploadRequest and CosXmlResultListener for asynchronous callback).

Parameters

Parameter Name	Description	Type	Required
bucket	Bucket name (bucket format of cos v5: xxx-appid, such as test-1253960454)	String	Yes
cosPath	Object key, which is the absolute path to the storage on COS	String	Yes
uploadId	uploadId returned when multipart upload is initialized	String	Yes
partNumberAndETag	Part No. and the corresponding MD5 value	Map<Integer,String>	Yes
signDuration	Validity of the signature (in sec)	Long	Yes
checkHeaderListForSign	Request header in signature for verification	Set<String>	No
checkParameterListForSing	Request parameters in signature for verification	Set<String>	No
cosXmlResultListener	Callback for upload result	CosXmlResultListener	No

Returned result

Request result is returned through member variables of CompleteMultiUploadResult object.

Member Variable Name	Description	Type
completeMultipartUpload	Result returned by successful request	CompleteMultipartResult

Member Variable Name	Description	Type
accessUrl	Return URL for accessing file when the request is successful	String

If an exception CosClientException or CosServiceException is thrown, please see **Description on SDK Exceptions** at the beginning.

Example

```
String bucket = "bucket";
String cosPath = "cosPath";
String uploadId = " uploadId returned when multipart upload is initialized | ";
int partNumber = 1;
String etag = "The etag returned after the part with the No. of partNumber is uploaded";
Map<Integer, String> partNumberAndETag = new HashMap<>();
partNumberAndETag.put(partNumber, etag);

CompleteMultiUploadRequest completeMultiUploadRequest = new CompleteMultiUploadRequest(bucket, cosPa
th, uploudId,

partNumberAndETag);
completeMultiUploadRequest.setSign(signDuration,null,null);

//Request using synchronization method
try {
CompleteMultiUploadResult completeMultiUploadResult = cosXmlService.completeMultiUpload(completeMultiU
ploadRequest);

Log.w("TEST","success: "+ completeMultiUploadResult.completeMultipartUpload.toString()());
} catch (CosXmlClientException e) {

Log.w("TEST","CosXmlClientException =" + e.toString());

} catch (CosXmlServiceException e) {

Log.w("TEST","CosXmlServiceException =" + e.toString());
}

//**Use an asynchronous callback request**
/**

cosXmlService.completeMultiUploadAsync(completeMultiUploadRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult cosXmlResult) {
```

```
Log.w("TEST","success");
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceExcepti
on

serviceException) {

String errorMsg = clientException != null ? clientException.toString() : serviceException.toString();
Log.w("TEST",errorMsg);
}
});

*/
```

Listing uploaded parts

This API is used to query the uploaded parts when uploading particular parts, which lists all the uploaded parts under a specified UploadId.

1. Call the `ListPartsRequest(String, String, String)` construction method to instantiate the ListPartsRequest object.
2. Call the listParts method of CosXmlService, input ListPartsRequest, and get the returned ListPartsResult object. (Alternatively, call the listPartsAsync method, and input ListPartsRequest and CosXmlResultListener for asynchronous callback).

Parameters

Parameter Name	Description	Type	Required
bucket	Bucket name (bucket format of cos v5: xxx-appid, such as test-1253960454)	String	Yes
cosPath	Object key, which is the absolute path to the storage on COS	String	Yes
uploadId	uploadId returned when multipart upload is initialized	String	Yes
signDuration	Validity of the signature (in sec)	Long	Yes
checkHeaderListForSign	Request header in signature for verification	Set<String>	No
checkParameterListForSing	Request parameters in signature for verification	Set<String>	No
cosXmlResultListener	Callback for upload result	CosXmlResultListener	No

Returned result

Request result is returned through member variables of ListPartsResult object.

Member Variable Name	Description	Type
listParts	Result returned by successful request	ListParts
httpCode	Request is successful when it is in [200, 300), otherwise request failed	Int

If an exception CosClientException or CosServiceException is thrown, please see **Description on SDK Exceptions** at the beginning.

Example

```
String bucket = "bucket";
String cosPath = "cosPath";
String uploadId = " uploadId returned when multipart upload is initialized | ";

ListPartsRequest listPartsRequest = new ListPartsRequest(bucket, cosPath, uploadId);
listPartsRequest.setSign(signDuration,null,null);

//Request using synchronization method
try {
ListPartsResult listPartsResult = cosXmlService.listParts(listPartsRequest);

Log.w("TEST","success: " + listPartsResult.listParts.toString());

} catch (CosXmlClientException e) {

Log.w("TEST","CosXmlClientException =" + e.toString());

} catch (CosXmlServiceException e) {

Log.w("TEST","CosXmlServiceException =" + e.toString());
}

//**Use an asynchronous callback request**
/**

cosXmlService.listPartsAsync(listPartsRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult cosXmlResult) {

Log.w("TEST","success");
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceExcepti
on
```

```
serviceException) {

String errorMsg = clientException != null ? clientException.toString() : serviceException.toString();
Log.w("TEST",errorMsg);
}
});

*/
```

Aborting and deleting uploaded parts

This API is used to abort a multipart upload operation and delete parts that are already uploaded.

1. Call the `AbortMultiUploadRequest(String, String, String)` construction method to instantiate the AbortMultiUploadRequest object.
2. Call the abortMultiUpload method of CosXmlService, input AbortMultiUploadRequest, and get the returned AbortMultiUploadResult object.
   (Alternatively, call the abortMultiUploadAsync method, and input AbortMultiUploadRequest and CosXmlResultListener for asynchronous callback).

Parameters

Parameter Name	Description	Type	Required
bucket	Bucket name (bucket format of cos v5: xxx-appid, such as test-1253960454)	String	Yes
cosPath	Object key, which is the absolute path to the storage on COS	String	Yes
uploadId	uploadId returned when multipart upload is initialized	String	Yes
signDuration	Validity of the signature (in sec)	Long	Yes
checkHeaderListForSign	Request header in signature for verification	Set<String>	No
checkParameterListForSing	Request parameters in signature for verification	Set<String>	No
cosXmlResultListener	Callback for upload result	CosXmlResultListener	No

Returned result

Request result is returned through member variables of AbortMultiUploadResult object.

Member Variable Name	Description	Type
httpCode	Request is successful when it is in [200, 300), otherwise request failed	Int

If an exception CosClientException or CosServiceException is thrown, please see **Description on SDK Exceptions** at the beginning.

Example

```
String bucket = "bucket";
String cosPath = "cosPath";
String uploadId = " uploadId returned when multipart upload is initialized | ";

AbortMultiUploadRequest abortMultiUploadRequest = new AbortMultiUploadRequest(bucket, cosPath, uploadId);
abortMultiUploadRequest.setSign(signDuration,null,null);

//Request using synchronization method
try {
AbortMultiUploadResult abortMultiUploadResult = cosXmlService.abortMultiUpload(abortMultiUploadRequest);
Log.w("TEST", "success");

} catch (CosXmlClientException e) {

Log.w("TEST","CosXmlClientException =" + e.toString());

} catch (CosXmlServiceException e) {

Log.w("TEST","CosXmlServiceException =" + e.toString());
}
//**Use an asynchronous callback request**
/**

cosXmlService.abortMultiUploadAsync(abortMultiUploadRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult cosXmlResult) {

Log.w("TEST","success");
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceExcepti
on

serviceException) {

String errorMsg = clientException != null ? clientException.toString() : serviceException.toString();
Log.w("TEST",errorMsg);
}
});

*/
```

Deleting Files

Deleting a single file

This API is used to delete a file in the specified bucket. The steps are as follows:

1. Call the `DeleteObjectRequest(String, String)` construction method to instantiate the DeleteObjectRequest object.
2. Call the completeMultiUpload method of CosXmlService, input DeleteObjectRequest, and get the returned DeleteObjectResult object.
   (Alternatively, call the deleteObjectAsync method, and input DeleteObjectRequest and CosXmlResultListener for asynchronous callback).

Parameters

Parameter Name	Description	Type	Required
bucket	Bucket name (bucket format of cos v5: xxx-appid, such as test-1253960454)	String	Yes
cosPath	Object key, which is the absolute path to the storage on COS	String	Yes
signDuration	Validity of the signature (in sec)	Long	Yes
checkHeaderListForSign	Request header in signature for verification	Set<String>	No
checkParameterListForSing	Request parameters in signature for verification	Set<String>	No
cosXmlResultListener	Callback for upload result	CosXmlResultListener	No

Returned result

Request result is returned through member variables of DeleteObjectResult object.

Member Variable Name	Description	Type
httpCode	Request is successful when it is in [200, 300), otherwise request failed	Int

If an exception CosClientException or CosServiceException is thrown, please see **Description on SDK Exceptions** at the beginning.

Example

```
String bucket = "bucket";
```

```
String cosPath = "cosPath";

DeleteObjectRequest deleteObjectRequest = new DeleteObjectRequest(bucket, cosPath);
deleteObjectRequest.setSign(signDuration,null,null);

//Delete using synchronous method
try {
DeleteObjectResult deleteObjectResult = cosXmlService.deleteObject(deleteObjectRequest);
Log.w("TEST","success ");

} catch (CosXmlClientException e) {
Log.w("TEST","CosXmlClientException =" + e.toString());
} catch (CosXmlServiceException e) {
Log.w("TEST","CosXmlServiceException =" + e.toString());
}

//**Use an asynchronous callback request**
/**

cosXmlService.deleteObjectAsync(deleteObjectRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult cosXmlResult) {

Log.w("TEST","success");
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceExcepti
on

serviceException) {

String errorMsg = clientException != null ? clientException.toString() : serviceException.toString();
Log.w("TEST",errorMsg);
}
});

*/
```

Deleting multiple files

This API is used to delete files in a specified bucket in batches. A maximum of 1,000 files can be deleted for a single request. The steps are as follows:

1. Call the DeleteMultiObjectRequest(String, List<String>) construction method to instantiate the DeleteMultiObjectRequest object.
2. Call the deleteMultiObject method of CosXmlService, input DeleteMultiObjectRequest, and get the returned DeleteMultiObjectResult object.

(Alternatively, call the deleteMultiObjectAsync method, and input DeleteMultiObjectRequest and CosXmlResultListener for asynchronous callback).

Parameters

Parameter Name	Description	Type	Required
bucket	Bucket name (bucket format of cos v5: xxx-appid, such as test-1253960454)	String	Yes
quiet	true: Only information of the file that failed to be deleted is returned; false: Deletion result of each file is returned	Boolean	Yes
objectList	The list of object keysto delete	List<String>	Yes
signDuration	Validity of the signature (in sec)	Long	Yes
checkHeaderListForSign	Request header in signature for verification	Set<String>	No
checkParameterListForSing	Request parameters in signature for verification	Set<String>	No
cosXmlResultListener	Callback for upload result	CosXmlResultListener	No

Returned result

Request result is returned through member variables of DeleteMultiObjectResult object.

Member Variable Name	Description	Type
httpCode	Request is successful when it is in [200, 300), otherwise request failed	Int

If an exception CosClientException or CosServiceException is thrown, please see **Description on SDK Exceptions** at the beginning.

Example

```
String bucket = "bucket";
List<String> objectList = new ArrayList<String>();
objectList.add("/2/test.txt");

DeleteMultiObjectRequest deleteMultiObjectRequest = new DeleteMultiObjectRequest();
deleteMultiObjectRequest.setQuiet(quiet);
deleteMultiObjectRequest.setSign(signDuration,null,null);

//Delete using synchronous method
try {
```

```
DeleteMultiObjectResult deleteMultiObjectResult =cosXmlService.deleteMultiObject(deleteMultiObjectRequest);

Log.w("TEST","success: " + deleteMultiObjectResult.deleteResult.toString());
} catch (CosXmlClientException e) {

Log.w("TEST","CosXmlClientException =" + e.toString());

} catch (CosXmlServiceException e) {

Log.w("TEST","CosXmlServiceException =" + e.toString());
}

//**Use an asynchronous callback request**
/**

cosXmlService.deleteMultiObjectAsync(deleteMultiObjectRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult cosXmlResult) {

Log.w("TEST","success");
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceExcepti
on

serviceException) {

String errorMsg = clientException != null ? clientException.toString() : serviceException.toString();
Log.w("TEST",errorMsg);
}
});

*/
```

Downloading a File

This API is used to download a file in the specified bucket locally. The steps are as follows:

1. Call the `GetObjectRequest(String, String, String)` construction method to instantiate the GetObjectRequest object.
2. Call the getObject method of CosXmlService, input GetObjectRequest, and get the returned GetObjectResult object. (Alternatively, call the getObjectAsync method, and input GetObjectRequest and CosXmlResultListener for asynchronous callback).

Parameters

Parameter Name	Description	Type	Required

Parameter Name	Description	Type	Required
bucket	Bucket name (bucket format of cos v5: xxx-appid, such as test-1253960454)	String	Yes
cosPath	Object key, which is the absolute path to the storage on COS	String	Yes
savaPath	Absolute path to the file downloaded locally	String	Yes
start	Where the requested file starts	Long	No
end	Where the requested file ends	Long	No
signDuration	Validity of the signature (in sec)	Long	Yes
checkHeaderListForSign	Request header in signature for verification	Set<String>	No
checkParameterListForSing	Request parameters in signature for verification	Set<String>	No
qCloudProgressListener	Callback for download progress	CosXmlProgressListener	No
cosXmlResultListener	Callback for upload result	CosXmlResultListener	No

Returned result

Request result is returned through member variables of GetObjectResult object.

Member Variable Name	Description	Type
httpCode	Request is successful when it is in [200, 300), otherwise request failed	Int

If an exception CosClientException or CosServiceException is thrown, please see **Description on SDK Exceptions** at the beginning.

Example

```
String bucket = "bucket";
String cosPath = "cosPath";
String savePath = "savePath";

GetObjectRequest getObjectRequest = GetObjectRequest(bucket, cosPath, savePath);

getObjectRequest.setSign(signDuration,null,null);
getObjectRequest.setProgressListener(new CosXmlProgressListener() {
```

```
@Override
public void onProgress(long progress, long max) {
float result = (float) (progress * 100.0/max);
Log.w("TEST","progress =" + (long)result + "%");
}
});

//Upload using synchronization method
try {
GetObjectResult getObjectResult =cosXmlService.getObject(getObjectRequest);

Log.w("TEST","success: " + getObjectResult.xCOSStorageClass);

} catch (CosXmlClientException e) {

Log.w("TEST","CosXmlClientException =" + e.toString());

} catch (CosXmlServiceException e) {

Log.w("TEST","CosXmlServiceException =" + e.toString());
}

//**Use an asynchronous callback request**
/**

cosXmlService.getObjectAsync(getObjectRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult cosXmlResult) {

Log.w("TEST","success");
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceExcepti
on

serviceException) {

String errorMsg = clientException != null ? clientException.toString() : serviceException.toString();
Log.w("TEST",errorMsg);
}
});

*/
```

Copying Objects

This API is used to copy a file from the source path to the destination path. The recommended file size is from 1M to 5G. Files larger than 5G will adopt multipart upload using Upload - Copy. The steps are as follows:

1. Call the `CopyObjectRequest(String,String, CopySourceStruct)` construction method to instantiate the CopyObjectRequest object.
2. Call the copyObject method of CosXmlService, input CopyObjectRequest, and get the returned CopyObjectResult object.
   (Alternatively, call the copyObjectAsync method, and input CopyObjectRequest and CosXmlResultListener for asynchronous callback).

Parameters

Parameter Name	Description	Type	Required
bucket	Bucket name (bucket format of cos v5: xxx-appid, such as test-1253960454)	String	Yes
cosPath	Target object key, which is the absolute path to the storage on COS	String	Yes
copySourceStruct	Source path structure	CopySourceStruct	Yes
signDuration	Validity of the signature (in sec)	Long	Yes
checkHeaderListForSign	Request header in signature for verification	Set<String>	No
checkParameterListForSing	Request parameters in signature for verification	Set<String>	No
cosXmlResultListener	Callback for upload result	CosXmlResultListener	No

Returned result

Request result is returned through member variables of CopyObjectResult object.

Member Variable Name	Description	Type
copyObject	Returns the copy result	CopyObject
httpCode	Request is successful when it is in [200, 300), otherwise request failed	Int

If an exception CosClientException or CosServiceException is thrown, please see **Description on SDK Exceptions** at the beginning.

Example

```
String bucket = "bucket";
```

```
String cosPath = "cosPath";
CopyObjectRequest.CopySourceStruct copySourceStruct = new CopyObjectRequest.CopySourceStruct("Source file's appid",
"Source file's bucket", "Source file's region", "Source file's cosPath");

CopyObjectRequest copyObjectRequest = new CopyObjectRequest(bucket, cosPath, copySourceStruct);
copyObjectRequest.setSign(signDuration,null,null);

//Use synchronization method
try {
CopyObjectResult copyObjectResult = cosXmlService.copyObject(copyObjectRequest);
//Successful
Log.w("TEST","success:" + copyObjectResult.copyObject);

} catch (CosXmlClientException e) {

Log.w("TEST","CosXmlClientException =" + e.toString());

} catch (CosXmlServiceException e) {

Log.w("TEST","CosXmlServiceException =" + e.toString());
}

//**Use an asynchronous callback request**
/**

cosXmlService.copyObjectAsync(copyObjectRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {

Log.w("TEST","success");
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceExcepti
on

serviceException) {

String errorMsg = clientException != null ? clientException.toString() : serviceException.toString();
Log.w("TEST",errorMsg);
}
});

*/
```

Creating a Bucket

This API is used to create a Bucket under the specified account. The steps are as follows:

1. Call the `PutBucketRequest(String)` construction method to instantiate the PutBucketRequest object.
2. Call putBucket method of CosXmlService, input PutBucketRequest, and get the returned PutBucketResult object. (Alternatively, call the putBucketAsync method, and input PutBucketRequest and CosXmlResultListener for asynchronous callback).

Parameters

Parameter Name	Description	Type	Required
bucket	Bucket name (bucket format of cos v5: xxx-appid, such as test-1253960454)	String	Yes
signDuration	Validity of the signature (in sec)	Long	Yes
checkHeaderListForSign	Request header in signature for verification	Set<String>	No
checkParameterListForSing	Request parameters in signature for verification	Set<String>	No
cosXmlResultListener	Callback for upload result	CosXmlResultListener	No

Returned result

Request request is returned through member variables of PutBucketResult object.

Member Variable Name	Description	Type
httpCode	Request is successful when it is in [200, 300), otherwise request failed	Int

If an exception CosClientException or CosServiceException is thrown, please see **Description on SDK Exceptions** at the beginning.

Example

```
PutBucketRequest putBucketRequest = new PutBucketRequest(bucket);
putBucketRequest.setSign(signDuration,null,null);

//Define the ACL attribute of Object. Valid values: private, public-read-write, public-read; Default: private
putBucketRequest.setXCOSACL("private");

//Grant read permission to the authorized user
ACLAccount readACLS = new ACLAccount();
readACLS.addACLAccount("OwnerUin", "SubUin");
putBucketRequest.setXCOSGrantRead(readACLS);

//Grant write permission to the authorized user
```

```
ACLAccount writeACLS = new ACLAccount();
writeACLS.addACLAccount("OwnerUin", "SubUin");
putBucketRequest.setXCOSGrantRead(writeACLS);

//Grant read and write permissions to the authorized user
ACLAccount writeandReadACLS = new ACLAccount();
writeandReadACLS.addACLAccount("OwnerUin", "SubUin");
putBucketRequest.setXCOSGrantRead(writeandReadACLS);

//Use synchronization method
try {
PutBucketResult putBucketResult = cosXmlService.putBucket(putBucketRequest);
//Successful
Log.w("TEST","success");

} catch (CosXmlClientException e) {

Log.w("TEST","CosXmlClientException =" + e.toString());

} catch (CosXmlServiceException e) {

Log.w("TEST","CosXmlServiceException =" + e.toString());
}

//**Use an asynchronous callback request**
/**

cosXmlService.putBucketAsync(putBucketRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {

Log.w("TEST","success");
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceExcepti
on

serviceException) {

String errorMsg = clientException != null ? clientException.toString() : serviceException.toString();
Log.w("TEST",errorMsg);
}
});

*/
```

Checking Whether Bucket Exists

This API is used to confirm the existence of the specified bucket. The steps are as follows:

1. Call the HeadBucketRequest(String) construction method to instantiate the HeadBucketRequest object.
2. Call the headBucket method of CosXmlService, input HeadBucketRequest, and get the returned HeadBucketResult object.
   (Alternatively, call the headBucketAsync method, and input HeadBucketRequest and CosXmlResultListener for asynchronous callback).

Parameters

Parameter Name	Description	Type	Required
bucket	Bucket name (bucket format of cos v5: xxx-appid, such as test-1253960454)	String	Yes
signDuration	Validity of the signature (in sec)	Long	Yes
checkHeaderListForSign	Request header in signature for verification	Set<String>	No
checkParameterListForSing	Request parameters in signature for verification	Set<String>	No
cosXmlResultListener	Callback for upload result	CosXmlResultListener	No

Returned result

Request request is returned through member variables of PutBucketResult object.

Member Variable Name	Description	Type
httpCode	Request is successful when it is in [200, 300), otherwise request failed	Int

If an exception CosClientException or CosServiceException is thrown, please see **Description on SDK Exceptions** at the beginning.

Example

```
HeadBucketRequest headBucketRequest = new HeadBucketRequest(bucket);
headBucketRequest.setSign(signDuration,null,null);

//Use synchronization method
try {
HeadBucketResult headBucketResult = cosXmlService.headBucket(headBucketRequest);
//Successful
Log.w("TEST","success");

} catch (CosXmlClientException e) {
```

```
Log.w("TEST","CosXmlClientException =" + e.toString());

} catch (CosXmlServiceException e) {

Log.w("TEST","CosXmlServiceException =" + e.toString());
}

//**Use an asynchronous callback request**
/**

cosXmlService.headBucketAsync(headBucketRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {

Log.w("TEST","success");
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceExcepti
on

serviceException) {

String errorMsg = clientException != null ? clientException.toString() : serviceException.toString();
Log.w("TEST",errorMsg);
}
});

*/
```

Listing Bucket

This API is used to list some or all of the objects under the Bucket. The steps are as follows:

1. Call the `GetBucketRequest(String)` construction method to instantiate the GetBucketRequest object.
2. Call getBucket method of CosXmlService, input GetBucketRequest, and get the returned GetBucketResult object. (Alternatively, call the getBucketAsync method, and input GetBucketRequest and CosXmlResultListener for asynchronous callback).

Parameters

Parameter Name	Description	Type	Required
bucket	Bucket name (bucket format of cos v5: xxx-appid, such as test-1253960454)	String	Yes
signDuration	Validity of the signature (in sec)	Long	Yes

Parameter Name	Description	Type	Required
checkHeaderListForSign	Request header in signature for verification	Set<String>	No
checkParameterListForSing	Request parameters in signature for verification	Set<String>	No
cosXmlResultListener	Callback for upload result	CosXmlResultListener	No

Returned result

Request request is returned through member variables of GetBucketResult object.

Member Variable Name	Description	Type
listBucket	Store all the information of Get Bucket request result	ListBucket
httpCode	Request is successful when it is in [200, 300), otherwise request failed	Int

If an exception CosClientException or CosServiceException is thrown, please see **Description on SDK Exceptions** at the beginning.

Example

```
GetBucketRequest getBucketRequest = new GetBucketRequest(bucket);
getBucketRequest.setSign(signDuration,null,null);

//Indicates the prefix match, which is used to specify the prefix address of the returned file
getBucketRequest.setPrefix("prefix");

//Maximum number of entries returned at a time. Default is 1,000
getBucketRequest.setMaxKeys(100);

//The delimiter is a sign. If Prefix exists,
//the same paths between Prefix and delimiter are grouped as the same type and defined as Common Prefix,
//and then all Common Prefixes are listed. If Prefix does not exist, the listing process starts from the beginning of the path.
getBucketRequest.setDelimiter('c');

//Use synchronization method
try {
GetBucketResult getBucketResult = cosXmlService.getBucket(getBucketRequest);
//Successful
Log.w("TEST","success :" + getBucketResult.listBucket.toString());

} catch (CosXmlClientException e) {

Log.w("TEST","CosXmlClientException =" + e.toString());
```

```
} catch (CosXmlServiceException e) {

Log.w("TEST","CosXmlServiceException =" + e.toString());
}

//**Use an asynchronous callback request**
/**

cosXmlService.getBucketAsync(getBucketRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {

Log.w("TEST","success");
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceExcepti
on

serviceException) {

String errorMsg = clientException != null ? clientException.toString() : serviceException.toString();
Log.w("TEST",errorMsg);
}
});

*/
```

Deleting Bucket

This API is used to delete a Bucket under a specified account. The Bucket must be empty before it can be deleted. The Bucket can be deleted only if its content is removed. The steps are as follows:

1. Call the `DeleteBucketRequest(String)` construction method to instantiate the DeleteBucketRequest object.
2. Call the deleteBucket method of CosXmlService, input DeleteBucketRequest, and get the returned DeleteBucketResult object.
   (Alternatively, call the deleteBucketAsync method, and input DeleteBucketRequest and CosXmlResultListener for asynchronous callback).

Parameters

Parameter Name	Description	Type	Required
bucket	Bucket name (bucket format of cos v5: xxx-appid, such as test-1253960454)	String	Yes
signDuration	Validity of the signature (in sec)	Long	Yes

Parameter Name	Description	Type	Required
checkHeaderListForSign	Request header in signature for verification	Set<String>	No
checkParameterListForSing	Request parameters in signature for verification	Set<String>	No
cosXmlResultListener	Callback for upload result	CosXmlResultListener	No

Returned result

Request result is returned through member variables of DeleteBucketResult object.

Member Variable Name	Description	Type
httpCode	Request is successful when it is in [200, 300), otherwise request failed	Int

If an exception CosClientException or CosServiceException is thrown, please see **Description on SDK Exceptions** at the beginning.

Example

```
DeleteBucketRequest deleteBucketRequest = new DeleteBucketRequest(bucket);
deleteBucketRequest.setSign(signDuration,null,null);

//Use synchronization method
try {
DeleteBucketResult deleteBucketResult = cosXmlService.deleteBucket(deleteBucketRequest);
//Successful
Log.w("TEST","success");

} catch (CosXmlClientException e) {

Log.w("TEST","CosXmlClientException =" + e.toString());

} catch (CosXmlServiceException e) {

Log.w("TEST","CosXmlServiceException =" + e.toString());
}

//**Use an asynchronous callback request**
/**

cosXmlService.deleteBucketAsync(deleteBucketRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {

Log.w("TEST","success");
```

```
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceExcepti
on

serviceException) {

String errorMsg = clientException != null ? clientException.toString() : serviceException.toString();
Log.w("TEST",errorMsg);
}
});

*/
```

Setting Bucket ACL

This API is used to specify the Bucket's access permission. The steps are as follows:

1. Call the `PutBucketACLRequest(String)` construction method to instantiate the PutBucketACLRequest object.
2. Call putBucketACL method of CosXmlService, input PutBucketACLRequest, and get the returned PutBucketACLResult object.
   (Alternatively, call the putBucketACLAsync method, and input PutBucketACLRequest and CosXmlResultListener for asynchronous callback).

Parameters

Parameter Name	Description	Type	Required
bucket	Bucket name (bucket format of cos v5: xxx-appid, such as test-1253960454)	String	Yes
xcosACL	Sets Bucket's access permissions. Valid values: private, public-read-write, public-read; Default: private	String	No
xcosGrantRead	Grants read permission to the authorized user	ACLAccount	No
xcosGrantWrite	Grants write permission to the authorized user	ACLAccount	No
xcosGrantRead	Grants read and write permission to the authorized user	ACLAccount	No
signDuration	Validity of the signature (in sec)	Long	Yes
checkHeaderListForSign	Request header in signature for verification	Set<String>	No

Parameter Name	Description	Type	Required
checkParameterListForSing	Request parameters in signature for verification	Set<String>	No
cosXmlResultListener	Callback for upload result	CosXmlResultListener	No

Returned result

Request result is returned through member variables of DeleteBucketResult object.

Member Variable Name	Description	Type
httpCode	Request is successful when it is in [200, 300), otherwise request failed	Int

If an exception CosClientException or CosServiceException is thrown, please see **Description on SDK Exceptions** at the beginning.

Example

```
PutBucketACLRequest putBucketACLRequest = new PutBucketACLRequest(bucket);
putBucketACLRequest.setSign(signDuration,null,null);

//Set access permission to the bucket
putBucketACLRequest.setXCOSACL("public-read");

//Grant read permission to the authorized user
ACLAccount readACLS = new ACLAccount();
readACLS.addACLAccount("OwnerUin", "SubUin");
putBucketRequest.setXCOSGrantRead(readACLS);

//Grant write permission to the authorized user
ACLAccount writeACLS = new ACLAccount();
writeACLS.addACLAccount("OwnerUin", "SubUin");
putBucketRequest.setXCOSGrantRead(writeACLS);

//Grant read and write permissions to the authorized user
ACLAccount writeandReadACLS = new ACLAccount();
writeandReadACLS.addACLAccount("OwnerUin", "SubUin");
putBucketRequest.setXCOSGrantRead(writeandReadACLS);

//Use synchronization method
try {
PutBucketACLResult putBucketACLResult = cosXmlService.putBucketACL(putBucketACLRequest);
//Successful
Log.w("TEST","success");

} catch (CosXmlClientException e) {
```

```
Log.w("TEST","CosXmlClientException =" + e.toString());

} catch (CosXmlServiceException e) {

Log.w("TEST","CosXmlServiceException =" + e.toString());
}


//**Use an asynchronous callback request**
/**

cosXmlService.putBucketACLAsync(putBucketACLRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {

Log.w("TEST","success");
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceExcepti
on

serviceException) {

String errorMsg = clientException != null ? clientException.toString() : serviceException.toString();
Log.w("TEST",errorMsg);
}
});

*/
```

Obtaining Bucket ACL

This API is used to obtain the Bucket ACL. The steps are as follows:

1. Call the `GetBucketACLRequest(String)` construction method to instantiate the GetBucketACLRequest object.
2. Call getBucketACL method of CosXmlService, input GetBucketACLRequest, and get the returned GetBucketACLResult object.
   (Alternatively, call the getBucketACLAsync method, and input GetBucketACLRequest and CosXmlResultListener for asynchronous callback).

Parameters

Parameter Name	Description	Type	Required
bucket	Bucket name (bucket format of cos v5: xxx-appid, such as test-1253960454)	String	Yes

Parameter Name	Description	Type	Required
signDuration	Validity of the signature (in sec)	Long	Yes
checkHeaderListForSign	Request header in signature for verification	Set<String>	No
checkParameterListForSing	Request parameters in signature for verification	Set<String>	No
cosXmlResultListener	Callback for upload result	CosXmlResultListener	No

Returned result

Request request is returned through member variables of GetBucketACLResult object.

Member Variable Name	Description	Type
accessControlPolicy	About authorized users and their permissions	AccessControlPolicy
httpCode	Request is successful when it is in [200, 300), otherwise request failed	Int

If an exception CosClientException or CosServiceException is thrown, please see **Description on SDK Exceptions** at the beginning.

Example

```
GetBucketACLRequest getBucketACLRequest = new DeleteBucketRequest(bucket);
getBucketACLRequest.setSign(signDuration,null,null);

//Use synchronization method
try {
GetBucketACLResult getBucketACLResult = cosXmlService.getBucketACL(getBucketACLRequest);
//Successful
Log.w("TEST","success: " +getBucketACLResult.accessControlPolicy.toString() );

} catch (CosXmlClientException e) {

Log.w("TEST","CosXmlClientException =" + e.toString());

} catch (CosXmlServiceException e) {

Log.w("TEST","CosXmlServiceException =" + e.toString());
}

//**Use an asynchronous callback request**
/**
```

```
cosXmlService.getBucketACLAsync(getBucketACLRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {

Log.w("TEST","success");
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceExcepti
on

serviceException) {

String errorMsg = clientException != null ? clientException.toString() : serviceException.toString();
Log.w("TEST",errorMsg);
}
});

*/
```

Setting Cross-domain Access Configuration

This API is used to configure cross-origin access for the specified Bucket. The steps are as follows:

1. Call the `PutBucketCORSRequest(String)` construction method to instantiate the PutBucketCORSRequest object.
2. Call putBucketCORS method of CosXmlService, input PutBucketCORSRequest, and get the returned PutBucketCORSResult object.
   (Alternatively, call the putBucketCORSAsync method, and input PutBucketCORSRequest and CosXmlResultListener for asynchronous callback).

Parameters

Parameter Name	Description	Type	Required
bucket	Bucket name (bucket format of cos v5: xxx-appid, such as test-1253960454)	String	Yes
cORSRule	Configurations on cross-origin access	CORSConfiguration.CORSRule	Yes
signDuration	Validity of the signature (in sec)	Long	Yes
checkHeaderListForSign	Request header in signature for verification	Set<String>	No

Parameter Name	Description	Type	Required
checkParameterListForSing	Request parameters in signature for verification	Set<String>	No
cosXmlResultListener	Callback for upload result	CosXmlResultListener	No

Returned result

Request request is returned through member variables of PutBucketCORSResult object.

Member Variable Name	Description	Type
httpCode	Request is successful when it is in [200, 300), otherwise request failed	Int

If an exception CosClientException or CosServiceException is thrown, please see **Description on SDK Exceptions** at the beginning.

Example

```
PutBucketCORSRequest putBucketCORSRequest = new PutBucketCORSRequest(bucket);

/**
CORSConfiguration.cORSRule: Configurations on cross-origin access
corsRule.id: Sets rule ID
cORSRule.allowedOrigin: Allowed access sources. The wildcard "*" is supported. Format: protocol://domain_name[:port], for example, http://www.qq.com
corsRule.maxAgeSeconds: Configure the valid period of the results obtained by OPTIONS request
corsRule.allowedMethod: Allowed HTTP operations, such as GET, PUT, HEAD, POST, and DELETE
corsRule.allowedHeader: When sending an OPTIONS request, notify the server end about which custom HTTP request headers are allowed to be used by subsequent requests. The wildcard "*" is supported.
corsRule.exposeHeader: Configure the custom header information that can be received by the browser from the server end
*/
CORSConfiguration.CORSRule corsRule = new CORSConfiguration.CORSRule();

corsRule.id = "123";
corsRule.allowedOrigin = "https://cloud.tencent.com";
corsRule.maxAgeSeconds = "5000";

List<String> methods = new LinkedList<>();
methods.add("PUT");
methods.add("POST");
methods.add("GET");
corsRule.allowedMethod = methods;

List<String> headers = new LinkedList<>();
headers.add("host");
```

```
headers.add("content-type");
corsRule.allowedHeader = headers;

List<String> exposeHeaders = new LinkedList<>();
headers.add("x-cos-metha-1");
corsRule.exposeHeader = headers;

//Configure cross-origin access
putBucketCORSRequest.addCORSRule(corsRule);

putBucketCORSRequest.setSign(signDuration,null,null);

//Use synchronization method
try {
PutBucketCORSResult putBucketCORSResult = cosXmlService.putBucketCORS(putBucketCORSRequest);
//Successful
Log.w("TEST","success");

} catch (CosXmlClientException e) {

Log.w("TEST","CosXmlClientException =" + e.toString());

} catch (CosXmlServiceException e) {

Log.w("TEST","CosXmlServiceException =" + e.toString());
}

//**Use an asynchronous callback request**
/**

cosXmlService.putBucketCORSAsync(putBucketCORSRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {

Log.w("TEST","success");
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceExcepti
on

serviceException) {

String errorMsg = clientException != null ? clientException.toString() : serviceException.toString();
Log.w("TEST",errorMsg);
}
});

*/
```

Obtaining Cross-domain Access Configuration

This API is used to obtain configurations on cross-origin access to the specified Bucket. The steps are as follows:

1. Call the `GetBucketCORSRequest(String)` construction method to instantiate the GetBucketCORSRequest object.
2. Call getBucketCORS method of CosXmlService, input GetBucketCORSRequest, and get the returned GetBucketCORSResult object.
   (Alternatively, call the getBucketCORSAsync method, and input GetBucketCORSRequest and CosXmlResultListener for asynchronous callback).

Parameters

Parameter Name	Description	Type	Required
bucket	Bucket name (bucket format of cos v5: xxx-appid, such as test-1253960454)	String	Yes
signDuration	Validity of the signature (in sec)	Long	Yes
checkHeaderListForSign	Request header in signature for verification	Set<String>	No
checkParameterListForSing	Request parameters in signature for verification	Set<String>	No
cosXmlResultListener	Callback for upload result	CosXmlResultListener	No

Returned result

Request request is returned through member variables of GetBucketCORSResult object.

Member Variable Name	Type	Description
cORSConfiguration	All configurations on cross-origin resource sharing	CORSConfiguration
httpCode	Request is successful when it is in [200, 300), otherwise request failed	Int

If an exception CosClientException or CosServiceException is thrown, please see **Description on SDK Exceptions** at the beginning.

Example

```
GetBucketCORSRequest getBucketCORSRequest = new GetBucketCORSRequest(bucket);
getBucketCORSRequest.setSign(signDuration,null,null);

//Use synchronization method
try {
```

```
GetBucketCORSResult getBucketCORSResult = cosXmlService.getBucketCORS(getBucketCORSRequest);
//Successful
Log.w("TEST","success :" + getBucketCORSResult.corsConfiguration.toString());

} catch (CosXmlClientException e) {

Log.w("TEST","CosXmlClientException =" + e.toString());

} catch (CosXmlServiceException e) {

Log.w("TEST","CosXmlServiceException =" + e.toString());
}

//**Use an asynchronous callback request**
/**

cosXmlService.getBucketCORSAsync(getBucketCORSRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {

GetBucketCORSResult getBucketCORSResult = (GetBucketCORSResult)result;
Log.w("TEST","success:" + getBucketCORSResult.corsConfiguration.toString());;
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceExcepti
on

serviceException) {

String errorMsg = clientException != null ? clientException.toString() : serviceException.toString();
Log.w("TEST",errorMsg);
}
});

*/
```

Deleting Cross-domain Access Configuration

This API is used to delete the cross-domain access configuration information of the specified Bucket. Specific steps are as follows:

1. Call the `DeleteBucketCORSRequest(String)` construction method to instantiate the DeleteBucketCORSRequest object.
2. Call the deleteBucketCORS method of CosXmlService, input DeleteBucketCORSRequest, and get the returned DeleteBucketCORSResult object.

(Alternatively, call the deleteBucketCORSAsync method, and input DeleteBucketCORSRequest and CosXmlResultListener for asynchronous callback).

Parameters

Parameter Name	Description	Type	Required
bucket	Bucket name (bucket format of cos v5: xxx-appid, such as test-1253960454)	String	Yes
signDuration	Validity of the signature (in sec)	Long	Yes
checkHeaderListForSign	Request header in signature for verification	Set<String>	No
checkParameterListForSing	Request parameters in signature for verification	Set<String>	No
cosXmlResultListener	Callback for upload result	CosXmlResultListener	No

Returned result

Request result is returned through member variable of DeleteBucketCORSResult object.

Member Variable Name	Description	Type
httpCode	Request is successful when it is in [200, 300), otherwise request failed	Int

If an exception CosClientException or CosServiceException is thrown, please see **Description on SDK Exceptions** at the beginning.

Example

```
DeleteBucketCORSRequest deleteBucketCORSRequest = new DeleteBucketCORSRequest(bucket);
deleteBucketCORSRequest.setSign(signDuration,null,null);

//Use synchronization method
try {
DeleteBucketCORSResult deleteBucketCORSResult = cosXmlService.deleteBucketCORS(deleteBucketCORSRequest);
//Successful
Log.w("TEST","success");

} catch (CosXmlClientException e) {

Log.w("TEST","CosXmlClientException =" + e.toString());

} catch (CosXmlServiceException e) {
```

```
Log.w("TEST","CosXmlServiceException =" + e.toString());
}

//**Use an asynchronous callback request**
/**

cosXmlService.deleteBucketCORSAsync(deleteBucketCORSRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {

Log.w("TEST","success");
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceExcepti
on

serviceException) {

String errorMsg = clientException != null ? clientException.toString() : serviceException.toString();
Log.w("TEST",errorMsg);
}
});

*/
```

Obtaining Bucket's Region Information

This API is used to obtain the region where the Bucket resides. The steps are as follows:

1. Call the `GetBucketLocationRequest(String)` construction method to instantiate the GetBucketLocationRequest object.
2. Call getBucketLocation method of CosXmlService, input GetBucketLocationRequest, and get the returned GetBucketLocationResult object.
   (Alternatively, call the getBucketLocationAsync method, and input GetBucketLocationRequest and CosXmlResultListener for asynchronous callback).

Parameters

Parameter Name	Description	Type	Required
bucket	Bucket name (bucket format of cos v5: xxx-appid, such as test-1253960454)	String	Yes
signDuration	Validity of the signature (in sec)	Long	Yes
checkHeaderListForSign	Request header in signature for verification	Set<String>	No

Parameter Name	Description	Type	Required
checkParameterListForSing	Request parameters in signature for verification	Set<String>	No
cosXmlResultListener	Callback for upload result	CosXmlResultListener	No

Returned result

Request result is returned through member variables of GetBucketLocationResult object.

Member Variable Name	Description	Type
locationConstraint	The region where the Bucket resides	LocationConstraint
httpCode	Request is successful when it is in [200, 300), otherwise request failed	Int

If an exception CosClientException or CosServiceException is thrown, please see **Description on SDK Exceptions** at the beginning.

Example

```
GetBucketLocationRequest getBucketLocationRequest = new DeleteBucketCORSRequest(bucket);
getBucketLocationRequest.setSign(signDuration,null,null);

//Use synchronization method
try {
GetBucketLocationResult getBucketLocationResult = cosXmlService.getBucketLocation(getBucketLocationReques
t);
//Successful
Log.w("TEST","success : " + getBucketLocationResult.LocationConstraint.location);

} catch (CosXmlClientException e) {

Log.w("TEST","CosXmlClientException =" + e.toString());

} catch (CosXmlServiceException e) {

Log.w("TEST","CosXmlServiceException =" + e.toString());
}

//**Use an asynchronous callback request**
/**

cosXmlService.getBucketLocationAsync(getBucketLocationRequest, new CosXmlResultListener() {
@Override
```

```
public void onSuccess(CosXmlRequest request, CosXmlResult result) {

Log.w("TEST","success");
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceExcepti
on

serviceException) {

String errorMsg = clientException != null ? clientException.toString() : serviceException.toString();
Log.w("TEST",errorMsg);
}
});

*/
```

Configuring Bucket Lifecycle

This API is used to set the lifecycle of a bucket. The steps are as follows:

1. Call the `PutBucketLifecycleRequest(String)` construction method to instantiate the PutBucketLifecycleRequest object.
2. Call putBucketLifecycle method of CosXmlService, input PutBucketLifecycleRequest, and get the returned PutBucketLifecycleResult object.
   (Alternatively, call the putBucketLifecycleAsync method, and input PutBucketLifecycleRequest and CosXmlResultListener for asynchronous callback).

Parameters

Parameter Name	Description	Type	Required
bucket	Bucket name (bucket format of cos v5: xxx-appid, such as test-1253960454)	String	Yes
rule	Rules for configuring a life cycle	LifecycleConfiguration.Rule	Yes
signDuration	Validity of the signature (in sec)	Long	Yes
checkHeaderListForSign	Request header in signature for verification	Set<String>	No
checkParameterListForSing	Request parameters in signature for verification	Set<String>	No
cosXmlResultListener	Callback for upload result	CosXmlResultListener	No

Returned result

Request result is returned through member variables of PutBucketLifecycleResult object.

Member Variable Name	Description	Type
httpCode	Request is successful when it is in [200, 300), otherwise request failed	Int

If an exception CosClientException or CosServiceException is thrown, please see **Description on SDK Exceptions** at the beginning.

Example

```
PutBucketLifecycleRequest putBucketLifecycleRequest = new PutBucketLifecycleRequest(bucket);
putBucketLifecycleRequest.setSign(signDuration,null,null);

//Declare lifecycle configuration rules
LifecycleConfiguration.Rule rule = new LifecycleConfiguration.Rule();
rule.id = "Lifecycle ID";
LifecycleConfiguration.Filter filter = new LifecycleConfiguration.Filter();
filter.prefix = "prefix/";
rule.filter = filter;
rule.status = "Enabled or Disabled";
LifecycleConfiguration.Transition transition = new LifecycleConfiguration.Transition();
transition.days = 100;
transition.storageClass = COSStorageClass.STANDARD.getStorageClass();
putBucketLifecycleRequest.setRuleList(rule);

//Use synchronization method
try {
PutBucketLifecycleResult putBucketLifecycleResult = cosXmlService.putBucketLifecycle(putBucketLifecycleReques
t);
//Successful
Log.w("TEST","success : " + getBucketLocationResult.region);

} catch (CosXmlClientException e) {

Log.w("TEST","CosXmlClientException =" + e.toString());

} catch (CosXmlServiceException e) {

Log.w("TEST","CosXmlServiceException =" + e.toString());
}

//**Use an asynchronous callback request**
/**

cosXmlService.putBucketLifecycleAsync(putBucketLifecycleRequest, new CosXmlResultListener() {
```

```
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {

Log.w("TEST","success");
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceExcepti
on

serviceException) {

String errorMsg = clientException != null ? clientException.toString() : serviceException.toString();
Log.w("TEST",errorMsg);
}
});

*/
```

Obtaining Bucket Lifecycle

This API is used to obtain the lifecycle configuration of a bucket. The steps are as follows:

1. Call the `GetBucketLifecycleRequest(String)` construction method to instantiate the GetBucketLifecycleRequest object.
2. Call getBucketLifecycle method of CosXmlService, input GetBucketLifecycleRequest, and get the returned GetBucketLifecycleResult object.
   (Alternatively, call the getBucketLifecycleAsync method, and input GetBucketLifecycleRequest and CosXmlResultListener for asynchronous callback).

Parameters

Parameter Name	Description	Type	Required
bucket	Bucket name (bucket format of cos v5: xxx-appid, such as test-1253960454)	String	Yes
signDuration	Validity of the signature (in sec)	Long	Yes
checkHeaderListForSign	Request header in signature for verification	Set<String>	No
checkParameterListForSing	Request parameters in signature for verification	Set<String>	No
cosXmlResultListener	Callback for upload result	CosXmlResultListener	No

Returned result

Request result is returned through member variables of getBucketLifecycle object.

Member Variable Name	Description	Type
lifecycleConfiguration	Lifecycle configurations	LifecycleConfiguration
httpCode	Request is successful when it is in [200, 300), otherwise request failed	Int

If an exception CosClientException or CosServiceException is thrown, please see **Description on SDK Exceptions** at the beginning.

Example

```
GetBucketLifecycleRequest getBucketLifecycleRequest = new DeleteBucketCORSRequest(bucket);
getBucketLifecycleRequest.setSign(signDuration,null,null);

//Use synchronization method
try {
GetBucketLifecycleResult getBucketLifecycleResult = cosXmlService.getBucketLifecycle(getBucketLifecycleReques
t);
//Successful
Log.w("TEST","success : " + getBucketLifecycleResult.lifecycleConfiguration.toString());

} catch (CosXmlClientException e) {

Log.w("TEST","CosXmlClientException =" + e.toString());

} catch (CosXmlServiceException e) {

Log.w("TEST","CosXmlServiceException =" + e.toString());
}

//**Use an asynchronous callback request**
/**

cosXmlService.getBucketLifecycleAsync(getBucketLifecycleResult, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {

Log.w("TEST","success");
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceExcepti
on
```

```
serviceException) {

String errorMsg = clientException != null ? clientException.toString() : serviceException.toString();
Log.w("TEST",errorMsg);
}
});

*/
```

Deleting Bucket Lifecycle

This API is used to delete the lifecycle configuration of a bucket. The steps are as follows:

1. Call the `DeleteBucketLifecycleRequest(String)` construction method to instantiate the DeleteBucketLifecycleRequest object.
2. Call the deleteBucketLifecycle method of CosXmlService, input DeleteBucketLifecycleRequest, and get the returned DeleteBucketLifecycleResult object.
   (Alternatively, call the deleteBucketLifecycleAsync method, and input DeleteBucketLifecycleRequest and CosXmlResultListener for asynchronous callback).

Parameters

Parameter Name	Description	Type	Required
bucket	Bucket name (bucket format of cos v5: xxx-appid, such as test-1253960454)	String	Yes
signDuration	Validity of the signature (in sec)	Long	Yes
checkHeaderListForSign	Request header in signature for verification	Set<String>	No
checkParameterListForSing	Request parameters in signature for verification	Set<String>	No
cosXmlResultListener	Callback for upload result	CosXmlResultListener	No

Returned result

Request result is returned through member variables of DeleteBucketLifecycleResult object.

Member Variable Name	Description	Type
httpCode	Request is successful when it is in [200, 300), otherwise request failed	Int

If an exception CosClientException or CosServiceException is thrown, please see **Description on SDK Exceptions** at the beginning.

Example

```
DeleteBucketLifecycleRequest deleteBucketLifecycleRequest = new DeleteBucketCORSRequest(bucket);
deleteBucketLifecycleRequest.setSign(signDuration,null,null);

//Use synchronization method
try {
DeleteBucketLifecycleResult deleteBucketCORSResult = cosXmlService.deleteBucketLifecycle(deleteBucketLifecycl
eRequest);
//Successful
Log.w("TEST","success ");

} catch (CosXmlClientException e) {

Log.w("TEST","CosXmlClientException =" + e.toString());

} catch (CosXmlServiceException e) {

Log.w("TEST","CosXmlServiceException =" + e.toString());
}

//**Use an asynchronous callback request**
/**

cosXmlService.deleteBucketLifecycleAsync(deleteBucketLifecycleRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {

Log.w("TEST","success");
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceExcepti
on

serviceException) {

String errorMsg = clientException != null ? clientException.toString() : serviceException.toString();
Log.w("TEST",errorMsg);
}
});

*/
```

Querying Parts being Uploaded in Bucket

This API is used to obtain multipart upload operations that are still in process. A maximum of 1,000 such operations can be listed at a time. The steps are as follows:

1. Call the `ListMultiUploadsRequest(String)` construction method to instantiate the ListMultiUploadsRequest object.
2. Call the listMultiUploads method of CosXmlService, input ListMultiUploadsRequest, and get the returned ListMultiUploadsResult object.
   (Alternatively, call the listMultiUploadsAsync method, and input ListMultiUploadsRequest and CosXmlResultListener for asynchronous callback).

Parameters

Parameter Name	Description	Type	Required
bucket	Bucket name (bucket format of cos v5: xxx-appid, such as test-1253960454)	String	Yes
signDuration	Validity of the signature (in sec)	Long	Yes
checkHeaderListForSign	Request header in signature for verification	Set<String>	No
checkParameterListForSing	Request parameters in signature for verification	Set<String>	No
cosXmlResultListener	Callback for upload result	CosXmlResultListener	No

Returned result

Request result is returned through member variables of ListMultiUploadsResult object.

Member Variable Name	Description	Type
listMultipartUploads	Information on all multipart upload operations	ListMultipartUploads
httpCode	Request is successful when it is in [200, 300), otherwise request failed	Int

If an exception CosClientException or CosServiceException is thrown, please see **Description on SDK Exceptions** at the beginning.

Example

```
ListMultiUploadsRequest listMultiUploadsRequest = new ListMultiUploadsRequest(bucket);
listMultiUploadsRequest.setSign(signDuration,null,null);

//Use synchronization method
try {
ListMultiUploadsResult listMultiUploadsResult = cosXmlService.listMultiUploads(listMultiUploadsRequest);
//Successful
Log.w("TEST","success: " + listMultiUploadsResult.listMultipartUploads.toString());
```

```
} catch (CosXmlClientException e) {

Log.w("TEST","CosXmlClientException =" + e.toString());

} catch (CosXmlServiceException e) {

Log.w("TEST","CosXmlServiceException =" + e.toString());
}

//**Use an asynchronous callback request**
/**

cosXmlService.listMultiUploadsAsync(listMultiUploadsRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {

Log.w("TEST","success");
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceExcepti
on

serviceException) {

String errorMsg = clientException != null ? clientException.toString() : serviceException.toString();
Log.w("TEST",errorMsg);
}
});

*/
```

Setting up Multiple Versions

This API is used to setup version control for a Bucket. The steps are as follows:

1. Call the PutBucketVersioningRequest construction method to instantiate the PutBucketVersioningRequest object.
2. Call the putBucketVersioning(PutBucketVersioningRequest) synchronization method of CosXmlService, input PutBucketVersioningRequest, and get the returned PutBucketVersioningResult object. (Alternatively, call the putBucketVersionAsync method, and input PutBucketVersioningRequest and CosXmlResultListener for asynchronous callback).

Parameters

Parameter Name	Description	Type	Required
bucket	Bucket name (bucket format of cos v5: xxx-appid, such as test-1253960454)	String	Yes

Parameter Name	Description	Type	Required
isEnable	Whether to enable multi-version control. Use true to enable or false to disable	boolean	Yes
signDuration	Validity of the signature (in sec)	Long	Yes
checkHeaderListForSign	Request header in signature for verification	Set<String>	No
checkParameterListForSing	Request parameters in signature for verification	Set<String>	No
cosXmlResultListener	Callback for upload result	CosXmlResultListener	No

Returned result

Request result is returned through member variables of PutBucketVersioningResult object.

Member Variable Name	Description	Type
httpCode	Request is successful when it is in [200, 300), otherwise request failed	Int

If an exception CosClientException or CosServiceException is thrown, please see **Description on SDK Exceptions** at the beginning.

Example

```
String bucket = "bucket";
PutBucketVersioningRequest request = new PutBucketVersioningRequest(bucket);
request.setEnableVersion(true);//Enable
request.setSign(signDuration,null,null); //Signature
try {
PutBucketVersioningResult result = cosXmlService.putBucketVersioning(request);
Log.w("TEST","success");
} catch (CosXmlClientException e) {
Log.w("TEST","CosXmlClientException =" + e.toString());
} catch (CosXmlServiceException e) {
Log.w("TEST","CosXmlServiceException =" + e.toString());
}
//**Use an asynchronous callback request**
/**

cosXmlService.putBucketVersionAsync(request, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult cosXmlResult) {

Log.w("TEST","success");
}
```

```
@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceExcepti
on

serviceException) {

String errorMsg = clientException != null ? clientException.toString() : serviceException.toString();
Log.w("TEST",errorMsg);
}
});
*/
```

Retrieving Version Control Configuration

This API is used to obtain the version control configuration of the specified Bucket.

1. Call the GetBucketVersioningRequest construction method to instantiate the GetBucketVersioningRequest object.
2. Call the getBucketVersioning(GetBucketVersioningRequest) synchronization method of CosXmlService, input GetBucketVersioningRequest, and get the returned GetBucketVersioningResult object. (Alternatively, call the getBucketVersioningAsync method, and input GetBucketVersioningRequest and CosXmlResultListener for asynchronous callback).

Parameters

Parameter Name	Description	Type	Required
bucket	Bucket name (bucket format of cos v5: xxx-appid, such as test-1253960454)	String	Yes
signDuration	Validity of the signature (in sec)	Long	Yes
checkHeaderListForSign	Request header in signature for verification	Set<String>	No
checkParameterListForSing	Request parameters in signature for verification	Set<String>	No
cosXmlResultListener	Callback for upload result	CosXmlResultListener	No

Returned result

Request result is returned through member variables of GetBucketVersioningResult object.

Member Variable Name	Description	Type
versioningConfiguration	Version control configuration	VersioningConfiguration
httpCode	Request is successful when it is in [200, 300), otherwise request failed	Int

If an exception CosClientException or CosServiceException is thrown, please see **Description on SDK Exceptions** at the beginning.

Example

```
String bucket = "bucket";
GetBucketVersioningRequest request = new GetBucketVersioningRequest(bucket);
request.setSign(signDuration,null,null); //Signature
try {
GetBucketVersioningResult result = cosXmlService.getBucketVersioning(request);
Log.w("TEST","success");
} catch (CosXmlClientException e) {
Log.w("TEST","CosXmlClientException =" + e.toString());
} catch (CosXmlServiceException e) {
Log.w("TEST","CosXmlServiceException =" + e.toString());
}
//**Use an asynchronous callback request**
/**

cosXmlService.getBucketVersioningAsync(request, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult cosXmlResult) {

Log.w("TEST","success");
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceExcepti
on

serviceException) {

String errorMsg = clientException != null ? clientException.toString() : serviceException.toString();
Log.w("TEST",errorMsg);
}
});
*/
```

Setting Up cross-origin replication

This API is used to configure asynchronous replication between buckets in different domains.

1. Call the PutBucketReplicationRequest construction method to instantiate the PutBucketReplicationRequest object.
2. Call the putBucketReplication(PutBucketReplicationRequest) synchronization method of CosXmlService, input PutBucketReplicationRequest, and get the returned PutBucketReplicationResult object. (Alternatively, call the putBucketReplicationAsync method, and input PutBucketReplicationRequest and CosXmlResultListener for asynchronous callback).

Parameters

Parameter Name	Description	Type	Required
bucket	Bucket name (bucket format of cos v5: xxx-appid, such as test-1253960454)	String	Yes
ownerUin	Sets Owner Uin for identifying the replication initiator	String	Yes
subUin	Sets sub Uin for identifying the replication initiator	String	Yes
ruleStruct	Cross-domain configurations. A maximum of 1,000 rules are supported. All rules must directed to one destination bucket.	RuleStruct	Yes
signDuration	Validity of the signature (in sec)	Long	Yes
checkHeaderListForSign	Request header in signature for verification	Set<String>	No
checkParameterListForSing	Request parameters in signature for verification	Set<String>	No
cosXmlResultListener	Callback for upload result	CosXmlResultListener	No

Returned result

Request result is returned through member variables of PutBucketReplicationResult object.

Member Variable Name	Description	Type
httpCode	Request is successful when it is in [200, 300), otherwise request failed	Int

If an exception CosClientException or CosServiceException is thrown, please see **Description on SDK Exceptions** at the beginning.

Example

```
String bucket = "bucket";
PutBucketReplicationRequest request = new PutBucketReplicationRequest(bucket);
PutBucketReplicationRequest.RuleStruct ruleStruct = new PutBucketReplicationRequest.RuleStruct();
ruleStruct.id = "replication_id";
ruleStruct.isEnable = true;
ruleStruct.appid = "1253960454";
ruleStruct.bucket = "replicationtest";
ruleStruct.region = "ap-beijing";
request.setReplicationConfigurationWithRule(ruleStruct);
request.setReplicationConfigurationWithRole("ownerUin", "subUin");
```

```
request.setSign(signDuration,null,null); //Signature
try {
PutBucketReplicationResult result = cosXmlService.putBucketReplication(request);
Log.w("TEST","success");
} catch (CosXmlClientException e) {
Log.w("TEST","CosXmlClientException =" + e.toString());
} catch (CosXmlServiceException e) {
Log.w("TEST","CosXmlServiceException =" + e.toString());
}
//**Use an asynchronous callback request**
/**

cosXmlService.putBucketReplicationAsync(request, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult cosXmlResult) {

Log.w("TEST","success");
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceExcepti
on

serviceException) {

String errorMsg = clientException != null ? clientException.toString() : serviceException.toString();
Log.w("TEST",errorMsg);
}
});
*/
```

Retrieving Cross-origin Configuration

This API is used to obtain the cross-origin configuration of the specified Bucket.

1. Call the GetBucketReplicationRequest construction method to instantiate the GetBucketReplicationRequest object.
2. Call the getBucketReplication(GetBucketReplicationRequest) synchronization method of CosXmlService, input GetBucketReplicationRequest, and get the returned GetBucketReplicationResult object. (Alternatively, call the getBucketReplicationAsync method, and input GetBucketReplicationRequest and CosXmlResultListener for asynchronous callback).

Parameters

Parameter Name	Description	Type	Required
bucket	Bucket name (bucket format of cos v5: xxx-appid, such as test-1253960454)	String	Yes

Parameter Name	Description	Type	Required
signDuration	Validity of the signature (in sec)	Long	Yes
checkHeaderListForSign	Request header in signature for verification	Set<String>	No
checkParameterListForSing	Request parameters in signature for verification	Set<String>	No
cosXmlResultListener	Callback for upload result	CosXmlResultListener	No

Returned result

Request result is returned through member variables of GetBucketReplicationResult object.

Member Variable Name	Description	Type
replicationConfiguration	Cross-origin configuration	ReplicationConfiguration
httpCode	Request is successful when it is in [200, 300), otherwise request failed	Int

If an exception CosClientException or CosServiceException is thrown, please see **Description on SDK Exceptions** at the beginning.

Example

```
String bucket = "bucket";
GetBucketReplicationRequest request = new GetBucketReplicationRequest(bucket);
request.setSign(signDuration,null,null); //Signature
try {
GetBucketReplicationResult result = cosXmlService.getBucketReplication(request);
Log.w("TEST","success");
} catch (CosXmlClientException e) {
Log.w("TEST","CosXmlClientException =" + e.toString());
} catch (CosXmlServiceException e) {
Log.w("TEST","CosXmlServiceException =" + e.toString());
}
//**Use an asynchronous callback request**
/**

cosXmlService.getBucketReplicationAsync(request, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult cosXmlResult) {

Log.w("TEST","success");
}
```

```
@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceExcepti
on

serviceException) {

String errorMsg = clientException != null ? clientException.toString() : serviceException.toString();
Log.w("TEST",errorMsg);
}
});
*/
```

Deleting Cross-origin Replication

This API is used to delete the cross-origin configuration of the specified Bucket.

1. Call the DeleteBucketReplicationRequest construction method to instantiate the DeleteBucketReplicationRequest object.
2. Call the deleteBucketReplication(DeleteBucketReplicationRequest) synchronization method of CosXmlService, input DeleteBucketReplicationRequest, and get the returned DeleteBucketReplicationResult object. (Alternatively, call the deleteBucketReplicationAsync method, and input DeleteBucketReplicationRequest and CosXmlResultListener for asynchronous callback).

Parameters

Parameter Name	Description	Type	Required
bucket	Bucket name (bucket format of cos v5: xxx-appid, such as test-1253960454)	String	Yes
signDuration	Validity of the signature (in sec)	Long	Yes
checkHeaderListForSign	Request header in signature for verification	Set<String>	No
checkParameterListForSing	Request parameters in signature for verification	Set<String>	No
cosXmlResultListener	Callback for upload result	CosXmlResultListener	No

Returned result

Request result is returned through member variables of DeleteBucketReplicationResult object.

Member Variable Name	Description	Type
httpCode	Request is successful when it is in [200, 300), otherwise request failed	Int

If an exception CosClientException or CosServiceException is thrown, please see **Description on SDK Exceptions** at the beginning.

Example

```
String bucket = "bucket";
DeleteBucketReplicationRequest request = new DeleteBucketReplicationRequest(bucket);
request.setSign(signDuration,null,null); //Signature
try {
DeleteBucketReplicationResult result = cosXmlService.deleteBucketReplication(request);
Log.w("TEST","success");
} catch (CosXmlClientException e) {
Log.w("TEST","CosXmlClientException =" + e.toString());
} catch (CosXmlServiceException e) {
Log.w("TEST","CosXmlServiceException =" + e.toString());
}
//**Use an asynchronous callback request**
/**

cosXmlService.deleteBucketReplicationAsync(request, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult cosXmlResult) {

Log.w("TEST","success");
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceExcepti
on

serviceException) {

String errorMsg = clientException != null ? clientException.toString() : serviceException.toString();
Log.w("TEST",errorMsg);
}
});
*/
```

C SDK
Getting Started

Last updated：2018-09-28 16:33:45

Preparations for Development

Related resources

Download XML C SDK resources of COS service from XML C SDK GitHub .
Download Demo from: XML C SDK Demo.

Development environment

1. Click here to download and install CMake tool (version 2.6.0 or above is recommended). Typical setup is as follows:

```
./configure
make
make install
```

2. Click here to download and install libcurl (version 7.32.0 or above is recommended). Typical setup is as follows:

```
./configure
make
make install
```

3. Click here to download and install apr (version 1.5.2 or above is recommended). Typical setup is as follows:

```
./configure
make
make install
```

4. Click here to download and install apr-util (version 1.5.4 or above is recommended). Typical setup is as follows:

```
./configure --with-apr=/your/apr/install/path
make
make install
```

5. Click here to download and install minixml (version 2.8 or above is recommended). Typical setup is as follows:

```
./configure
make
sudo make install
```

Installing SDK

Install via source code. Download source code from GitHub. Typical compiling command is as follows:

```
cmake .
make
make install
```

SDK Initialization

Initializing SDK Operating Environment

```
int main(int argc, char *argv[])
{
/*Call cos_http_io_initialize method at the program entry to initialize global resources internally, including networ
k, memory, etc.*/
? ?if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
exit(1);
}

/* Call COS SDK API to upload/download file */
/* ... User logic code is ignored here */

/* Call cos_http_io_deinitialize method to release global resources assigned previously before the program ends
*/
cos_http_io_deinitialize();
return 0;
}
```

Initializing Request Options

```
/*Equivalent to apr_pool_t, the memory pool for memory management. The implementation code can be found i
n apr library */
cos_pool_t *pool;
cos_request_options_t *options;

/* Re-create a new memory pool. The second parameter is NULL, which indicates it is not inherited from other me
mory pools */
cos_pool_create(&pool, NULL);

/* Create and initialize options. This parameter contains global configuration information such as endpoint, acces
s_key_id, acces_key_secret, is_cname, and curl parameters.
* The memory of the options is assigned by the pool. After the pool is released, the memory of the options is also
released. There is no need to release the memory separately.
*/
options = cos_request_options_create(pool);
options->config = cos_config_create(options->pool);

/* cos_str_set initializes the cos_string_t type with a char* type string*/
cos_str_set(&options->config->endpoint, "<user's Endpoint>"); //Enter Endpoint according to the COS service do
```

```
main name of user's region
cos_str_set(&options->config->access_key_id, "<user's SecretId>"); //SecretId obtained after the COS service is reg
istered
cos_str_set(&options->config->access_key_secret, "<user's SecretKey>"); //SecretKey obtained after the COS servic
e is registered
cos_str_set(&options->config->appid, "<user's AppId>"); //AppId obtained after the COS service is registered

/* Whether CNAME is used */
options->config->is_cname = 0;

/* Used to set network-related parameters, such as timeout*/
options->ctl = cos_http_controller_create(options->pool, 0);
```

How to Use SDK

1. Initialize the SDK.
2. Set the request options.
   For more information on the definitions of SecretId, SecretKey, Bucket and other terms and how to obtain them, please see COS Glossary.
   - APPID is an account ID assigned by the system after the application for a Tencent cloud account is submitted.
   - Both access_key_id and access_key_secret are account API keys.
   - endpoint is the COS access domain name, which can be viewed on the **Available Regions for COS** page of Tencent Cloud.
     For example, the endpoint of Guangzhou region is cos.ap-guangzhou.myqcloud.com.
3. Set the required parameters for the API.
4. Call the SDK API to initiate the request and obtain the response result.

Creating Bucket

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_acl_e cos_acl = COS_ACL_PRIVATE;
cos_string_t bucket;
cos_table_t *resp_headers = NULL;

/* Re-create a new memory pool. The second parameter is NULL, which indicates it is not inherited from other me
mory pools */
cos_pool_create(&p, NULL);

/* Create and initialize options. This parameter contains global configuration information such as endpoint, acces
s_key_id, acces_key_secret, is_cname, and curl parameters.
* The memory of the options is assigned by the pool. After the pool is released, the memory of the options is also
released. There is no need to release the memory separately.
*/
```

```
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);

/* Set appid, endpoint, access_key_id, acces_key_secret, is_cname, curl parameters and other configuration inform
ation */
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
/* The bucket name entered must be in a format of {name}-{appid} */
cos_str_set(&bucket, TEST_BUCKET_NAME);

/* Call API to create a bucket*/
s = cos_create_bucket(options, &bucket, cos_acl, &resp_headers);
if (cos_status_is_ok(s)) {
printf("create bucket succeeded\n");
} else {
printf("create bucket failed\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

Uploading files

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_string_t bucket;
cos_string_t object;
cos_string_t file;
cos_table_t *resp_headers = NULL;

/* Re-create a new memory pool. The second parameter is NULL, which indicates it is not inherited from other me
mory pools */
cos_pool_create(&p, NULL);

/* Create and initialize options. This parameter contains global configuration information such as endpoint, acces
s_key_id, acces_key_secret, is_cname, and curl parameters.
* The memory of the options is assigned by the pool. After the pool is released, the memory of the options is also
released. There is no need to release the memory separately.
*/
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);
```

```
/*Set appid, endpoint, access_key_id, acces_key_secret, is_cname, curl parameters and other configuration informa
tion */
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
/* The bucket name entered must be in a format of {name}-{appid} */
cos_str_set(&bucket, TEST_BUCKET_NAME);

/*Call API to upload file */
cos_str_set(&file, TEST_DOWNLOAD_NAME);
cos_str_set(&object, TEST_OBJECT_NAME);
s = cos_put_object_from_file(options, &bucket, &object, &file, NULL, &resp_headers);
if (cos_status_is_ok(s)) {
printf("put object succeeded\n");
} else {
printf("put object failed\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

Downloading files

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_string_t bucket;
cos_string_t object;
cos_string_t file;
cos_table_t *resp_headers = NULL;

/* Re-create a new memory pool. The second parameter is NULL, which indicates it is not inherited from other me
mory pools */
cos_pool_create(&p, NULL);

/* Create and initialize options. This parameter contains global configuration information such as endpoint, acces
s_key_id, acces_key_secret, is_cname, and curl parameters.
* The memory of the options is assigned by the pool. After the pool is released, the memory of the options is also
released. There is no need to release the memory separately.
*/
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);

/* Set appid, endpoint, access_key_id, acces_key_secret, is_cname, curl parameters and other configuration inform
ation */
```

```
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
/*The bucket name entered must be in a format of {name}-{appid}*/
cos_str_set(&bucket, TEST_BUCKET_NAME);

/* Call API to download file */
cos_str_set(&file, TEST_DOWNLOAD_NAME);
cos_str_set(&object, TEST_OBJECT_NAME);
s = cos_get_object_to_file(options, &bucket, &object, NULL, NULL, &file, &resp_headers);
if (cos_status_is_ok(s)) {
printf("get object succeeded\n");
} else {
printf("get object failed\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

API Documentation

Last updated：2018-07-25 18:03:01

XML C SDK operations of COS service return the result corresponding to each API calling, including response code, error code, error description, etc. See the exception types at the end of the document.

> For more information on the definitions of SecretId, SecretKey, Bucket and other terms and how to obtain them, please see COS Glossary.

The following describes how to use each API in the SDK. For the sake of brevity, subsequent examples only illustrate how to use the API rather than how to handle exceptions.

```
cos_status_t *s = NULL;
s = cos_put_object_from_file(options, &bucket, &object, &file, &headers, &resp_headers);
if (!s && !cos_status_is_ok(s)) {
// Output logs for exceptions and handle exceptions as needed
cos_warn_log("failed to put object from file", buf);
if (s->error_code) cos_warn_log("status->error_code: %s", s->error_code);
if (s->error_msg) cos_warn_log("status->error_msg: %s", s->error_msg);
if (s->req_id) cos_warn_log("status->req_id: %s", s->req_id);
}
```

Bucket Operations

Put Bucket

Feature description

This API (Put Bucket) is used to create a Bucket under the specified account.

Method prototype

```
cos_status_t *cos_create_bucket(const cos_request_options_t *options,
const cos_string_t *bucket,
cos_acl_e cos_acl,
cos_table_t **resp_headers);
```

Parameters

Parameter Name	Description	Type
options	COS request option	Struct

Parameter Name	Description	Type
bucket	Bucket name, which must be in a format of {name}-{appid}	String
cos_acl	Allows users to define permissions. Valid values: COS_ACL_PRIVATE(0), COS_ACL_PUBLIC_READ(1) and COS_ACL_PUBLIC_READ_WRITE(2). Default: COS_ACL_PRIVATE(0)	Enum
resp_headers	Returns HTTP response header	Struct

Returned result

Returned Result	Description	Type
code	Error code	Int
error_code	Error description	String
error_msg	Error message	String
req_id	Request message ID	String

Example

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_acl_e cos_acl = COS_ACL_PRIVATE;
cos_string_t bucket;
cos_table_t *resp_headers = NULL;

//create memory pool
cos_pool_create(&p, NULL);

//init request options
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
cos_str_set(&bucket, TEST_BUCKET_NAME);

//create bucket
```

```
s = cos_create_bucket(options, &bucket, cos_acl, &resp_headers);
if (cos_status_is_ok(s)) {
printf("create bucket succeeded\n");
} else {
printf("create bucket failed\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

Delete Bucket

Feature description

This API (Delete Bucket) is used to delete a Bucket under the specified account. The Bucket must be empty before it can be deleted.

Method prototype

```
cos_status_t *cos_delete_bucket(const cos_request_options_t *options,
const cos_string_t *bucket,
cos_table_t **resp_headers);
```

Parameters

Parameter Name	Description	Type
options	COS request option	Struct
bucket	Bucket name, which must be in a format of {name}-{appid}	String
resp_headers	Returns HTTP response header	Struct

Returned result

Returned Result	Description	Type
code	Error code	Int
error_code	Error description	String
error_msg	Error message	String
req_id	Request message ID	String

Example

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
```

```
cos_request_options_t *options = NULL;
cos_string_t bucket;
cos_table_t *resp_headers = NULL;

//create memory pool
cos_pool_create(&p, NULL);

//init request options
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
cos_str_set(&bucket, TEST_BUCKET_NAME);

//delete bucket
s = cos_delete_bucket(options, &bucket, &resp_headers);
if (cos_status_is_ok(s)) {
printf("create bucket succeeded\n");
} else {
printf("create bucket failed\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

Get Bucket

Feature description

This API (Get Bucket) is equivalent to List Object. It is used to list some or all of the Objects under the Bucket. Read permission is required to initiate this request.

Method prototype

```
cos_status_t *cos_list_object(const cos_request_options_t *options,
const cos_string_t *bucket,
cos_list_object_params_t *params,
cos_table_t **resp_headers);
```

Parameters

Parameter Name	Description	Type
options	COS request option	Struct

Parameter Name	Description	Type
bucket	Bucket name, which must be in a format of {name}-{appid}	String
params	Parameter for the Get Bucket operation	Struct
encoding_type	The encoding method of the returned value	String
prefix	Indicates the prefix match, which is used to specify the prefix address of the returned file	String
marker	Entries are listed using UTF-8 binary order by default, starting from the marker	String
delimiter	Parameter for the Get Bucket operation	String
max_ret	Maximum number of entries returned at a time. Default is 1,000	Struct
truncated	Indicates whether the returned entry is truncated. Value: 'true' or 'false'	Boolean
next_marker	If the returned entry is truncated, NextMarker represents the starting point of the next entry	String
object_list	Lists the object information returned through the Get Bucket operation	Struct
key	The name of the Object returned through the Get Bucket operation	Struct
last_modified	The time when the Object returned through the Get Bucket operation was last modified	Struct
etag	The SHA-1 algorithm check value of the Object returned through the Get Bucket operation	Struct
size	The size of the Object returned through the Get Bucket operation (in bytes)	Struct
owner_id	The Object owner UID returned through the Get Bucket operation	Struct
storage_class	The storage level of the Object returned through the Get Bucket operation	Struct
common_prefix_list	The same paths between Prefix and delimiter are grouped as the same type and defined as Common Prefix	Struct
resp_headers	Returns HTTP response header	Struct

Returned result

Returned Result	Description	Type
code	Error code	Int
error_code	Error description	String
error_msg	Error message	String
req_id	Request message ID	String

Returned Result	Description	Type

Example

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_string_t bucket;
cos_table_t *resp_headers = NULL;

//create memory pool
cos_pool_create(&p, NULL);

//init request options
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
cos_str_set(&bucket, TEST_BUCKET_NAME);

//get bucket(list object)
cos_list_object_params_t *list_params = NULL;
cos_list_object_content_t *content = NULL;
list_params = cos_create_list_object_params(p);
s = cos_list_object(options, &bucket, list_params, &resp_headers);
if (cos_status_is_ok(s)) {
printf("list object succeeded\n");
cos_list_for_each_entry(cos_list_object_content_t, content, &list_params->object_list, node) {
key = printf("%.*s\n", content->key.len, content->key.data);
}
} else {
printf("list object failed\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

Put Bucket ACL

Feature description

Put Bucket ACL is used to write the Bucket ACL. You can import ACL information by using Header: "x-cos-acl", "x-cos-grant-read", "x-cos-grant-write", "x-cos-grant-full-control".

Method prototype

```
cos_status_t *cos_put_bucket_acl(const cos_request_options_t *options,
const cos_string_t *bucket,
cos_acl_e cos_acl,
const cos_string_t *grant_read,
const cos_string_t *grant_write,
const cos_string_t *grant_full_ctrl,
cos_table_t **resp_headers);
```

Parameters

Parameter Name	Description	Type
options	COS request option	Struct
bucket	Bucket name, which must be in a format of {name}-{appid}	String
cos_acl	Allows users to define permissions. Valid values: COS_ACL_PRIVATE(0),COS_ACL_PUBLIC_READ(1),COS_ACL_PUBLIC_READ_WRITE(2). Default: COS_ACL_PRIVATE(0)	Enum
grant_read	Users to whom the read permission is granted	String
grant_write	Users to whom the write permission is granted	String
grant_full_ctrl	Users to whom both read and write permissions are granted	String
resp_headers	Returns HTTP response header	Struct

Returned result

Returned Result	Description	Type
code	Error code	Int
error_code	Error description	String
error_msg	Error message	String
req_id	Request message ID	String

Example

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_string_t bucket;
cos_table_t *resp_headers = NULL;
```

```
//create memory pool
cos_pool_create(&p, NULL);

//init request options
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
cos_str_set(&bucket, TEST_BUCKET_NAME);

//put bucket acl
cos_string_t read;
cos_str_set(&read, "id=\"qcs::cam::uin/12345:uin/12345\", id=\"qcs::cam::uin/45678:uin/45678\"");
s = cos_put_bucket_acl(options, &bucket, cos_acl, &read, NULL, NULL, &resp_headers);
if (cos_status_is_ok(s)) {
printf("put bucket acl succeeded\n");
} else {
printf("put bucket acl failed\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

Get Bucket ACL

Feature description

This API (Get Bucket ACL) is used to obtain the ACL (Access Control List) of a Bucket. Only the Bucket owner has the access to this API.

Method prototype

```
cos_status_t *cos_get_bucket_acl(const cos_request_options_t *options,
const cos_string_t *bucket,
cos_acl_params_t *acl_param,
cos_table_t **resp_headers)
```

Parameters

Parameter Name	Description	Type
options	COS request option	Struct

Parameter Name	Description	Type
bucket	Bucket name, which must be in a format of {name}-{appid}	String
acl_param	Parameter for the Get Bucket ACL operation	Struct
owner_id	The Bucket owner ID returned through the Get Bucket ACL operation	String
owner_name	The Bucket owner name returned through the Get Bucket ACL operation	String
object_list	Information of authorized user and permissions returned through the Get Bucket ACL operation	Struct
type	Authorized account type returned through the Get Bucket ACL operation	String
id	Authorized user ID returned through the Get Bucket ACL operation	String
name	Authorized user name returned through the Get Bucket ACL operation	String
permission	Information of authorized user and permissions returned through the Get Bucket ACL operation	String
resp_headers	Returns HTTP response header	Struct

Returned result

Returned Result	Description	Type
code	Error code	Int
error_code	Error description	String
error_msg	Error message	String
req_id	Request message ID	String

Example

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_string_t bucket;
cos_table_t *resp_headers = NULL;

//create memory pool
cos_pool_create(&p, NULL);

//init request options
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
```

```
init_test_config(options->config, is_cname);
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
cos_str_set(&bucket, TEST_BUCKET_NAME);

//get bucket acl
cos_acl_params_t *acl_params = NULL;
acl_params = cos_create_acl_params(p);
s = cos_get_bucket_acl(options, &bucket, acl_params, &resp_headers);
if (cos_status_is_ok(s)) {
printf("get bucket acl succeeded\n");
printf("acl owner id:%s, name:%s\n", acl_params->owner_id.data, acl_params->owner_name.data);
cos_acl_grantee_content_t *acl_content = NULL;
cos_list_for_each_entry(cos_acl_grantee_content_t, acl_content, &acl_params->grantee_list, node) {
printf("acl grantee type:%s, id:%s, name:%s, permission:%s\n", acl_content->type.data, acl_content->id.data, acl_co
ntent->name.data, acl_content->permission.data);
}
} else {
printf("get bucket acl failed\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

Put Bucket Lifecycle

Feature description

This API (Put Bucket Lifecycle) is used to write Bucket lifecycle rules.

Method prototype

```
cos_status_t *cos_put_bucket_lifecycle(const cos_request_options_t *options,
const cos_string_t *bucket,
cos_list_t *lifecycle_rule_list,
cos_table_t **resp_headers);
```

Parameters

Parameter Name	Description	Type
options	COS request option	Struct
bucket	Bucket name, which must be in a format of {name}-{appid}	String

Parameter Name	Description	Type
lifecycle_rule_list	Parameter for the Put Bucket Lifecycle operation	Struct
id	Lifecycle rule ID	String
prefix	Specifies the prefix to which the rule applies.	String
status	Indicates whether the rule is enabled. Enumerated values: Enabled, Disabled.	String
expire	Rule expiration attribute	Struct
days	Indicates the number of days before the deletion operation is performed.	Int
date	Indicates when the deletion operation is performed.	String
transition	Rule transition attribute, which indicates when the Object is transited to Standard_IA	Struct
days	Indicates the number of days before the transition operation is performed.	Int
date	Indicates when the transition operation is performed.	String
storage_class	Specifies the target storage class to which the object is transited. Enumerated values: Standard_IA.	String
abort	Sets the maximum length of time allowed for a multipart upload.	Struct
days	Indicates the number of days within which the upload must be completed after the multipart upload starts.	Int
resp_headers	Returns HTTP response header	Struct

Returned result

Returned Result	Description	Type
code	Error code	Int
error_code	Error description	String
error_msg	Error message	String
req_id	Request message ID	String

Example

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
```

```
cos_string_t bucket;
cos_table_t *resp_headers = NULL;

//create memory pool
cos_pool_create(&p, NULL);

//init request options
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
cos_str_set(&bucket, TEST_BUCKET_NAME);

//put bucket lifecycle
cos_list_t rule_list;
cos_list_init(&rule_list);
cos_lifecycle_rule_content_t *rule_content = NULL;

rule_content = cos_create_lifecycle_rule_content(p);
cos_str_set(&rule_content->id, "testrule1");
cos_str_set(&rule_content->prefix, "abc/");
cos_str_set(&rule_content->status, "Enabled");
rule_content->expire.days = 60;
cos_list_add_tail(&rule_content->node, &rule_list);

rule_content = cos_create_lifecycle_rule_content(p);
cos_str_set(&rule_content->id, "testrule2");
cos_str_set(&rule_content->prefix, "efg/");
cos_str_set(&rule_content->status, "Disabled");
cos_str_set(&rule_content->transition.storage_class, "Standard_IA");
rule_content->transition.days = 30;
cos_list_add_tail(&rule_content->node, &rule_list);

rule_content = cos_create_lifecycle_rule_content(p);
cos_str_set(&rule_content->id, "testrule3");
cos_str_set(&rule_content->prefix, "xxx/");
cos_str_set(&rule_content->status, "Enabled");
rule_content->abort.days = 1;
cos_list_add_tail(&rule_content->node, &rule_list);

s = cos_put_bucket_lifecycle(options, &bucket, &rule_list, &resp_headers);
if (cos_status_is_ok(s)) {
printf("put bucket lifecycle succeeded\n");
} else {
printf("put bucket lifecycle failed\n");
}
```

```
//destroy memory pool
cos_pool_destroy(p);
```

Get Bucket Lifecycle

Feature description

This API (Get Bucket Lifecycle) is used to obtain Bucket lifecycle rules.

Method prototype

```
cos_status_t *cos_get_bucket_lifecycle(const cos_request_options_t *options,
const cos_string_t *bucket,
cos_list_t *lifecycle_rule_list,
cos_table_t **resp_headers);
```

Parameters

Parameter Name	Description	Type
options	COS request option	Struct
bucket	Bucket name, which must be in a format of {name}-{appid}	String
lifecycle_rule_list	Parameter for the Get Bucket Lifecycle operation	Struct
id	Lifecycle rule ID	String
prefix	The prefix to which the rule applies	String
status	Indicates whether the rule is enabled. Enumerated values: Enabled, Disabled.	String
expire	Rule expiration attribute	Struct
days	Indicates the number of days before the deletion operation is performed.	Int
date	Indicates when the deletion operation is performed.	String
transition	Rule transition attribute, which indicates when the Object is transited to Standard_IA	Struct
days	Indicates the number of days before the transition operation is performed.	Int
date	Indicates when the transition operation is performed.	String
storage_class	Specifies the target storage class to which the object is transited. Enumerated values: Standard_IA.	String
abort	Sets the maximum time length allowed for a multipart upload.	Struct

Parameter Name	Description	Type
days	Indicates the number of days within which the upload must be completed after the multipart upload starts.	Int
resp_headers	Returns HTTP response header	Struct

Returned result

Returned Result	Description	Type
code	Error code	Int
error_code	Error description	String
error_msg	Error message	String
req_id	Request message ID	String

Example

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_string_t bucket;
cos_table_t *resp_headers = NULL;

//create memory pool
cos_pool_create(&p, NULL);

//init request options
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
cos_str_set(&bucket, TEST_BUCKET_NAME);

//get bucket lifecycle
cos_list_t rule_list_ret;
cos_list_init(&rule_list_ret);
s = cos_get_bucket_lifecycle(options, &bucket, &rule_list_ret, &resp_headers);
if (cos_status_is_ok(s)) {
printf("get bucket lifecycle succeeded\n");
```

```
} else {
printf("get bucket lifecycle failed\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

Delete Bucket Lifecycle

Feature description

This API (Delete Bucket Lifecycle) is used to delete Bucket lifecycle rules.

Method prototype

```
cos_status_t *cos_delete_bucket_lifecycle(const cos_request_options_t *options,
const cos_string_t *bucket,
cos_table_t **resp_headers);
```

Parameters

Parameter Name	Description	Type
options	COS request option	Struct
bucket	Bucket name, which must be in a format of {name}-{appid}	String
resp_headers	Returns HTTP response header	Struct

Returned result

Returned Result	Description	Type
code	Error code	Int
error_code	Error description	String
error_msg	Error message	String
req_id	Request message ID	String

Example

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_string_t bucket;
cos_table_t *resp_headers = NULL;
```

```
//create memory pool
cos_pool_create(&p, NULL);

//init request options
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
cos_str_set(&bucket, TEST_BUCKET_NAME);

//delete bucket lifecycle
s = cos_delete_bucket_lifecycle(options, &bucket, &resp_headers);
if (cos_status_is_ok(s)) {
printf("delete bucket lifecycle succeeded\n");
} else {
printf("delete bucket lifecycle failed\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

Put Bucket CORS

Feature description

This API (Put Bucket CORS) is used to set the cross-origin resource sharing permission of the Bucket.

Method prototype

```
cos_status_t *cos_put_bucket_cors(const cos_request_options_t *options,
const cos_string_t *bucket,
cos_list_t *cors_rule_list,
cos_table_t **resp_headers);
```

Parameters

Parameter Name	Description	Type
options	COS request option	Struct
bucket	Bucket name, which must be in a format of {name}-{appid}	String
cors_rule_list	Parameter for the Put Bucket CORS operation	Struct
id	Sets rule ID	String

Parameter Name	Description	Type
allowed_origin	Allowed access sources. The wildcard `*` is supported	String
allowed_method	Allowed HTTP operations. Enumerated values: GET, PUT, HEAD, POST, DELETE	String
allowed_header	When an OPTIONS request is sent, notifies the server about which custom HTTP request headers are allowed for subsequent requests. Wildcard "*" is supported.	String
expose_header	Sets the custom header information that can be received by the browser from the server end	String
max_age_seconds	Sets the validity period of the results obtained by OPTIONS	Int
resp_headers	Returns HTTP response header	Struct

Returned result

Returned Result	Description	Type
code	Error code	Int
error_code	Error description	String
error_msg	Error message	String
req_id	Request message ID	String

Example

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_string_t bucket;
cos_table_t *resp_headers = NULL;

//create memory pool
cos_pool_create(&p, NULL);

//init request options
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
cos_str_set(&bucket, TEST_BUCKET_NAME);
```

```
//put bucket cors
cos_list_t rule_list;
cos_list_init(&rule_list);
cos_cors_rule_content_t *rule_content = NULL;

rule_content = cos_create_cors_rule_content(p);
cos_str_set(&rule_content->id, "testrule1");
cos_str_set(&rule_content->allowed_origin, "http://www.qq1.com");
cos_str_set(&rule_content->allowed_method, "GET");
cos_str_set(&rule_content->allowed_header, "*");
cos_str_set(&rule_content->expose_header, "xxx");
rule_content->max_age_seconds = 3600;
cos_list_add_tail(&rule_content->node, &rule_list);

rule_content = cos_create_cors_rule_content(p);
cos_str_set(&rule_content->id, "testrule2");
cos_str_set(&rule_content->allowed_origin, "http://www.qq2.com");
cos_str_set(&rule_content->allowed_method, "GET");
cos_str_set(&rule_content->allowed_header, "*");
cos_str_set(&rule_content->expose_header, "yyy");
rule_content->max_age_seconds = 7200;
cos_list_add_tail(&rule_content->node, &rule_list);

rule_content = cos_create_cors_rule_content(p);
cos_str_set(&rule_content->id, "testrule3");
cos_str_set(&rule_content->allowed_origin, "http://www.qq3.com");
cos_str_set(&rule_content->allowed_method, "GET");
cos_str_set(&rule_content->allowed_header, "*");
cos_str_set(&rule_content->expose_header, "zzz");
rule_content->max_age_seconds = 60;
cos_list_add_tail(&rule_content->node, &rule_list);

//put cors
s = cos_put_bucket_cors(options, &bucket, &rule_list, &resp_headers);
if (cos_status_is_ok(s)) {
printf("put bucket cors succeeded\n");
} else {
printf("put bucket cors failed\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

Get Bucket CORS

Feature description

This API (Get Bucket CORS) is used to obtain the cross-origin resource sharing permission configuration of the Bucket.

Method prototype

```
cos_status_t *cos_get_bucket_cors(const cos_request_options_t *options,
const cos_string_t *bucket,
cos_list_t *cors_rule_list,
cos_table_t **resp_headers);
```

Parameters

Parameter Name	Description	Type
options	COS request option	Struct
bucket	Bucket name, which must be in a format of {name}-{appid}	String
cors_rule_list	Parameter for the Get Bucket CORS operation	Struct
id	Sets rule ID	String
allowed_origin	Allowed access sources. The wildcard `*` is supported	String
allowed_method	Allowed HTTP operations. Enumerated values: GET, PUT, HEAD, POST, DELETE	String
allowed_header	When an OPTIONS request is sent, notifies the server about which custom HTTP request headers are allowed for subsequent requests. Wildcard "*" is supported.	String
expose_header	Sets the custom header information that can be received by the browser from the server end	String
max_age_seconds	Sets the validity period of the results obtained by OPTIONS	Int
resp_headers	Returns HTTP response header	Struct

Returned result

Returned Result	Description	Type
code	Error code	Int
error_code	Error description	String
error_msg	Error message	String
req_id	Request message ID	String

Example

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_string_t bucket;
cos_table_t *resp_headers = NULL;
```

```
//create memory pool
cos_pool_create(&p, NULL);

//init request options
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
cos_str_set(&bucket, TEST_BUCKET_NAME);

//get bucket cors
cos_list_t rule_list_ret;
cos_list_init(&rule_list_ret);
s = cos_get_bucket_cors(options, &bucket, &rule_list_ret, &resp_headers);
if (cos_status_is_ok(s)) {
printf("get bucket cors succeeded\n");
cos_cors_rule_content_t *content = NULL;
cos_list_for_each_entry(cos_cors_rule_content_t, content, &rule_list_ret, node) {
printf("cors id:%s, allowed_origin:%s, allowed_method:%s, allowed_header:%s, expose_header:%s, max_age_second
s:%d\n",
content->id.data, content->allowed_origin.data, content->allowed_method.data, content->allowed_header.data,
content->expose_header.data, content->max_age_seconds);
}
} else {
printf("get bucket cors failed\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

Delete Bucket CORS

Feature description

This API (Delete Bucket CORS) is used to delete the cross-origin resource sharing permission configuration of the Bucket.

Method prototype

```
cos_status_t *cos_delete_bucket_cors(const cos_request_options_t *options,
const cos_string_t *bucket,
cos_table_t **resp_headers);
```

Parameters

Parameter Name	Description	Type
options	COS request option	Struct
bucket	Bucket name, which must be in a format of {name}-{appid}	String
resp_headers	Returns HTTP response header	Struct

Returned result

Returned Result	Description	Type
code	Error code	Int
error_code	Error description	String
error_msg	Error message	String
req_id	Request message ID	String

Example

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_string_t bucket;
cos_table_t *resp_headers = NULL;

//create memory pool
cos_pool_create(&p, NULL);

//init request options
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
cos_str_set(&bucket, TEST_BUCKET_NAME);

//delete bucket cors
s = cos_delete_bucket_cors(options, &bucket, &resp_headers);
if (cos_status_is_ok(s)) {
printf("delete bucket cors succeeded\n");
} else {
printf("delete bucket cors failed\n");
```

```
}

//destroy memory pool
cos_pool_destroy(p);
```

Put Bucket Versioning

Feature description

This API (Put Bucket Versioning) is used to enable or suspend the version control of the Bucket.

Method prototype

```
cos_status_t *cos_put_bucket_versioning(const cos_request_options_t *options,
const cos_string_t *bucket,
cos_versioning_content_t *versioning,
cos_table_t **resp_headers);
```

Parameters

Parameter Name	Description	Type
options	COS request option	Struct
bucket	Bucket name, which must be in a format of {name}-{appid}	String
versioning	Parameter for the Put Bucket Versioning operation	Struct
status	Indicates whether version is enabled. Enumerated values: Suspended, Enabled.	String
resp_headers	Returns HTTP response header	Struct

Returned result

Returned Result	Description	Type
code	Error code	Int
error_code	Error description	String
error_msg	Error message	String
req_id	Request message ID	String

Example

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_string_t bucket;
```

```
cos_table_t *resp_headers = NULL;

//create memory pool
cos_pool_create(&p, NULL);

//init request options
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
cos_str_set(&bucket, TEST_BUCKET_NAME);

//put bucket versioning
cos_versioning_content_t *versioning = NULL;
versioning = cos_create_versioning_content(p);
cos_str_set(&versioning->status, "Enabled");
s = cos_put_bucket_versioning(options, &bucket, versioning, &resp_headers);
if (cos_status_is_ok(s)) {
printf("put bucket versioning succeeded\n");
} else {
printf("put bucket versioning failed\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

Get Bucket Versioning

Feature description

This API (Get Bucket Versioning) is used to obtain the information on the version control of the Bucket.

Method prototype

```
cos_status_t *cos_get_bucket_versioning(const cos_request_options_t *options,
const cos_string_t *bucket,
cos_versioning_content_t *versioning,
cos_table_t **resp_headers);
```

Parameters

Parameter Name	Description	Type
options	COS request option	Struct

Parameter Name	Description	Type
bucket	Bucket name, which must be in a format of {name}-{appid}	String
versioning	Parameter for the Get Bucket Versioning operation	Struct
status	Indicates whether version is enabled. Enumerated values: Suspended, Enabled.	String
resp_headers	Returns HTTP response header	Struct

Returned result

Returned Result	Description	Type
code	Error code	Int
error_code	Error description	String
error_msg	Error message	String
req_id	Request message ID	String

Example

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_string_t bucket;
cos_table_t *resp_headers = NULL;

//create memory pool
cos_pool_create(&p, NULL);

//init request options
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
cos_str_set(&bucket, TEST_BUCKET_NAME);

//get bucket versioning
cos_versioning_content_t *versioning = NULL;
versioning = cos_create_versioning_content(p);
s = cos_get_bucket_versioning(options, &bucket, versioning, &resp_headers);
if (cos_status_is_ok(s)) {
```

```
printf("put bucket versioning succeeded\n");
printf("bucket versioning status: %s\n", versioning->status.data);
} else {
printf("put bucket versioning failed\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

Put Bucket Replication

Feature description

This API (Put Bucket Replication) is used to add replication configuration to the bucket for which versioning is enabled. If the bucket already has a replication configuration, the request will replace the existing configuration.

Method prototype

```
cos_status_t *cos_put_bucket_replication(const cos_request_options_t *options,
const cos_string_t *bucket,
cos_replication_params_t *replication_param,
cos_table_t **resp_headers);
```

Parameters

Parameter Name	Description	Type
options	COS request option	Struct
bucket	Bucket name, which must be in a format of {name}-{appid}	String
replication_param	Parameter for the Put Bucket Replication operation	Struct
role	Operator's account information	String
rule_list	replication configuration information	Struct
id	Indicates the name of a specific Rule.	String
status	Indicates whether the rule is enabled. Enumerated values: Enabled, Disabled.	String
prefix	Prefix match. Prefixes cannot overlap, otherwise an error is returned.	String
dst_bucket	Destination bucket ID. Format: resource identifier qcs:id/0:cos:[Region]:appid/[APPID]:[Bucketname].	String
storage_class	The storage class of Object. Enumerated values: Standard, Standard_IA. Default value: the original bucket class.	String
resp_headers	Returns HTTP response header	Struct

Returned result

Returned Result	Description	Type
code	Error code	Int
error_code	Error description	String
error_msg	Error message	String
req_id	Request message ID	String

Example

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_string_t bucket;
cos_table_t *resp_headers = NULL;

//create memory pool
cos_pool_create(&p, NULL);

//init request options
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
cos_str_set(&bucket, TEST_BUCKET_NAME);

//put bucket replication
cos_replication_params_t *replication_param = NULL;
replication_param = cos_create_replication_params(p);
cos_str_set(&replication_param->role, "qcs::cam::uin/100000616666:uin/100000616666");

cos_replication_rule_content_t *rule = NULL;
rule = cos_create_replication_rule_content(p);
cos_str_set(&rule->id, "Rule_01");
cos_str_set(&rule->status, "Enabled");
cos_str_set(&rule->prefix, "test1");
cos_str_set(&rule->dst_bucket, "qcs:id/0:cos:cn-east:appid/1253686666:replicationtest");
cos_list_add_tail(&rule->node, &replication_param->rule_list);

rule = cos_create_replication_rule_content(p);
cos_str_set(&rule->id, "Rule_02");
```

```
cos_str_set(&rule->status, "Disabled");
cos_str_set(&rule->prefix, "test2");
cos_str_set(&rule->storage_class, "Standard_IA");
cos_str_set(&rule->dst_bucket, "qcs:id/0:cos:cn-east:appid/1253686666:replicationtest");
cos_list_add_tail(&rule->node, &replication_param->rule_list);

rule = cos_create_replication_rule_content(p);
cos_str_set(&rule->id, "Rule_03");
cos_str_set(&rule->status, "Enabled");
cos_str_set(&rule->prefix, "test3");
cos_str_set(&rule->storage_class, "Standard_IA");
cos_str_set(&rule->dst_bucket, "qcs:id/0:cos:cn-east:appid/1253686666:replicationtest");
cos_list_add_tail(&rule->node, &replication_param->rule_list);

s = cos_put_bucket_replication(options, &bucket, replication_param, &resp_headers);
if (cos_status_is_ok(s)) {
printf("put bucket replication succeeded\n");
} else {
printf("put bucket replication failed\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

Get Bucket Replication

Feature description

This API (Get Bucket Replication) is used to read the cross-origin replication configuration information in a bucket.

Method prototype

```
cos_status_t *cos_get_bucket_replication(const cos_request_options_t *options,
const cos_string_t *bucket,
cos_replication_params_t *replication_param,
cos_table_t **resp_headers);
```

Parameters

Parameter Name	Description	Type
options	COS request option	Struct
bucket	Bucket name, which must be in a format of {name}-{appid}	String
replication_param	Parameter for the Get Bucket Replication operation	Struct
role	Operator's account information	String
rule_list	replication configuration information	Struct

Parameter Name	Description	Type
id	Indicates the name of a specific Rule.	String
status	Indicates whether the rule is enabled. Enumerated values: Enabled, Disabled.	String
prefix	Prefix match. Prefixes cannot overlap, otherwise an error is returned.	String
dst_bucket	Destination bucket ID. Format: resource identifier qcs:id/0:cos:[Region]:appid/[APPID]:[Bucketname].	String
storage_class	The storage class of Object. Enumerated values: Standard, Standard_IA. Default value: the original bucket class.	String
resp_headers	Returns HTTP response header	Struct

Returned result

Returned Result	Description	Type
code	Error code	Int
error_code	Error description	String
error_msg	Error message	String
req_id	Request message ID	String

Example

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_string_t bucket;
cos_table_t *resp_headers = NULL;

//create memory pool
cos_pool_create(&p, NULL);

//init request options
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
cos_str_set(&bucket, TEST_BUCKET_NAME);
```

```
//get bucket replication
cos_replication_params_t *replication_param2 = NULL;
replication_param2 = cos_create_replication_params(p);
s = cos_get_bucket_replication(options, &bucket, replication_param2, &resp_headers);

if (cos_status_is_ok(s)) {
printf("get bucket replication succeeded\n");
printf("ReplicationConfiguration role: %s\n", replication_param2->role.data);
cos_replication_rule_content_t *content = NULL;
cos_list_for_each_entry(cos_replication_rule_content_t, content, &replication_param2->rule_list, node) {
printf("ReplicationConfiguration rule, id:%s, status:%s, prefix:%s, dst_bucket:%s, storage_class:%s\n",
content->id.data, content->status.data, content->prefix.data, content->dst_bucket.data, content->storage_class.d
ata);
}
} else {
printf("get bucket replication failed\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

Delete Bucket Replication

Feature description

This API (Delete Bucket Replication) is used to delete the cross-origin replication configuration in the Bucket.

Method prototype

```
cos_status_t *cos_delete_bucket_replication(const cos_request_options_t *options,
const cos_string_t *bucket,
cos_table_t **resp_headers);
```

Parameters

Parameter Name	Description	Type
options	COS request option	Struct
bucket	Bucket name, which must be in a format of {name}-{appid}	String
resp_headers	Returns HTTP response header	Struct

Returned result

Returned Result	Description	Type
code	Error code	Int

Returned Result	Description	Type
error_code	Error description	String
error_msg	Error message	String
req_id	Request message ID	String

Example

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_string_t bucket;
cos_table_t *resp_headers = NULL;

//create memory pool
cos_pool_create(&p, NULL);

//init request options
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
cos_str_set(&bucket, TEST_BUCKET_NAME);

//delete bucket cors
s = cos_delete_bucket_replication(options, &bucket, &resp_headers);
if (cos_status_is_ok(s)) {
printf("delete bucket replication succeeded\n");
} else {
printf("delete bucket replication failed\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

Object Operations

Get Object

Feature description

This API (Delete Object) is used to download a file (Object) locally. This operation requires that the user have the read permission for the target Object or the read permission for the target Object be available for everyone (public-read).

Method prototype

```
cos_status_t *cos_get_object_to_file(const cos_request_options_t *options,
const cos_string_t *bucket,
const cos_string_t *object,
cos_table_t *headers,
cos_table_t *params,
cos_string_t *filename,
cos_table_t **resp_headers);
```

Parameters

Parameter Name	Description	Type
options	COS request option	Struct
bucket	Bucket name, which must be in a format of {name}-{appid}	String
object	Object name	String
headers	Additional COS request header	Struct
params	Parameter for the COS request operation	Struct
filename	The file name of an Object stored locally	String
resp_headers	Returns HTTP response header	Struct

Returned result

Returned Result	Description	Type
code	Error code	Int
error_code	Error description	String
error_msg	Error message	String
req_id	Request message ID	String

Example

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_string_t bucket;
cos_string_t object;
```

```
cos_string_t file;
cos_table_t *resp_headers = NULL;

//create memory pool
cos_pool_create(&p, NULL);

//init request options
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
cos_str_set(&bucket, TEST_BUCKET_NAME);

//get object
cos_str_set(&file, TEST_DOWNLOAD_NAME);
cos_str_set(&object, TEST_OBJECT_NAME);
s = cos_get_object_to_file(options, &bucket, &object, NULL, NULL, &file, &resp_headers);
if (cos_status_is_ok(s)) {
printf("get object succeeded\n");
} else {
printf("get object failed\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

Head Object

Feature description

This API (Head Object) is used to get the metadata of an Object. It has the same permissions as Get Object.

Method prototype

```
cos_status_t *cos_head_object(const cos_request_options_t *options,
const cos_string_t *bucket,
const cos_string_t *object,
cos_table_t *headers,
cos_table_t **resp_headers);
```

Parameters

Parameter Name	Description	Type
options	COS request option	Struct

Parameter Name	Description	Type
bucket	Bucket name, which must be in a format of {name}-{appid}	String
object	Object name	String
headers	Additional COS request header	Struct
resp_headers	Returns HTTP response header	Struct

Returned result

Returned Result	Description	Type
code	Error code	Int
error_code	Error description	String
error_msg	Error message	String
req_id	Request message ID	String

Example

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_string_t bucket;
cos_string_t object;
cos_table_t *resp_headers = NULL;

//create memory pool
cos_pool_create(&p, NULL);

//init request options
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
cos_str_set(&bucket, TEST_BUCKET_NAME);

//head object
cos_str_set(&object, TEST_OBJECT_NAME);
s = cos_head_object(options, &bucket, &object, NULL, &resp_headers);
if (cos_status_is_ok(s)) {
```

```
printf("head object succeeded\n");
} else {
printf("head object failed\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

Put Object

Feature description

This API (Put Object) is used to upload a file (Object) to the specified Bucket.

Method prototype

```
cos_status_t *cos_put_object_from_file(const cos_request_options_t *options,
const cos_string_t *bucket,
const cos_string_t *object,
const cos_string_t *filename,
cos_table_t *headers,
cos_table_t **resp_headers);
```

Parameters

Parameter Name	Description	Type
options	COS request option	Struct
bucket	Bucket name, which must be in a format of {name}-{appid}	String
object	Object name	String
filename	The file name of an Object stored locally	String
headers	Additional COS request header	Struct
resp_headers	Returns HTTP response header	Struct

Returned result

Returned Result	Description	Type
code	Error code	Int
error_code	Error description	String
error_msg	Error message	String
req_id	Request message ID	String

Example

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_string_t bucket;
cos_string_t object;
cos_string_t file;
cos_table_t *resp_headers = NULL;

//create memory pool
cos_pool_create(&p, NULL);

//init request options
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
cos_str_set(&bucket, TEST_BUCKET_NAME);

//put object
cos_str_set(&file, TEST_DOWNLOAD_NAME);
cos_str_set(&object, TEST_OBJECT_NAME);
s = cos_put_object_from_file(options, &bucket, &object, &file, NULL, &resp_headers);
if (cos_status_is_ok(s)) {
printf("put object succeeded\n");
} else {
printf("put object failed\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

Delete Object

Feature description

This API (Delete Object) is used to delete a file (Object).

Method prototype

```
cos_status_t *cos_delete_object(const cos_request_options_t *options,
const cos_string_t *bucket,
```

```
const cos_string_t *object,
cos_table_t **resp_headers);
```

Parameters

Parameter Name	Description	Type
options	COS request option	Struct
bucket	Bucket name, which must be in a format of {name}-{appid}	String
object	Object name	String
resp_headers	Returns HTTP response header	Struct

Returned result

Returned Result	Description	Type
code	Error code	Int
error_code	Error description	String
error_msg	Error message	String
req_id	Request message ID	String

Example

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_string_t bucket;
cos_string_t object;
cos_table_t *resp_headers = NULL;

//create memory pool
cos_pool_create(&p, NULL);

//init request options
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
cos_str_set(&bucket, TEST_BUCKET_NAME);
```

```
//delete object
cos_str_set(&object, TEST_OBJECT_NAME);
s = cos_delete_object(options, &bucket, &object, &resp_headers);
if (cos_status_is_ok(s)) {
printf("delete object succeeded\n");
} else {
printf("delete object failed\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

Delete Multiple Object

Feature description

This API (Delete Multiple Object) is used for batch deletion of files. A maximum of 1000 files are allowed to be deleted at a time. COS provides two modes for returned results: Verbose and Quiet. Verbose mode will return the result of deletion of each Object, while Quiet mode only returns the information of the Objects with an error.

Method prototype

```
cos_status_t *cos_delete_objects(const cos_request_options_t *options,
const cos_string_t *bucket,
cos_list_t *object_list,
int is_quiet,
cos_table_t **resp_headers,
cos_list_t *deleted_object_list);
```

Parameters

Parameter Name	Description	Type
options	COS request option	Struct
bucket	Bucket name, which must be in a format of {name}-{appid}	String
object_list	List of objects to be deleted	Struct
key	Name of object to be deleted	String
is_quiet	Indicates whether the Quiet mode is enabled. True(1): Quiet mode is enabled; False(0): Verbose mode is enabled. Default is False(0).	Boolean
resp_headers	Returns HTTP response header	Struct
deleted_object_list	List of deleted objects	Struct

Returned result

Returned Result	Description	Type
code	Error code	Int
error_code	Error description	String
error_msg	Error message	String
req_id	Request message ID	String

Example

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_string_t bucket;
cos_string_t object;
cos_table_t *resp_headers = NULL;

//create memory pool
cos_pool_create(&p, NULL);

//init request options
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
cos_str_set(&bucket, TEST_BUCKET_NAME);

//init delete object list
char *object_name1 = TEST_OBJECT_NAME1;
char *object_name2 = TEST_OBJECT_NAME2;
cos_object_key_t *content1 = NULL;
cos_object_key_t *content2 = NULL;
cos_list_t object_list;
cos_list_t deleted_object_list;
cos_list_init(&object_list);
cos_list_init(&deleted_object_list);
content1 = cos_create_cos_object_key(p);
cos_str_set(&content1->key, object_name1);
cos_list_add_tail(&content1->node, &object_list);
content2 = cos_create_cos_object_key(p);
cos_str_set(&content2->key, object_name2);
```

```
cos_list_add_tail(&content2->node, &object_list);

//delete objects
int is_quiet = COS_TRUE;
cos_str_set(&object, TEST_OBJECT_NAME);
s = cos_delete_objects(options, &bucket, &object_list, is_quiet, &resp_headers, &deleted_object_list);
if (cos_status_is_ok(s)) {
printf("delete objects succeeded\n");
} else {
printf("delete objects failed\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

Put Object ACL

Feature description

This API (Put Object ACL) is used to write the ACL configuration of an Object in the Bucket. You can import ACL information by using Header: "x-cos-acl", "x-cos-grant-read", "x-cos-grant-write", "x-cos-grant-full-control".

Method prototype

```
cos_status_t *cos_put_object_acl(const cos_request_options_t *options,
const cos_string_t *bucket,
const cos_string_t *object,
cos_acl_e cos_acl,
const cos_string_t *grant_read,
const cos_string_t *grant_write,
const cos_string_t *grant_full_ctrl,
cos_table_t **resp_headers);
```

Parameters

Parameter Name	Description	Type
options	COS request option	Struct
bucket	Bucket name, which must be in a format of {name}-{appid}	String
object	Object name	String
cos_acl	Allow users to define permissions. Valid values: COS_ACL_PRIVATE(0), COS_ACL_PUBLIC_READ(1) and COS_ACL_PUBLIC_READ_WRITE(2). Default: COS_ACL_PRIVATE(0)	Enum
grant_read	Users to whom the read permission is granted	String
grant_write	Users to whom the write permission is granted	String

Parameter Name	Description	Type
grant_full_ctrl	Users to whom both read and write permissions are granted	String
resp_headers	Returns HTTP response header	Struct

Returned result

Returned Result	Description	Type
code	Error code	Int
error_code	Error description	String
error_msg	Error message	String
req_id	Request message ID	String

Example

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_string_t bucket;
cos_string_t object;
cos_table_t *resp_headers = NULL;

//create memory pool
cos_pool_create(&p, NULL);

//init request options
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
cos_str_set(&bucket, TEST_BUCKET_NAME);

//put object acl
cos_str_set(&object, TEST_OBJECT_NAME);
cos_string_t read;
cos_str_set(&read, "id=\"qcs::cam::uin/12345:uin/12345\", id=\"qcs::cam::uin/45678:uin/45678\"");
s = cos_put_object_acl(options, &bucket, &object, cos_acl, &read, NULL, NULL, &resp_headers);
if (cos_status_is_ok(s)) {
```

```
printf("put object acl succeeded\n");
} else {
printf("put object acl failed\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

Get Object ACL

Feature description

This API (Get Object ACL) is used to obtain access permission of an Object under a Bucket. Only the Bucket owner has the access to this API.

Method prototype

```
cos_status_t *cos_get_object_acl(const cos_request_options_t *options,
const cos_string_t *bucket,
const cos_string_t *object,
cos_acl_params_t *acl_param,
cos_table_t **resp_headers)
```

Parameters

Parameter Name	Description	Type
options	COS request option	Struct
bucket	Bucket name, which must be in a format of {name}-{appid}	String
object	Object name	String
acl_param	Parameter for the Get Object ACL operation	Struct
owner_id	The Bucket owner ID returned through the Get Object ACL operation	String
owner_name	The Bucket owner name returned through the Get Object ACL operation	String
object_list	Information of authorized user and permissions returned through the Get Object ACL operation	Struct
type	Authorized account type returned through the Get Object ACL operation	String
id	Authorized user ID returned through the Get Object ACL operation	String
name	Authorized user name returned through the Get Object ACL operation	String
permission	Information of authorized user and permissions returned through the Get Object ACL operation	String

Parameter Name	Description	Type
resp_headers	Returns HTTP response header	Struct

Returned result

Returned Result	Description	Type
code	Error code	Int
error_code	Error description	String
error_msg	Error message	String
req_id	Request message ID	String

Example

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_string_t bucket;
cos_string_t object;
cos_table_t *resp_headers = NULL;

//create memory pool
cos_pool_create(&p, NULL);

//init request options
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
cos_str_set(&bucket, TEST_BUCKET_NAME);

//get object acl
cos_acl_params_t *acl_params2 = NULL;
acl_params2 = cos_create_acl_params(p);
s = cos_get_object_acl(options, &bucket, &object, acl_params2, &resp_headers);
if (cos_status_is_ok(s)) {
printf("get object acl succeeded\n");
printf("acl owner id:%s, name:%s\n", acl_params2->owner_id.data, acl_params2->owner_name.data);
acl_content = NULL;
```

```
cos_list_for_each_entry(cos_acl_grantee_content_t, acl_content, &acl_params2->grantee_list, node) {
printf("acl grantee id:%s, name:%s, permission:%s\n", acl_content->id.data, acl_content->name.data, acl_content->permission.data);
}
} else {
printf("get object acl failed\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

Put Object Copy

Feature description

This API (Put Object Copy) is used to copy a file from source path to the destination path.

Method prototype

```
cos_status_t *cos_copy_object(const cos_request_options_t *options,
const cos_string_t *copy_source,
const cos_string_t *dest_bucket,
const cos_string_t *dest_object,
cos_table_t *headers,
cos_copy_object_params_t *copy_object_param,
cos_table_t **resp_headers);
```

Parameters

Parameter Name	Description	Type
options	COS request option	Struct
copy_source	Path of the source file	String
dest_bucket	Destination bucket name, which must be in a format of {name}-{appid}	String
dest_object	Destination object name	String
headers	Additional COS request header	Struct
copy_object_param	Parameter for the Put Object Copy operation	Struct
etag	MD5 algorithm check value for the returned file	String
last_modify	Returns the last modification time of the file in GMT format	String
resp_headers	Returns HTTP response header	Struct

Returned result

Returned Result	Description	Type
code	Error code	Int
error_code	Error description	String
error_msg	Error message	String
req_id	Request message ID	String

Example

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_string_t bucket;
cos_string_t object;
cos_table_t *resp_headers = NULL;

//create memory pool
cos_pool_create(&p, NULL);

//init request options
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
cos_str_set(&bucket, TEST_BUCKET_NAME);

//put object copy
cos_str_set(&object, TEST_OBJECT_NAME);
cos_string_t copy_source;
cos_str_set(&copy_source, TEST_COPY_SRC);
cos_copy_object_params_t *params = NULL;
params = cos_create_copy_object_params(p);
s = cos_copy_object(options, &copy_source, &bucket, &object, NULL, params, &resp_headers);
if (cos_status_is_ok(s)) {
printf("put object copy succeeded\n");
} else {
printf("put object copy failed\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

Multipart Upload Operations

Initiate Multipart Upload

Feature description

This API (Initiate Multipart Upload) is used to initialize multipart upload. After the request is executed successfully, Upload ID is returned for the subsequent Upload Part requests.

Method prototype

```
cos_status_t *cos_init_multipart_upload(const cos_request_options_t *options,
const cos_string_t *bucket,
const cos_string_t *object,
cos_string_t *upload_id,
cos_table_t *headers,
cos_table_t **resp_headers);
```

Parameters

Parameter Name	Description	Type
options	COS request option	Struct
bucket	Bucket name, which must be in a format of {name}-{appid}	String
object	Object name	String
upload_id	Upload ID returned through the operation	String
headers	Additional COS request header	Struct
resp_headers	Returns HTTP response header	Struct

Returned result

Returned Result	Description	Type
code	Error code	Int
error_code	Error description	String
error_msg	Error message	String
req_id	Request message ID	String

Example

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_string_t bucket;
cos_string_t object;
cos_string_t file;
cos_table_t *resp_headers = NULL;

//create memory pool
cos_pool_create(&p, NULL);

//init request options
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
cos_str_set(&bucket, TEST_BUCKET_NAME);

//init mulitipart upload
cos_str_set(&object, TEST_OBJECT_NAME);
s = cos_init_multipart_upload(options, &bucket, &object,
&upload_id, headers, &resp_headers);
if (cos_status_is_ok(s)) {
printf("init multipart upload succeeded\n");
} else {
printf("init multipart upload failed\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

Upload Part

Feature description

This API (Upload Part) is used to implement multipart upload after initialization. A file can be split into 10000 chunks at most (minimum is 1) for multipart upload, and the size of each file chunk should be between 1 MB and 5 GB. Parameters partNumber and uploadId are required for Upload Part (partNumber is the file chunk No. Out-of-order upload is supported).

Method prototype

```
cos_status_t *cos_upload_part_from_file(const cos_request_options_t *options,
const cos_string_t *bucket,
const cos_string_t *object,
const cos_string_t *upload_id,
int part_num,
cos_upload_file_t *upload_file,
cos_table_t **resp_headers);
```

Parameters

Parameter Name	Description	Type
options	COS request option	Struct
bucket	Bucket name, which must be in a format of {name}-{appid}	String
object	Object name	String
upload_id	Upload task number	String
part_num	Part number	Int
upload_file	Information on the local file to be uploaded	Struct
resp_headers	Returns HTTP response header	Struct

Returned result

Returned Result	Description	Type
code	Error code	Int
error_code	Error description	String
error_msg	Error message	String
req_id	Request message ID	String

Example

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_string_t bucket;
cos_string_t object;
cos_string_t file;
cos_table_t *resp_headers = NULL;
int part_num = 1;
int64_t pos = 0;
```

```
int64_t file_length = 0;

//create memory pool
cos_pool_create(&p, NULL);

//init request options
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
cos_str_set(&bucket, TEST_BUCKET_NAME);

//upload part from file
int res = COSE_OK;
cos_upload_file_t *upload_file = NULL;
cos_file_buf_t *fb = cos_create_file_buf(p);
res = cos_open_file_for_all_read(p, TEST_MULTIPART_FILE, fb);
if (res != COSE_OK) {
cos_error_log("Open read file fail, filename:%s\n", TEST_MULTIPART_FILE);
return;
}
file_length = fb->file_last;
apr_file_close(fb->file);
while(pos < file_length) {
upload_file = cos_create_upload_file(p);
cos_str_set(&upload_file->filename, TEST_MULTIPART_FILE);
upload_file->file_pos = pos;
pos += 2 * 1024 * 1024;
upload_file->file_last = pos < file_length ? pos : file_length; //2MB
s = cos_upload_part_from_file(options, &bucket, &object, &upload_id,
part_num++, upload_file, &resp_headers);

if (cos_status_is_ok(s)) {
printf("upload part succeeded\n");
} else {
printf("upload part failed\n");
}
}

//destroy memory pool
cos_pool_destroy(p);
```

Complete Multipart Upload

Feature description

Complete Multipart Upload is used to complete the entire multipart upload. After you have uploaded all the file chunks using Upload Parts, you can use this API to complete the upload. When using this API, you need to provide the PartNumber and ETag for every chunk in Body, to verify the accuracy of chunks.

Method prototype

```
cos_status_t *cos_complete_multipart_upload(const cos_request_options_t *options,
const cos_string_t *bucket,
const cos_string_t *object,
const cos_string_t *upload_id,
cos_list_t *part_list,
cos_table_t *headers,
cos_table_t **resp_headers);
```

Parameters

Parameter Name	Description	Type	
options	COS request option		Struct
bucket	Bucket name, which must be in a format of {name}-{appid}	String	
object	Object name	String	
upload_id	Upload task number	String	
part_list	Parameter for the Complete Multipart Upload operation	Struct	
part_number	Part number	String	
etag	ETag value of a part, which is sha1 check value. It must be enclosed in double quotes, such as "3a0f1fd698c235af9cf098cb74aa25bc"	String	
headers	Additional COS request header	Struct	
resp_headers	Returns HTTP response header	Struct	

Returned result

Returned Result	Description	Type
code	Error Codes	Int
error_code	Error description	String
error_msg	Error message	String
req_id	Request message ID	String

Example

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_string_t bucket;
cos_string_t object;
cos_string_t file;
cos_table_t *resp_headers = NULL;
cos_list_part_content_t *part_content = NULL;
cos_complete_part_content_t *complete_part_content = NULL;
int part_num = 1;
int64_t pos = 0;
int64_t file_length = 0;

//create memory pool
cos_pool_create(&p, NULL);

//init request options
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
cos_str_set(&bucket, TEST_BUCKET_NAME);

//list part
params = cos_create_list_upload_part_params(p);
params->max_ret = 1000;
cos_list_init(&complete_part_list);
s = cos_list_upload_part(options, &bucket, &object, &upload_id,
params, &resp_headers);

if (cos_status_is_ok(s)) {
printf("List multipart succeeded\n");
} else {
printf("List multipart failed\n");
cos_pool_destroy(p);
return;
}

cos_list_for_each_entry(cos_list_part_content_t, part_content, &params->part_list, node) {
complete_part_content = cos_create_complete_part_content(p);
cos_str_set(&complete_part_content->part_number, part_content->part_number.data);
cos_str_set(&complete_part_content->etag, part_content->etag.data);
cos_list_add_tail(&complete_part_content->node, &complete_part_list);
}
```

```
//complete multipart
s = cos_complete_multipart_upload(options, &bucket, &object, &upload_id,
&complete_part_list, complete_headers, &resp_headers);

if (cos_status_is_ok(s)) {
printf("Complete multipart upload from file succeeded, upload_id:%.*s\n",
upload_id.len, upload_id.data);
} else {
printf("Complete multipart upload from file failed\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

List Parts

Feature description

This API (List Parts) is used to query the uploaded parts in a specific multipart upload process.

Method prototype

```
cos_status_t *cos_list_upload_part(const cos_request_options_t *options,
const cos_string_t *bucket,
const cos_string_t *object,
const cos_string_t *upload_id,
cos_list_upload_part_params_t *params,
cos_table_t **resp_headers);
```

Parameters

Parameter Name	Description	Type
options	COS request option	Struct
bucket	Bucket name, which must be in a format of {name}-{appid}	String
object	Object name	String
upload_id	Upload task number	String
params	Parameter for the List Parts operation	Struct
part_number_marker	Entries are listed using UTF-8 binary order by default, starting from the marker	String
encoding_type	The encoding method of the returned value	String
max_ret	Maximum number of entries returned at a time. Default is 1,000	String

Parameter Name	Description	Type
truncated	Indicates whether the returned entry is truncated. Value: 'true' or 'false'	Boolean
next_part_number_marker	If the returned entry is truncated, NextMarker represents the starting point of the next entry	String
part_list	Information on completed part	Struct
part_number	Part number	String
size	Part size (in bytes)	String
etag	SHA-1 algorithm check value for the part	String
last_modified	The time when the part was last modified	String
resp_headers	Returns HTTP response header	Struct

Returned result

Returned Result	Description	Type
code	Error code	Int
error_code	Error description	String
error_msg	Error message	String
req_id	Request message ID	String

Example

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_string_t bucket;
cos_string_t object;
cos_string_t file;
cos_table_t *resp_headers = NULL;
cos_list_part_content_t *part_content = NULL;
cos_complete_part_content_t *complete_part_content = NULL;
int part_num = 1;
int64_t pos = 0;
int64_t file_length = 0;

//create memory pool
cos_pool_create(&p, NULL);

//init request options
```

```
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);
cos_str_set(&options->config->endpoint, TEST_COS_ENDPOINT);
cos_str_set(&options->config->access_key_id, TEST_ACCESS_KEY_ID);
cos_str_set(&options->config->access_key_secret, TEST_ACCESS_KEY_SECRET);
cos_str_set(&options->config->appid, TEST_APPID);
options->config->is_cname = is_cname;
options->ctl = cos_http_controller_create(options->pool, 0);
cos_str_set(&bucket, TEST_BUCKET_NAME);

//list part
params = cos_create_list_upload_part_params(p);
params->max_ret = 1000;
cos_list_init(&complete_part_list);
s = cos_list_upload_part(options, &bucket, &object, &upload_id,
params, &resp_headers);

if (cos_status_is_ok(s)) {
printf("List multipart succeeded\n");
} else {
printf("List multipart failed\n");
cos_pool_destroy(p);
return;
}

cos_list_for_each_entry(cos_list_part_content_t, part_content, &params->part_list, node) {
complete_part_content = cos_create_complete_part_content(p);
cos_str_set(&complete_part_content->part_number, part_content->part_number.data);
cos_str_set(&complete_part_content->etag, part_content->etag.data);
cos_list_add_tail(&complete_part_content->node, &complete_part_list);
}

//complete multipart
s = cos_complete_multipart_upload(options, &bucket, &object, &upload_id,
&complete_part_list, complete_headers, &resp_headers);

if (cos_status_is_ok(s)) {
printf("Complete multipart upload from file succeeded, upload_id:%.*s\n",
upload_id.len, upload_id.data);
} else {
printf("Complete multipart upload from file failed\n");
}

//destroy memory pool
cos_pool_destroy(p);
```

Abort Multipart Upload

Feature description

This API (Abort Multipart Upload) is used to abort a multipart upload operation and delete uploaded file chunks. When Abort Multipart Upload is called, a failure is returned for any request that is using Upload Parts.

Method prototype

```
cos_status_t *cos_abort_multipart_upload(const cos_request_options_t *options,
const cos_string_t *bucket,
const cos_string_t *object,
cos_string_t *upload_id,
cos_table_t **resp_headers);
```

Parameters

Parameter Name	Description	Type
options	COS request option	Struct
bucket	Bucket name, which must be in a format of {name}-{appid}	String
object	Object name	String
upload_id	Upload task number	String
resp_headers	Returns HTTP response header	Struct

Returned result

Returned Result	Description	Type
code	Error code	Int
error_code	Error description	String
error_msg	Error message	String
req_id	Request message ID	String

Example

```
cos_pool_t *p = NULL;
cos_string_t bucket;
cos_string_t object;
int is_cname = 0;
cos_table_t *headers = NULL;
cos_table_t *resp_headers = NULL;
cos_request_options_t *options = NULL;
cos_string_t upload_id;
cos_status_t *s = NULL;

//create memory pool & init request options
```

```
cos_pool_create(&p, NULL);
options = cos_request_options_create(p);
init_test_request_options(options, is_cname);
headers = cos_table_make(p, 1);
cos_str_set(&bucket, TEST_BUCKET_NAME);
cos_str_set(&object, TEST_MULTIPART_OBJECT);

//init multipart upload
s = cos_init_multipart_upload(options, &bucket, &object,
&upload_id, headers, &resp_headers);

if (cos_status_is_ok(s)) {
printf("Init multipart upload succeeded, upload_id:%.*s\n",
upload_id.len, upload_id.data);
} else {
printf("Init multipart upload failed\n");
cos_pool_destroy(p);
return;
}

//abort multipart upload
s = cos_abort_multipart_upload(options, &bucket, &object, &upload_id,
&resp_headers);

if (cos_status_is_ok(s)) {
printf("Abort multipart upload succeeded, upload_id::%.*s\n",
upload_id.len, upload_id.data);
} else {
printf("Abort multipart upload failed\n");
}

cos_pool_destroy(p);
```

List Multipart Uploads

Feature description

This API (List Multiparts Uploads) is used to query multipart upload operations that are still in process. Up to 1000 such operations can be listed each time.

Method prototype

```
cos_status_t *cos_list_multipart_upload(const cos_request_options_t *options,
const cos_string_t *bucket,
cos_list_multipart_upload_params_t *params,
cos_table_t **resp_headers);
```

Parameters

Parameter Name	Description	Type

Parameter Name	Description	Type
options	COS request option	Struct
bucket	Bucket name, which must be in a format of {name}-{appid}	String
params	Parameter for the List Multipart Uploads operation	Struct
encoding_type	The encoding method of the returned value	String
prefix	Indicates the prefix match, which is used to specify the prefix address of the returned file	String
upload_id_marker	If the returned entry is truncated, NextMarker represents the starting point of the next entry	String
delimiter	Delimiter is a sign. If Prefix exists, the same paths between Prefix and delimiter are grouped as the same type and defined as Common Prefix, and then all Common Prefixes are listed. If Prefix does not exist, the listing process starts from the beginning of the path.	String
max_ret	Maximum number of entries returned at a time. Default is 1,000	String
key_marker	Used together with upload-id-marker. If upload-id-marker is not specified, entries whose ObjectNames are in front of key-marker (according to alphabetical order) will be listed. If upload-id-marker is specified, entries whose ObjectNames are in front of key-marker (according to alphabetical order) will be listed, and entries whose ObjectNames are equal to key-marker and UploadIDs are in front of upload-id-marker (according to alphabetical order) will also be listed.	String
upload_id_marker	Used together with key-marker. If key-marker is not specified, upload-id-marker will be ignored. If key-marker is specified, entries whose ObjectNames are in front of key-marker (according to alphabetical order) will be listed, and entries whose ObjectNames are equal to key-marker and UploadIDs are in front of upload-id-marker (according to alphabetical order) will also be listed.	String
truncated	Indicates whether the returned entry is truncated. Value: 'true' or 'false'	Boolean
next_key_marker	If the returned entry is truncated, NextMarker represents the starting point of the next entry	String
next_upload_id_marker	If the returned entry is truncated, NextMarker represents the starting point of the next entry	String
upload_list	Provides information on multipart upload	Struct
key	Object name	String
upload_id	The ID of current multipart upload	String

Parameter Name	Description	Type
initiated	Indicates the start time of current multipart upload.	String
resp_headers	Returns HTTP response header	Struct

```
typedef struct {
cos_list_t node;
cos_string_t key;
cos_string_t upload_id;
cos_string_t initiated;
} cos_list_multipart_upload_content_t;
```

Returned result

Returned Result	Description	Type
code	Error code	Int
error_code	Error description	String
error_msg	Error message	String
req_id	Request message ID	String

Example

```
cos_pool_t *p = NULL;
cos_string_t bucket;
int is_cname = 0;
cos_table_t *resp_headers = NULL;
cos_request_options_t *options = NULL;
cos_status_t *s = NULL;
cos_list_multipart_upload_params_t *list_multipart_params = NULL;

//create memory pool & init request options
cos_pool_create(&p, NULL);
options = cos_request_options_create(p);
init_test_request_options(options, is_cname);
cos_str_set(&bucket, TEST_BUCKET_NAME);

//list multipart upload
list_multipart_params = cos_create_list_multipart_upload_params(p);
list_multipart_params->max_ret = 999;
s = cos_list_multipart_upload(options, &bucket, list_multipart_params, &resp_headers);
log_status(s);

cos_pool_destroy(p);
```

Exception Description

If the SDK operation fails, the result information can be found in the cos_status_t structure returned by the API.

The cos_status_t structure comprises of the following:

cos_status_t Member	Description	Type
code	Status code of the response. 4xx represents the request failure caused by the client, and 5xx represents the failure caused by the server exception For more information, please see COS Error Message	Int
error_code	Error Code returned by body when request fails. For more information, please see COS Error Message	String
error_msg	Error Message returned by body when request fails. For more information, please see COS Error Message	String
req_id	Request ID, which is used to identify the user's unique request	String

C++ SDK
Getting Started

Last updated：2018-07-20 17:57:21

Preparations for Development

Obtaining SDK

Download XML C++ SDK resources of COS service from: XML C++ SDK Download.
Download Demo from: XML C++ SDK Demo.

Environment

Dependent static library: jsoncpp boost_system boost_thread Poco (in lib folder);
Dependent dynamic library: ssl crypto rt z (installation is required).
JsonCpp libraries and header files are available in the SDK. If you want to install it yourself, compile the libraries installed in the following steps and replace the relevant library and header files in the SDK. If the above libraries are already installed on the system, you can also delete the relevant libraries and header files in the SDK. The installation steps are as follows:

1. Install the CMake tool.

```
yum install -y gcc gcc-c++ make automake
//Install the required packages such as GCC (skip this step if they have been installed)
yum install -y wget

// The cmake version should be greater than 3.5
wget https://cmake.org/files/v3.5/cmake-3.5.2.tar.gz
tar -zxvf cmake-3.5.2.tar.gz
cd cmake-3.5.2.tar.gz
./bootstrap --prefix=/usr
gmake
gmake install
```

2. Install Boost libraries and header files.

```
wget http://sourceforge.net/projects/boost/files/boost/1.54.0/boost_1_54_0.tar.gz
tar -xzvf boost_1_54_0.tar.gz
cd boost_1_54_0
./bootstrap.sh --prefix=/usr/local
./b2 install --with=all
# Boost libraries are installed in the directory /usr/local/lib
```

3. Install OpenSSL.

```
yum install openssl openssl-devel
```

Or

```
wget https://www.openssl.org/source/openssl-1.0.1t.tar.gz
tar -xzvf ./openssl-1.0.1t.tar.gz
cd openssl-1.0.1t/
./config --prefix=/usr/local/ssl --openssldir=/usr/local/ssl

cd /usr/local/
ln -s ssl openssl
echo "/usr/local/openssl/lib" >> /etc/ld.so.conf
ldconfig

# Add header/lib file search path (can be written to ~/.bashrc)
LIBRARY_PATH=/usr/local/ssl/lib/:$LIBRARY_PATH
CPLUS_INCLUDE_PATH=/usr/local/ssl/include/:$CPLUS_INCLUDE_PATH
```

4. Obtain Poco libraries and header files from Poco official website and install them (download the complete version).

```
./configure --omit=Data/ODBC,Data/MySQL
make
make install
```

5. Obtain APP ID, SecretID, and SecretKey from the console.

For more information on definitions of SecretId, SecretKey, Bucket and other terms and how to obtain them, please see COS Glossary.

You can specify the path to the local Boost header files by modifying the following statement in `CMakeList.txt` file:

```
SET(BOOST_HEADER_DIR "/root/boost_1_61_0")
```

Configuring SDK

Directly download the source code provided on GitHub in the step Obtaining SDK to integrate it into your development environment. Execute the following command:

```
cd ${cos-cpp-sdk}
mkdir -p build
cd build
cmake ..
make
```

There are examples of common APIs in cos_demo.cpp. The generated cos_demo can be run directly, and the name of the generated static library is libcossdk.a. The generated libcossdk.a is placed in the lib path of your own project, and the include directory is copied to the include path of your project.

Getting Started

Initialization

```
"SecretId":"**********************************", // Use "AccessKey" for the SDK configuration file earlier than V5.4.3.
"SecretKey":"**********************************",
"Region":"cn-north",
// COS region (ensure it is correct)
"SignExpiredTime":360,
// Signature timeout (in sec)
"ConnectTimeoutInms":6000,
// connect timeout (in ms)
"HttpTimeoutInms":60000,
// http timeout (in ms)
"UploadPartSize":1048576,
// Size of each part of file in multipart upload; it ranges from 1 MB to 5 GB. Default is 1 MB.
"UploadThreadPoolSize":5,
// Thread pool size for uploading a single file in multiple parts
"DownloadSliceSize":4194304,
// Size of each part of file in multipart download
"DownloadThreadPoolSize":5,
// Thread pool size for downloading a single file
"AsynThreadPoolSize":2,
// Thread pool size for asynchronous upload and download
"LogoutType":1,
// Log output type. 0: Do not output; 1: Output to screen; 2: Output to syslog
"LogLevel":3
// Log level. 1: ERR; 2: WARN; 3: INFO; 4: DBG
```

Creating Bucket

```
#include "cos_api.h"
#include "cos_sys_config.h"
#include "cos_defines.h"

int main(int argc, char *argv[]) {
// 1. Specify the path to configuration file and initialize CosConfig
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);

// 2. Construct a request to create a Bucket
std::string bucket_name = "cpp_sdk_v5-123456789"; // Bucket Name
qcloud_cos::PutBucketReq req(bucket_name);
```

```
qcloud_cos::PutBucketResp resp;

// 3. Call the API for creating Bucket
qcloud_cos::CosResult result = cos.PutBucket(req, &resp);

// 4. Process the result of the call
if (result.IsSucc()) {
// Created successfully
} else {
// Failed to create the Bucket. You can call the CosResult's member function to output error information, such as r
equestID, etc.
std::cout << "ErrorInfo=" << result.GetErrorInfo() << std::endl;
std::cout << "HttpStatus=" << result.GetHttpStatus() << std::endl;
std::cout << "ErrorCode=" << result.GetErrorCode() << std::endl;
std::cout << "ErrorMsg=" << result.GetErrorMsg() << std::endl;
std::cout << "ResourceAddr=" << result.GetResourceAddr() << std::endl;
std::cout << "XCosRequestId=" << result.GetXCosRequestId() << std::endl;
std::cout << "XCosTraceId=" << result.GetXCosTraceId() << std::endl;
}
}
```

Uploading files

```
#include "cos_api.h"
#include "cos_sys_config.h"
#include "cos_defines.h"

int main(int argc, char *argv[]) {
// 1. Specify the path to configuration file and initialize CosConfig
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);

// 2. Construct a request to upload a file
std::string bucket_name = "cpp_sdk_v5-123456789"; // Name of destination Bucket for upload
std::string object_name = "object_name"; // object_name is the object key, which is the unique identifier of the ob
ject in the bucket. For example, in the object's access domain name bucket1-1250000000.cos.ap-guangzhou.myq
cloud.com/doc1/pic1.jpg, the object key is doc1/pic1.jpg.
// The local file path is required in the constructor of the request
qcloud_cos::PutObjectByFileReq req(bucket_name, object_name, "/path/to/local/file");
req.SetXCosStorageClass("STANDARD_IA"); // Call Set method to set metadata or ACL
qcloud_cos::PutObjectByFileResp resp;

// 3. Call the API for uploading file
qcloud_cos::CosResult result = cos.PutObject(req, &resp);

// 4. Process the result of the call
if (result.IsSucc()) {
// File is uploaded successfully
} else {
// Failed to upload the file. You can call the CosResult's member function to output error information, such as req
```

```
uestID, etc.
std::cout << "ErrorInfo=" << result.GetErrorInfo() << std::endl;
std::cout << "HttpStatus=" << result.GetHttpStatus() << std::endl;
std::cout << "ErrorCode=" << result.GetErrorCode() << std::endl;
std::cout << "ErrorMsg=" << result.GetErrorMsg() << std::endl;
std::cout << "ResourceAddr=" << result.GetResourceAddr() << std::endl;
std::cout << "XCosRequestId=" << result.GetXCosRequestId() << std::endl;
std::cout << "XCosTraceId=" << result.GetXCosTraceId() << std::endl;
}
}
```

Downloading files

```
#include "cos_api.h"
#include "cos_sys_config.h"
#include "cos_defines.h"

int main(int argc, char *argv[]) {
// 1. Specify the path to configuration file and initialize CosConfig
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);

// 2. Construct a request to create a Bucket
std::string bucket_name = "cpp_sdk_v5-123456789"; // Name of destination Bucket for upload
std::string object_name = "object_name"; // object_name is the object key, which is the unique identifier of the ob
ject in the bucket. For example, in the object's access domain name bucket1-1250000000.cos.ap-guangzhou.myq
cloud.com/doc1/pic1.jpg, the object key is doc1/pic1.jpg.
std::string local_path = "/tmp/object_name";
// Parameters appid, bucketname and object, as well as the local path (including file name) are required for the r
equest
qcloud_cos::GetObjectByFileReq req(bucket_name, object_name, local_path);
qcloud_cos::GetObjectByFileResp resp;

// 3. Call the API for creating Bucket
qcloud_cos::CosResult result = cos.GetObject(req, &resp);

// 4. Process the result of the call
if (result.IsSucc()) {
// File is downloaded successfully
} else {
// Failed to download the file. You can call the CosResult's member function to output error information, such as
requestID, etc.
std::cout << "ErrorInfo=" << result.GetErrorInfo() << std::endl;
std::cout << "HttpStatus=" << result.GetHttpStatus() << std::endl;
std::cout << "ErrorCode=" << result.GetErrorCode() << std::endl;
std::cout << "ErrorMsg=" << result.GetErrorMsg() << std::endl;
std::cout << "ResourceAddr=" << result.GetResourceAddr() << std::endl;
std::cout << "XCosRequestId=" << result.GetXCosRequestId() << std::endl;
std::cout << "XCosTraceId=" << result.GetXCosTraceId() << std::endl;
```

```
}
}
```

API Documentation

Last updated：2018-09-21 12:52:57

Generating Signature

Feature of Sign

Generate signature

Method prototype 1

```
static std::string Sign(const std::string& secret_id,
const std::string& secret_key,
const std::string& http_method,
const std::string& in_uri,
const std::map<std::string, std::string>& headers,
const std::map<std::string, std::string>& params);
```

Parameters

Parameter Name	Parameter Description	Type
secret_id	The project identity ID owned by a developer, which is used for identity authentication	String
secret_key	The project identity key owned by a developer	String
http_method	HTTP method, such as POST/GET/HEAD/PUT. It is case-insensitive	String
in_uri	HTTP uri	String
headers	Key-value pair of HTTP header	map<string,string>
params	Key-value pair of HTTP params	map<string,string>

Returned result

A signature is returned, which can be used within the specified validity period (which is set via CosSysConfig and defaults to 60 sec). A returned empty string indicates the generation of signature failed.

Method prototype 2

```
static std::string Sign(const std::string& secret_id,
const std::string& secret_key,
const std::string& http_method,
const std::string& in_uri,
const std::map<std::string, std::string>& headers,
```

```
const std::map<std::string, std::string>& params,
uint64_t start_time_in_s,
uint64_t end_time_in_s);
```

Parameters

Parameter Name	Parameter Description	Type
secret_id	The project identity ID owned by a developer, which is used for identity authentication	String
secret_key	The project identity key owned by a developer	String
http_method	HTTP method, such as POST/GET/HEAD/PUT. It is case-insensitive	String
in_uri	HTTP uri	String
headers	Key-value pair of HTTP header	map <string,string>
params	Key-value pair of HTTP params	map <string,string>
start_time_in_s	Start time of the signature's validity period	uint64_t
end_time_in_s	End time of the signature's validity period	uint64_t

Returned result

- A signature (String) is returned, which can be used within the specified validity period. A returned empty string indicates the generation of signature failed.

Service/Bucket/Object Operations

All Service/Bucket/Object-related method prototypes are shown as follows:

```
CosResult Operator(BaseReq, BaseResp)
```

CosResult

CosResult encapsulates the error code and corresponding error message returned when an error occurs with the request. For more information, please see Error Codes.

> **Note:**
> The CoSResult object is always returned for requests encapsulated in the SDK. After an API call is completed, you can use the IsSucc() member function to determine whether the call is successful.**

Member Functions

Function	Description
bool isSucc()	Returns a Boolean value to indicate whether this call is successful. The CosResult member functions described below only apply when False is returned. If True is returned, the response content can be obtained from OperatorResp.
string GetErrorCode()	Obtains the error code returned by COS to determine the error scenario.
string GetErrorMsg()	Contains details of error.
string GetResourceAddr()	Resource address: Bucket address or Object address.
string GetXCosRequestId()	When a request is sent, the server automatically generates a unique ID for the request. The request-id can be used by COS to quickly locate any problem occurred.
string GetXCosTraceId()	If an error occurs with a request, the server automatically generates a unique ID for the error. The trace-id can be used by COS to quickly locate any problem occurred. If an error occurs with a request, trace-id corresponds to request-id on an one-to-one basis.
string GetErrorInfo()	Obtains the SDK internal error information.
int GetHttpStatus()	Obtains HTTP status code.

BaseReq/BaseResp

BaseReq and BaseResp encapsulate the request and response, respectively. The caller only needs to generate OperatorReq, such as GetBucketReq as described below, based on the type of operation you want to perform, and populate it.
After the function is returned, call BaseResp's member functions to get the request result.

- For Request, you only need to focus on its constructors unless otherwise specified.
- Response for all methods has member functions that can be used to obtain common response headers.
  The common member functions of Response are as follows. For the descriptions of fields, please see Common Response Headers.

```
uint64_t GetContentLength();
std::string GetContentType();
std::string GetEtag();
std::string GetConnection();
std::string GetDate();
std::string GetServer();
std::string GetXCosRequestId();
std::string GetXCosTraceId();
```

Bucket Operations

Get Bucket

Feature description

This API (Get Bucket) is equivalent to List Object. It is used to list some or all of the Objects under the Bucket. Read permission is required to initiate this request. For more information about this API, please see Get Bucket.

Method prototype

```
CosResult GetBucket(const GetBucketReq& req, GetBucketResp* resp);
```

Parameters

Parameter	Description
req	GetBucketReq, the request for GetBucket operation.
resp	GetBucketResp, the response for GetBucket operation.

GetBucketResp provides the following member functions to obtain the content of the response in XML format returned via Get Bucket.

```
std::vector<Content> GetContents();
std::string GetName();
std::string GetPrefix();
std::string GetMarker();
uint64_t GetMaxKeys();
bool IsTruncated();
std::vector<std::string> GetCommonPrefixes();
```

Content is composed as follows:

```
struct Content {
std::string m_key; // Object's Key
std::string m_last_modified; // The time when Object was last modified
std::string m_etag; // The MD-5 algorithm check value of the file
std::string m_size; // File size (in bytes)
std::vector<std::string> m_owner_ids; // Information of the Bucket owner
std::string m_storage_class; // The storage class of Object. Enumerated values: STANDARD, STANDARD_IA
};
```

Example

```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);

std::string bucket_name = "cpp_sdk_v5-123456789";

// The bucket_name is required in the constructor of GetBucketReq
qcloud_cos::GetBucketReq req(bucket_name);
qcloud_cos::GetBucketResp resp;
qcloud_cos::CosResult result = cos.GetBucket(req, &resp);

// The call is successful. Call resp's member functions to obtain the response content
if (result.IsSucc()) {
std::cout << "Name=" << resp.GetName() << std::endl;
std::cout << "Prefix=" << resp.GetPrefix() << std::endl;
std::cout << "Marker=" << resp.GetMarker() << std::endl;
std::cout << "MaxKeys=" << resp.GetMaxKeys() << std::endl;
} else {
std::cout << "ErrorInfo=" << result.GetErrorInfo() << std::endl;
std::cout << "HttpStatus=" << result.GetHttpStatus() << std::endl;
std::cout << "ErrorCode=" << result.GetErrorCode() << std::endl;
std::cout << "ErrorMsg=" << result.GetErrorMsg() << std::endl;
std::cout << "ResourceAddr=" << result.GetResourceAddr() << std::endl;
std::cout << "XCosRequestId=" << result.GetXCosRequestId() << std::endl;
std::cout << "XCosTraceId=" << result.GetXCosTraceId() << std::endl;
}
```

Put Bucket

Feature description

This API (Put Bucket) is used to create a Bucket under specified account. This API does not support anonymous requests. To create a Bucket, you should use a request with Authorization signature. The Bucket creator defaults to the Bucket owner. For more information about this API, please see Put Bucket.

Method prototype

```
CosResult PutBucket(const PutBucketReq& req, PutBucketResp* resp);
```

Parameters

Parameter	Description
req	PutBucketReq, the request for PutBucket operation.
resp	PutBucketResp, the response for PutBucket operation.

PutBucketReq provides the following member functions:

```
// Defines the ACL attribute of Bucket. Valid values: private, public-read-write, public-read
// Default: private
void SetXCosAcl(const std::string& str);

// Grants read permission to the authorized user. Format: x-cos-grant-read: id=" ",id=" ".
// For authorization to a sub-account,id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>"
// For authorization to a root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>"
void SetXCosGrantRead(const std::string& str);

// Grants write permission to the authorized user. Format: x-cos-grant-write: id=" ",id=" "./
// For authorization to a sub-account,id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>",
// For authorization to a root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>"
void SetXCosGrantWrite(const std::string& str);

// Grants read and write permissions to the authorized user. Format: x-cos-grant-full-control: id=" ",id=" ".
// For authorization to a sub-account,id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>",
// For authorization to a root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>"
void SetXCosGrantFullControl(const std::string& str);
```

Example

```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);

std::string bucket_name = "cpp_sdk_v5-123456789";

qcloud_cos::PutBucketReq req(bucket_name);
qcloud_cos::PutBucketResp resp;
qcloud_cos::CosResult result = cos.PutBucket(req, &resp);

// The call is successful. Call resp's member functions to obtain the response content
if (result.IsSucc()) {
// ...
} else {
Failed to create the Bucket. You can call CosResult's member functions to output error information, such as reque
stID, etc.
}
```

Delete Bucket

Feature description

This API (Delete Bucket) is used to delete a Bucket under a specified account. The Bucket must be empty before it can be deleted. The Bucket can be deleted only if its content is removed. For more information about this API, please see Delete Bucket.

Method prototype

```
CosResult DeleteBucket(const DeleteBucketReq& req, DeleteBucketResp* resp);
```

Parameters

Parameter	Description
req	DeleteBucketReq, the request for DeleteBucket operation.
resp	DeletBucketResp, the response for DeletBucket operation.

Example

```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);

std::string bucket_name = "cpp_sdk_v5-123456789";

// The bucket_name is required in the constructor of DeleteBucketReq
qcloud_cos::DeleteBucketReq req(bucket_name);
qcloud_cos::DeleteBucketResp resp;
qcloud_cos::CosResult result = cos.DeleteBucket(req, &resp);

// The call is successful. Call resp's member functions to obtain the response content
if (result.IsSucc()) {
// ...
} else {
// Failed to delete the Bucket. Call CosResult's member functions to output error information, such as requestID, etc.
}
```

Put Bucket Replication

Feature description

This API (Put Bucket Replication) is used to add replication configuration to the bucket for which versioning is enabled. If the bucket already has a replication configuration, the request will replace the existing configuration. For more information about this API, please see Put Bucket Replication.

Method prototype

```
CosResult PutBucketReplication(const DPutBucketReplicationReq& req, PutBucketReplicationResp* resp);
```

Parameters

Parameter	Description
req	PutBucketReplicationReq, the request for PutBucketReplication operation.
resp	PutBucketReplicationResp, the response for PutBucketReplication operation.

```
// Sets the Replication initiator's role. Format: qcs::cam::uin/[UIN]:uin/[Subaccount]
void SetRole(const std::string& role);

// Adds ReplicationRule
void AddReplicationRule(const ReplicationRule& rule);

// Sets ReplicationRules
void SetReplicationRule(const std::vector<ReplicationRule>& rules);
```

ReplicationRule is composed as follows:

```
struct ReplicationRule {
bool m_is_enable; // Whether the Rule takes effect
std::string m_id; // Optional field, used to specify the name of a specific Rule
std::string m_prefix; // Prefix match policy. Prefixes cannot overlap, otherwise an error is returned. This is left empt
y for root directory.
std::string m_dest_bucket; // Identifies destination Bucket. Resource identifier: qcs:id/0:cos:[region]:appid/[AppId]:
[bucketname]
std::string m_dest_storage_class; // Storage class (optional). Enumerated values: Standard, Standard_IA. The origin
al bucket class is used if it is left empty.

ReplicationRule();
ReplicationRule(const std::string& prefix,
const std::string& dest_bucket,
const std::string& storage_class = "",
const std::string& id = "",
bool is_enable = true);
};
```

Example

```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);

std::string bucket_name = "cpp_sdk_v5-123456789";

// The bucket_name is required in the constructor of PutBucketReplicationReq
qcloud_cos::PutBucketReplicationReq req(bucket_name);
req.SetRole("qcs::cam::uin/***:uin/****");
qcloud_cos::ReplicationRule rule("sevenyou_10m", "qcs:id/0:cos:cn-south:appid/***:sevenyousouthtest", "", "RuleId
_01", true);
req.AddReplicationRule(rule)

qcloud_cos::PutBucketReplicationResp resp;
qcloud_cos::CosResult result = cos.PutBucketReplication(req, &resp);

// The call is successful. Call resp's member functions to obtain the response content
if (result.IsSucc()) {
// ...
```

```
} else {
// Failed to set cross-origin replication. Call CosResult's member functions to output error information, such as requestID, etc.
}
```

Get Bucket Replication

Feature description

This API (Get Bucket Replication) is used to read the cross-origin replication configuration information in a bucket. For more information about this API, please see Get Bucket Replication.

Method prototype

```
CosResult GetBucketReplication(const DGetBucketReplicationReq& req, GetBucketReplicationResp* resp);
```

Parameters

Parameter	Description
req	GetBucketReplicationReq, the request for GetBucketReplication operation.
resp	GetBucketReplicationResp, the response for GetBucketReplication operation.

```
// Obtains the Replication initiator's role
std::string GetRole();

// Obtains ReplicationRules. For the definition of ReplicationRule, please see Put Bucket Replication
std::vector<ReplicationRule> GetRules();
```

Example

```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);

std::string bucket_name = "cpp_sdk_v5-123456789";

// The bucket_name is required in the constructor of GetBucketReplicationReq
qcloud_cos::GetBucketReplicationReq req(bucket_name);
qcloud_cos::GetBucketReplicationResp resp;
qcloud_cos::CosResult result = cos.GetBucketReplication(req, &resp);

// The call is successful. Call resp's member functions to obtain the response content
if (result.IsSucc()) {
// ...
} else {
// Failed to obtain cross-region replication configuration. Call CosResult's member functions to output error infor
```

```
mation, such as requestID, etc.
}
```

Delete Bucket Replication

Feature description

This API (Delete Bucket Replication) is used to delete the cross-origin replication configuration in a bucket. For more information about this API, please see Delete Bucket Replication.

Method prototype

```
CosResult DeleteBucketReplication(const DDeleteBucketReplicationReq& req, DeleteBucketReplicationResp* resp);
```

Parameters

Parameter	Description
req	DeleteBucketReplicationReq, the request for DeleteBucketReplication operation.
resp	DeleteBucketReplicationResp, the response for DeleteBucketReplication operation.

Example

```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);

std::string bucket_name = "cpp_sdk_v5-123456789";

// The bucket_name is required in the constructor of DeleteBucketReplicationReq
qcloud_cos::DeleteBucketReplicationReq req(bucket_name);
qcloud_cos::DeleteBucketReplicationResp resp;
qcloud_cos::CosResult result = cos.DeleteBucketReplication(req, &resp);

// The call is successful. Call resp's member functions to obtain the response content
if (result.IsSucc()) {
// ...
} else {
// Failed to delete cross-region replication configuration. Call CosResult's member functions to output error information, such as requestID, etc.
}
```

Put Bucket Lifecycle

Feature description

COS allows you to manage the lifecycle of an Object in Bucket by configuring lifecycle. The lifecycle configuration contains one or more rule sets that will be applied to a group of object rules (each rule defines an operation for COS).

These operations are divided into the following two types:

- Transition: Specify the time when the storage class of an object is changed to another one. For example, you can specify that the storage class of an object is changed to STANDARD_IA (applicable to the objects that are accessed infrequently) 30 days after it is created.
- Expiration: Specify the expiration time of Object. COS will automatically delete the expired objects.
  This API (Put Bucket Lifecycle) is used to create a new lifecycle configuration for a Bucket. If lifecycle has been configured for the Bucket, you can use this API to create a new configuration to overwrite the existing one. For more information about this API, please see Put Bucket Lifecycle.

Method prototype

```
CosResult PutBucketLifecycle(const DPutBucketLifecycleReq& req, PutBucketLifecycleResp* resp);
```

Parameters

Parameter	Description
req	PutBucketLifecycleReq, the request for PutBucketLifecycle operation.
resp	PutBucketLifecycleResp, the response for PutBucketLifecycle operation.

```
// Adds LifecycleRule
void AddRule(const LifecycleRule& rule)

// Sets LifecycleRule
void SetRule(const std::vector<LifecycleRule>& rules)
```

LifecycleRule is composed as follows:

```
struct LifecycleTag {
std::string key;
std::string value;
};

class LifecycleFilter {
public:
LifecycleFilter();

std::string GetPrefix();
std::vector<LifecycleTag> GetTags();

void SetPrefix(const std::string& prefix);
void SetTags(const std::vector<LifecycleTag>& tags);
void AddTag(const LifecycleTag& tag);

bool HasPrefix();
bool HasTags();
```

```
private:
std::string m_prefix; // Specifies the prefix to which the rule applies. The objects matching the prefix are subject to the rule. Only one prefix is allowed.
std::vector<LifecycleTag> m_tags; // Tag. You can specify more than one tags, or leave it empty.
};

class LifecycleTransition {
public:
LifecycleTransition();

uint64_t GetDays();
std::string GetDate();
std::string GetStorageClass();

void SetDays(uint64_t days);
void SetDate(const std::string& date);
void SetStorageClass(const std::string& storage_class);

bool HasDays();
bool HasDate();
bool HasStorageClass();

private:
// Using both Days and Date in the same rule is not allowed.
uint64_t m_days; // Specifies the number of days from the last modification date of object until the operation specified by the rule is performed. It should be a non-negative integer.
std::string m_date; // Indicates when the operation specified by the rule is performed.
std::string m_storage_class; // Specifies the storage class to which the object is switched. Enumerated values: Standard_IA
};

class LifecycleExpiration {
public:
LifecycleExpiration();

uint64_t GetDays();
std::string GetDate();
bool IsExpiredObjDelMarker();

void SetDays(uint64_t days);
void SetDate(const std::string& date);
void SetExpiredObjDelMarker(bool marker);

bool HasDays();
bool HasDate();
bool HasExpiredObjDelMarker();

private:
// Using both Days and Date in the same rule is not allowed.
uint64_t m_days; // Specifies the number of days from the last modification date of object until the operation sp
```

```
ecified by the rule is performed. It should be a positive integer.
std::string m_date; // Indicates when the operation specified by the rule is performed.
bool m_expired_obj_del_marker; // Indicates whether to delete expired object. Enumerated value: True, False.
};

class LifecycleNonCurrTransition {
public:
LifecycleNonCurrTransition();

uint64_t GetDays();
std::string GetStorageClass();

void SetDays(uint64_t days);
void SetStorageClass(const std::string& storage_class);

bool HasDays();
bool HasStorageClass();

private:
uint64_t m_days; // Specifies the number of days from the last modification date of object until the operation sp
ecified by the rule is performed. It should be a non-negative integer.
std::string m_storage_class; // Specifies the storage class to which the object is switched. Enumerated values: Stan
dard_IA
};

class LifecycleNonCurrExpiration {
public:
LifecycleNonCurrExpiration();

uint64_t GetDays();

void SetDays(uint64_t days);

bool HasDays();

private:
uint64_t m_days; // Specifies the number of days from the last modification date of object until the operation sp
ecified by the rule is performed. It should be a positive integer.
};

struct AbortIncompleteMultipartUpload {
uint64_t m_days_after_init; // Indicates the number of days within which the multipart upload must be completed
after it starts.
};

class LifecycleRule {
public:
LifecycleRule();

void SetIsEnable(bool is_enable);
void SetId(const std::string& id);
```

```
void SetFilter(const LifecycleFilter& filter);
void AddTransition(const LifecycleTransition& rh);
void SetExpiration(const LifecycleExpiration& rh);
void SetNonCurrTransition(const LifecycleNonCurrTransition& rh);
void SetNonCurrExpiration(const LifecycleNonCurrExpiration& rh);
void SetAbortIncompleteMultiUpload(const AbortIncompleteMultipartUpload& rh);

bool IsEnable();
std::string GetId();
LifecycleFilter GetFilter();
std::vector<LifecycleTransition> GetTransitions();
LifecycleExpiration GetExpiration();
LifecycleNonCurrTransition GetNonCurrTransition();
LifecycleNonCurrExpiration GetNonCurrExpiration();
AbortIncompleteMultipartUpload GetAbortIncompleteMultiUpload();

bool HasIsEnable();
bool HasId();
bool HasFilter();
bool HasExpiration();
bool HasNonCurrTransition();
bool HasNonCurrExpiration();
bool HasAbortIncomMultiUpload();

private:
bool m_is_enable; // Whether the rule takes effect
std::string m_id; // Rule ID
LifecycleFilter m_filter; / / Filter, which specifies the object scope on which the rule takes effect
std::vector<LifecycleTransition> m_transitions; // Transition operation
LifecycleExpiration m_expiration;// Expiration operation
LifecycleNonCurrTransition m_non_curr_transition; // Transition operation on Object of non-current version
LifecycleNonCurrExpiration m_non_curr_expiration; // Expiration operation on Object of non-current version
AbortIncompleteMultipartUpload m_abort_multi_upload; // Sets the maximum time length allowed for a multipar
t upload.
}
```

Example

```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);

std::string bucket_name = "cpp_sdk_v5-123456789";

// The bucket_name is required in the constructor of PutBucketLifecycleReq
qcloud_cos::PutBucketLifecycleReq req(bucket_name);
// Sets rule 1
{
qcloud_cos::LifecycleRule rule;
rule.SetIsEnable(true);
rule.SetId("lifecycle_rule00");
```

```
qcloud_cos::LifecycleFilter filter;
filter.SetPrefix("sevenyou_e1");
rule.SetFilter(filter);
qcloud_cos::LifecycleExpiration expiration;
expiration.SetDays(1);
rule.SetExpiration(expiration);
req.AddRule(rule);
}

// Sets rule 2
{
qcloud_cos::LifecycleRule rule;
rule.SetIsEnable(true);
rule.SetId("lifecycle_rule01");
qcloud_cos::LifecycleFilter filter;
filter.SetPrefix("sevenyou_e2");
rule.SetFilter(filter);
qcloud_cos::LifecycleExpiration expiration;
expiration.SetDays(3);
rule.SetExpiration(expiration);
req.AddRule(rule);
}

qcloud_cos::PutBucketLifecycleResp resp;
qcloud_cos::CosResult result = cos.PutBucketLifecycle(req, &resp);

// The call is successful. Call resp's member functions to obtain the response content
if (result.IsSucc()) {
// ...
} else {
// Failed to set the lifecycle. Call CosResult's member functions to output error information, such as requestID, etc.
}
```

Get Bucket Lifecycle

Feature description

This API (Get Bucket Lifecycle) is used to obtain the lifecycle configuration of a Bucket. If no lifecycle rule is configured for the Bucket, NoSuchLifecycleConfiguration is returned. For more information about this API, please see Get Bucket Lifecycle.

Method prototype

```
CosResult GetBucketLifecycle(const GetBucketLifecycleReq& req, GetBucketLifecycleResp* resp);
```

Parameters

Parameter	Description
req	GetBucketLifecycleReq, the request for GetBucketLifecycle operation.

Parameter	Description
resp	GetBucketLifecycleResp, the response for GetBucketLifecycleoperation.

```
// Obtains LifecycleRules
std::vector<LifecycleRule> GetRules()
```

For the definition of LifecycleRule, please see Put Bucket Lifecycle.

Example

```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);

std::string bucket_name = "cpp_sdk_v5-123456789";

// The bucket_name is required in the constructor of GetBucketLifecycleReq
qcloud_cos::GetBucketLifecycleReq req(bucket_name);
qcloud_cos::GetBucketLifecycleResp resp;
qcloud_cos::CosResult result = cos.GetBucketLifecycle(req, &resp);

// The call is successful. Call resp's member functions to obtain the response content
if (result.IsSucc()) {
// ...
} else {
// Failed to obtain lifecycle configuration. Call CosResult's member functions to output error information, such as requestID, etc.
}
```

Delete Bucket Lifecycle

Feature description

This API (Delete Bucket Lifecycle) is used to delete the lifecycle configuration of a Bucket. If no lifecycle rule is configured for the Bucket, NoSuchLifecycleConfiguration is returned. For more information about this API, please see Delete Bucket Lifecycle.

Method prototype

```
CosResult DeleteBucketLifecycle(const DeleteBucketLifecycleReq& req, DeleteBucketLifecycleResp* resp);
```

Parameters

Parameter	Description
req	DeleteBucketLifecycleReq, the request for DeleteBucketLifecycle operation.
resp	DeleteBucketLifecycleResp, the response for DeleteBucketLifecycle.

Example

```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);

std::string bucket_name = "cpp_sdk_v5-123456789";

// The bucket_name is required in the constructor of DeleteBucketLifecycleReq
qcloud_cos::DeleteBucketLifecycleReq req(bucket_name);
qcloud_cos::DeleteBucketLifecycleResp resp;
qcloud_cos::CosResult result = cos.DeleteBucketLifecycle(req, &resp);

// The call is successful. Call resp's member functions to obtain the response content
if (result.IsSucc()) {
// ...
} else {
// Failed to delete lifecycle configuration. Call CosResult's member functions to output error information, such as requestID, etc.
}
```

Put Bucket CORS

Feature description

This API (Put Bucket CORS) is used to set cross-origin resource sharing permission for a Bucket by importing configuration files in XML format (file size limit: 64 KB). By default, the Bucket owner has the permission to use this API and can grant the permission to others. For more information about this API, please see Put Bucket CORS.

Method prototype

```
CosResult PutBucketCORS(const DPutBucketCORSReq& req, PutBucketCORSResp* resp);
```

Parameters

Parameter	Description
req	PutBucketCORSReq, the request for PutBucketCORS operation.
resp	PutBucketCORSResp, the response for PutBucketCORS operation.

```
// Adds CORSRule
void AddRule(const CORSRule& rule);

// Sets CORSRule
void SetRules(const std::vector<CORSRule>& rules)
```

CORSRule is composed as follows:

```
struct CORSRule {
std::string m_id; // Sets rule ID (optional)
std::string m_max_age_secs; // Sets the validity period of the results obtained by OPTIONS
std::vector<std::string> m_allowed_headers; // When an OPTIONS request is sent, notifies the server about which c
ustom HTTP request headers are allowed for subsequent requests. Wildcard "*" is supported.
std::vector<std::string> m_allowed_methods; // Allowed HTTP operations. Enumerated values: GET, PUT, HEAD, PO
ST, DELETE.
std::vector<std::string> m_allowed_origins; // Allowed access sources. Wildcard "*" is supported. Format: protoco
l://domain name[:port], for example, http://www.qq.com
std::vector<std::string> m_expose_headers; // Sets the custom header information that can be received by the bro
wser from the server. |
};
```

Example

```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);

std::string bucket_name = "cpp_sdk_v5-123456789";

// The bucket_name is required in the constructor of PutBucketCORSReq
qcloud_cos::PutBucketCORSReq req(bucket_name);
qcloud_cos::CORSRule rule;
rule.m_id = "123";
rule.m_allowed_headers.push_back("x-cos-meta-test");
rule.m_allowed_origins.push_back("http://www.qq.com");
rule.m_allowed_origins.push_back("http://cloud.tentent.com");
rule.m_allowed_methods.push_back("PUT");
rule.m_allowed_methods.push_back("GET");
rule.m_max_age_secs = "600";
rule.m_expose_headers.push_back("x-cos-expose");
req.AddRule(rule);

qcloud_cos::PutBucketCORSResp resp;
qcloud_cos::CosResult result = cos.PutBucketCORS(req, &resp);

// The call is successful. Call resp's member functions to obtain the response content
if (result.IsSucc()) {
// ...
} else {
// Failed to set the cross-origin resource sharing permission. Call CosResult's member functions to output error inf
ormation, such as requestID, etc.
}
```

Get Bucket CORS

Feature description

This API (Get Bucket CORS) is used by the Bucket owner to configure cross-origin resource sharing on a bucket. CORS (Cross-Origin Resource Sharing) is a W3C standard. By default, the Bucket owner has the permission to use this API and can grant the permission to others. For more information about this API, please see Get Bucket CORS.

Method prototype

```
CosResult GetBucketCORS(const DGetBucketCORSReq& req, GetBucketCORSResp* resp);
```

Parameters

Parameter	Description
req	GetBucketCORSReq, the request for GetBucketCORS operation.
resp	GetBucketCORSResp, the response for GetBucketCORS operation.

```
// Obtains CORSRules. For the definition of CORSRule, please see Put Bucket CORS.
std::vector<CORSRule> GetCORSRules();
```

Example

```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);

std::string bucket_name = "cpp_sdk_v5-123456789";

// The bucket_name is required in the constructor of GetBucketCORSReq
qcloud_cos::GetBucketCORSReq req(bucket_name);
qcloud_cos::GetBucketCORSResp resp;
qcloud_cos::CosResult result = cos.GetBucketCORS(req, &resp);

// The call is successful. Call resp's member functions to obtain the response content
if (result.IsSucc()) {
// ...
} else {
// Failed to obtain CORS configuration. Call CosResult's member functions to output error information, such as requestID, etc.
}
```

Delete Bucket CORS

Feature description

This API (Delete Bucket CORS) is used to delete the CORS configuration of a Bucket. If no CORS information is configured for the Bucket, NoSuchCORSConfiguration is returned. For more information about this API, please see Delete Bucket CORS.

Method prototype

```
CosResult DeleteBucketCORS(const DDeleteBucketCORSReq& req, DeleteBucketCORSResp* resp);
```

Parameters

Parameter	Description
req	DeleteBucketCORSReq, the request for DeleteBucketCORS operation.
resp	DeleteBucketCORSResp, the response for DeleteBucketCORS operation.

Example

```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);

std::string bucket_name = "cpp_sdk_v5-123456789";

// The bucket_name is required in the constructor of DeleteBucketCORSReq
qcloud_cos::DeleteBucketCORSReq req(bucket_name);
qcloud_cos::DeleteBucketCORSResp resp;
qcloud_cos::CosResult result = cos.DeleteBucketCORS(req, &resp);

// The call is successful. Call resp's member functions to obtain the response content
if (result.IsSucc()) {
// ...
} else {
// Failed to delete CORS configuration. Call CosResult's member functions to output error information, such as requestID, etc.
}
```

Put Bucket ACL

Feature description

This API (Put Bucket ACL) is used to write ACL for a Bucket. You can import ACL information either by using Header: "x-cos-acl", "x-cos-grant-read", "x-cos-grant-write", "x-cos-grant-full-control", or by using Body in XML format. For more information about this API, please see Put Bucket ACL.

Method prototype

```
CosResult PutBucketACL(const DPutBucketACLReq& req, PutBucketACLResp* resp);
```

Parameters

Parameter	Description
req	PutBucketACLReq, the request for PutBucketACL operation.

Parameter	Description
resp	PutBucketACLResp, the response for PutBucketACL operation.

```
// Defines the ACL attribute of Bucket. Valid values: private, public-read-write, public-read
// Default: private
void SetXCosAcl(const std::string& str);

// Grants read permission to the authorized user. Format: x-cos-grant-read: id=" ",id=" ".
// For authorization to a sub-account, id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>"
// For authorization to a root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>"
void SetXCosGrantRead(const std::string& str);

// Grants write permission to the authorized user. Format: x-cos-grant-write: id=" ",id=" "./
// For authorization to a sub-account, ,id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>",
// For authorization to a root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>"
void SetXCosGrantWrite(const std::string& str);

// Grants read and write permissions to the authorized user. Format: x-cos-grant-full-control: id=" ",id=" ".
//For authorization to a sub-account, id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>",
// For authorization to a root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>"
void SetXCosGrantFullControl(const std::string& str);

// Bucket owner ID
void SetOwner(const Owner& owner);

// Sets the information of authorized user and permissions
void SetAccessControlList(const std::vector<Grant>& grants);

// Adds authorization information for a single Bucket
void AddAccessControlList(const Grant& grant);
```

Note:
APIs such as SetXCosAcl, SetXCosGrantRead, SetXCosGrantWrite, and SetXCosGrantFullControl cannot be used with SetAccessControlList and AddAccessControlList. This is because the former kind of APIs are implemented by setting HTTP Header, while the latter kind of APIs are implemented by adding content in XML format to the Body. You can only use either of the two kinds of APIs. The first kinds of APIs are preferred in SDK.

ACLRule is composed as follows:

```
struct Grantee {
// "type" can be RootAccount or SubAccount.
// If type is RootAccount, you can enter account ID in "uin" or in "uin" of id, or replace uin/<OwnerUin> and uin/
<SubUin> with "anyone" (all types of users).
// If type is RootAcount, uin represents root account, and SubAccount represents sub-account.
```

```
std::string m_type;
std::string m_id; // qcs::cam::uin/<OwnerUin>:uin/<SubUin>
std::string m_display_name; // Optional
std::string m_uri;
};

struct Grant {
Grantee m_grantee; // Resource information of the authorized user
std::string m_perm; // Indicates the permission granted to the authorized user. Enumerated values: READ, WRITE, and FULL_CONTROL
};
```

Example

```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);

std::string bucket_name = "cpp_sdk_v5-123456789";

// The bucket_name is required in the constructor of PutBucketACLReq
qcloud_cos::PutBucketACLReq req(bucket_name);
qcloud_cos::ACLRule rule;
rule.m_id = "123";
rule.m_allowed_headers.push_back("x-cos-meta-test");
rule.m_allowed_origins.push_back("http://www.qq.com");
rule.m_allowed_origins.push_back("http://cloud.tentent.com");
rule.m_allowed_methods.push_back("PUT");
rule.m_allowed_methods.push_back("GET");
rule.m_max_age_secs = "600";
rule.m_expose_headers.push_back("x-cos-expose");
req.AddRule(rule);

qcloud_cos::PutBucketACLResp resp;
qcloud_cos::CosResult result = cos.PutBucketACL(req, &resp);

// The call is successful. Call resp's member functions to obtain the response content
if (result.IsSucc()) {
// ...
} else {
// Sets the ACL. Call CosResult's member functions to output error information, such as requestID, etc.
}
```

Get Bucket ACL

Feature description

This API (Get Bucket ACL) is used to obtain the ACL (Access Control List) of a Bucket. Only the Bucket owner has the access to this API. For more information about this API, please see Get Bucket ACL.

Method prototype

```
CosResult GetBucketACL(const DGetBucketACLReq& req, GetBucketACLResp* resp);
```

Parameters

Parameter	Description
req	GetBucketACLReq, the request for GetBucketACL operation.
resp	GetBucketACLResp, the response for GetBucketACL operation.

```
std::string GetOwnerID();
std::string GetOwnerDisplayName();
std::vector<Grant> GetAccessControlList();
```

Example

```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);

std::string bucket_name = "cpp_sdk_v5-123456789";

// The bucket_name is required in the constructor of GetBucketACLReq
qcloud_cos::GetBucketACLReq req(bucket_name);
qcloud_cos::GetBucketACLResp resp;
qcloud_cos::CosResult result = cos.GetBucketACL(req, &resp);

// The call is successful. Call resp's member functions to obtain the response content
if (result.IsSucc()) {
// ...
} else {
// Failed to obtain the ACL. Call CosResult's member functions to output error information, such as requestID, etc.
}
```

Object Operations

Parameter object_name is the object key, which is the unique identifier of an object in the bucket. For example, in the object's access domain name bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg, the object key is doc1/pic1.jpg.

For more information about "object key", please see Object Overview.

Get Object

Feature description

This API (Get Object) is used to download a file (Object) locally or to a specified stream. This operation requires that the user have the read permission for the target Object or the read permission for the target Object be available for everyone (public-read).

Method prototype

```
// Downloads Object to a local file
CosResult GetObject(const GetObjectByFileReq& req, GetObjectByFileResp* resp);

// Downloads Object to a stream
CosResult GetObject(const GetObjectByStreamReq& req, GetObjectByStreamResp* resp);

// Downloads Object to a local file (multi-thread)
CosResult GetObject(const MultiGetObjectReq& req, MultiGetObjectResp* resp);
```

Parameters

Parameter	Description
req	GetObjectByFileReq/GetObjectByStreamReq/MultiGetObjectReq, the request for GetObject operation.
resp	GetObjectByFileResp/GetObjectByStreamResp/MultiGetObjectResp, the response for GetObject operation.

The member functions are as follows:

```
// Sets the Content-Type parameter in the response header
void SetResponseContentType(const std::string& str);

// Sets the Content-Language parameter in the response header
void SetResponseContentLang(const std::string& str);

// Sets the Content-Expires parameter in the response header
void SetResponseExpires(const std::string& str);

// Sets the Cache-Control parameter in the response header
void SetResponseCacheControl(const std::string& str);

// Sets the Content-Disposition parameter in the response header
void SetResponseContentDisposition(const std::string& str);

// Sets the Content-Encoding parameter in the response header
void SetResponseContentEncoding(const std::string& str);
```

GetObjectResp not only reads the member functions of common headers, but also provides the following member functions:

```
// Obtains the last modification time of the Object. Date format (string): "28 Oct 2014 20:30:00 GMT"
std::string GetLastModified();

// Obtains Object type, which indicates whether the Object is appendable for upload. Enumerated values: normal or appendable
std::string GetXCosObjectType();

// Obtains the storage class of an Object. Enumerated values: STANDARD,STANDARD_IA
std::string GetXCosStorageClass();

// Returns all custom meta in the form of map. The key of a map does not contain the prefix "x-cos-meta-"
std::map<std::string, std::string> GetXCosMetas();

// Obtains the custom meta. The parameter can be "*" in "x-cos-meta-*"
std::string GetXCosMeta(const std::string& key);

// Obtains the algorithm used by server for encryption
std::string GetXCosServerSideEncryption();
```

Example

```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);

std::string bucket_name = "cpp_sdk_v5-123456789";
std::string object_name = "object_name";
std::string local_path = "/tmp/object_name";

// Downloads to a local file
{
// Parameters appid, bucketname and object, as well as the local path (including file name) are required for the request
qcloud_cos::GetObjectByFileReq req(bucket_name, object_name, local_path);
qcloud_cos::GetObjectByFileResp resp;
qcloud_cos::CosResult result = cos.GetObject(req, &resp);
if (result.IsSucc()) {
// Download is successful. Call GetObjectByFileResp's member functions
} else {
// Download failed. Call CosResult's member functions to output error information, such as requestID, etc.
}
}

// Download to a stream
{
// Parameters appid, bucketname and object, as well as output stream are required for the request
std::ostringstream os;
qcloud_cos::GetObjectByStreamReq req(bucket_name, object_name, os);
qcloud_cos::GetObjectByStreamResp resp;
qcloud_cos::CosResult result = cos.GetObject(req, &resp);
```

```
if (result.IsSucc()) {
// Download is successful. Call GetObjectByStreamResp's member functions
} else {
// Download failed. Call CosResult's member functions to output error information, such as requestID, etc.
}
}

// Downloads file locally in multiple threads
{
// Parameters appid, bucketname and object, as well as the local path (including file name) are required for the r
equest
qcloud_cos::MultiGetObjectReq req(bucket_name, object_name, local_path);
qcloud_cos::MultiGetObjectResp resp;
qcloud_cos::CosResult result = cos.GetObject(req, &resp);
if (result.IsSucc()) {
// Download is successful. Call MultiGetObjectResp's member functions
} else {
// Download failed. Call CosResult's member functions to output error information, such as requestID, etc.
}
}
```

Head Object

Feature description

This API (Head Object) is used to get the metadata of an Object. It has the same permissions as Get Object.

Method prototype

```
CosResult HeadObject(const HeadObjectReq& req, HeadObjectResp* resp);
```

Parameters

Parameter	Description
req	HeadObjectReq, the request for HeadObject operation.
resp	HeadObjectResp, the response for HeadObject operation.

HeadObjectResp not only reads the member functions of common headers, but also provides the following member functions:

```
std::string GetXCosObjectType();

std::string GetXCosStorageClass();

// Obtains the custom meta. The parameter can be "*" in "x-cos-meta-*"
std::string GetXCosMeta(const std::string& key);
```

```
// Returns all custom meta in the form of map. The key of a map does not contain the prefix "x-cos-meta-"
std::map<std::string, std::string> GetXCosMetas();

// Obtains the algorithm used by server for encryption
std::string GetXCosServerSideEncryption();
```

Example

```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);

std::string bucket_name = "cpp_sdk_v5-123456789";
std::string object_name = "object_name";
qcloud_cos::HeadObjectReq req(bucket_name, object_name);
qcloud_cos::HeadObjectResp resp;
qcloud_cos::CosResult result = cos.HeadObject(req, &resp);
if (result.IsSucc()) {
// Download is successful. Call HeadObjectResp's member functions
} else {
// Download failed. Call CosResult's member functions to output error information, such as requestID, etc.
}
```

Put Object

Feature description

This API (Put Object) is used to upload a file (Object) to the specified Bucket.

Method prototype

```
/// Uploads via Stream
CosResult PutObject(const PutObjectByStreamReq& req, PutObjectByStreamResp* resp);

/// Uploads a local file
CosResult PutObject(const PutObjectByFileReq& req, PutObjectByFileResp* resp);
```

Parameters

Parameter	Description
req	PutObjectByStreamReq/PutObjectByFileReq, the request for PutObject operation.
resp	PutObjectByStreamResp/PutObjectByFileResp, the response for PutObject operation.

PutObject*Req contains the following member functions:

```
// Cache-Control, the caching policy defined in RFC 2616 and saved as Object metadata.
void SetCacheControl(const std::string& str);
```

```
// Content-Disposition, the file name defined in RFC 2616 and saved as Object metadata.
void SetContentDisposition(const std::string& str);

// Content-Encoding, the encoding format defined in RFC 2616 and saved as Object metadata.
void SetContentEncoding(const std::string& str);

// Content-Type, the content type (MIME) defined in RFC 2616 and saved as Object metadata.
void SetContentType(const std::string& str);

// Expect If Expect: 100-continue is used, the request content will not be sent until the receipt of response from se
rver.
void SetExpect(const std::string& str);

// Expires, the expiration time defined in RFC 2616 and saved as Object metadata.
void SetExpires(const std::string& str);

// The header information that can be defined by users, which is returned as Object metadata. The size is limited t
o 2 KB.
void SetXCosMeta(const std::string& key, const std::string& value);

// x-cos-storage-class, which is used to set the storage class of Object. Enumerated values: STANDARD, STANDAR
D_IA
// Default: STANDARD (supported only in South China region)
void SetXCosStorageClass(const std::string& storage_class);

// Defines the ACL attribute of Object. Valid values: private, public-read-write, public-read.
// Default: private
void SetXcosAcl(const std::string& str);

// Grants read permission to the authorized user. Format: x-cos-grant-read: id=" ",id=" ".
// For authorization to a sub-account, id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>"
// For authorization to a root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>"
void SetXcosGrantRead(const std::string& str);

// Grants write permission to the authorized user. Format: x-cos-grant-write: id=" ",id=" "./
// For authorization to a sub-account,id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>",
// For authorization to a root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>"
void SetXcosGrantWrite(const std::string& str);

// Grants read and write permissions to the authorized user. Format: x-cos-grant-full-control: id=" ",id=" ".
//For authorization to a sub-account,id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>",
// For authorization to a root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>"
void SetXcosGrantFullControl(const std::string& str);

/// Sets the algorithm used by server for encryption. AES256 is supported.
void SetXCosServerSideEncryption(const std::string& str);
```

PutObject*Resp contains the following member functions:

```
/// Obtains the version number of Object. If multiple versions of an object are not enabled in a Bucket, empty stri
ng is returned.
std::string GetVersionId();

/// Algorithm used by server for encryption
std::string GetXCosServerSideEncryption();
```

Example

```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);

std::string bucket_name = "cpp_sdk_v5-123456789";
std::string object_name = "object_name";

// Simple upload (stream)
{
std::istringstream iss("put object");
// istream is required in the constructor of the request
qcloud_cos::PutObjectByStreamReq req(bucket_name, object_name, iss);
// Calls Set method to set metadata or ACL
req.SetXCosStorageClass("STANDARD_IA");
qcloud_cos::PutObjectByStreamResp resp;
qcloud_cos::CosResult result = cos.PutObject(req, &resp);

if (result.IsSucc()) {
// The call is successful. Call resp's member functions to obtain the response content
do sth
} else {
// The call failed. Call result's member functions to obtain the response content
std::cout << "ErrorInfo=" << result.GetErrorInfo() << std::endl;
std::cout << "HttpStatus=" << result.GetHttpStatus() << std::endl;
std::cout << "ErrorCode=" << result.GetErrorCode() << std::endl;
std::cout << "ErrorMsg=" << result.GetErrorMsg() << std::endl;
std::cout << "ResourceAddr=" << result.GetResourceAddr() << std::endl;
std::cout << "XCosRequestId=" << result.GetXCosRequestId() << std::endl;
std::cout << "XCosTraceId=" << result.GetXCosTraceId() << std::endl;
}
}

// Simple upload (file)
{
// The local file path is required in the constructor of the request
qcloud_cos::PutObjectByFileReq req(bucket_name, object_name, "/path/to/local/file");
// Calls Set method to set metadata or ACL
req.SetXCosStorageClass("STANDARD_IA");
qcloud_cos::PutObjectByFileResp resp;
qcloud_cos::CosResult result = cos.PutObject(req, &resp);
if (result.IsSucc()) {
```

```
// The call is successful. Call resp's member functions to obtain the response content
do sth
} else {
// The call failed. Call result's member functions to obtain the response content
std::cout << "ErrorInfo=" << result.GetErrorInfo() << std::endl;
std::cout << "HttpStatus=" << result.GetHttpStatus() << std::endl;
std::cout << "ErrorCode=" << result.GetErrorCode() << std::endl;
std::cout << "ErrorMsg=" << result.GetErrorMsg() << std::endl;
std::cout << "ResourceAddr=" << result.GetResourceAddr() << std::endl;
std::cout << "XCosRequestId=" << result.GetXCosRequestId() << std::endl;
std::cout << "XCosTraceId=" << result.GetXCosTraceId() << std::endl;
}
}
```

Delete Object

Feature description

This API (Delete Object) is used to delete a file (Object) from a Bucket of COS. This operation requires that the user have the WRITE permission for the Bucket. For more information about this API, please see Delete Object.

Method prototype

```
CosResult DeleteObject(const DeleteObjectReq& req, DeleteObjectResp* resp);
```

Parameters

Parameter	Description
req	DeleteObjectReq, the request for DeleteObject operation.
resp	DeletObjectResp, the response for DeletObject operation.

Example

```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);

std::string bucket_name = "cpp_sdk_v5-123456789";
std::string object_name = "test_object";

qcloud_cos::DeleteObjectReq req(bucket_name, object_name);
qcloud_cos::DeleteObjectResp resp;
qcloud_cos::CosResult result = cos.DeleteObject(req, &resp);

// The call is successful. Call resp's member functions to obtain the response content
if (result.IsSucc()) {
// ...
} else {
```

```
// Failed to delete Object. Call CosResult's member functions to output error information, such as requestID, etc.
}
```

Multipart Upload Operations

Initiate Multipart Upload

Feature description

This API (Initiate Multipart Upload) is used to initialize multipart upload. After the request is executed successfully, Upload ID is returned for the subsequent Upload Part requests.

Method prototype

```
CosResult InitMultiUpload(const InitMultiUploadReq& req, InitMultiUploadResp* resp);
```

Parameters

Parameter	Description
req	InitMultiUploadReq, the request for InitMultiUpload operation.
resp	InitMultiUploadResp, the response for InitMultiUpload operation.

InitMultiUploadReq contains the following member functions:

```
// Cache-Control, the caching policy defined in RFC 2616 and saved as Object metadata.
void SetCacheControl(const std::string& str);

// Content-Disposition, the file name defined in RFC 2616 and saved as Object metadata.
void SetContentDisposition(const std::string& str);

// Content-Encoding, the encoding format defined in RFC 2616 and saved as Object metadata.
void SetContentEncoding(const std::string& str);

// Content-Type, the content type (MIME) defined in RFC 2616 and saved as Object metadata.
void SetContentType(const std::string& str);

// Expires, the expiration time defined in RFC 2616 and saved as Object metadata.
void SetExpires(const std::string& str);

// The header information that can be defined by users, which is returned as Object metadata. The size is limited to 2 KB.
void SetXCosMeta(const std::string& key, const std::string& value);

// x-cos-storage-class, which is used to set the storage class of Object. Enumerated values: STANDARD, STANDARD_IA
// Default: STANDARD
```

```
void SetXCosStorageClass(const std::string& storage_class);

// Defines the ACL attribute of Object. Valid values: private, public-read-write, public-read.
// Default: private
void SetXcosAcl(const std::string& str);

// Grants read permission to the authorized user. Format: x-cos-grant-read: id=" ",id=" ".
// For authorization to a sub-account, id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>"
// For authorization to a root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>"
void SetXcosGrantRead(const std::string& str);

// Grants write permission to the authorized user. Format: x-cos-grant-write: id=" ",id=" "./
// For authorization to a sub-account, id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>",
// For authorization to a root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>"
void SetXcosGrantWrite(const std::string& str);

// Grants read and write permissions to the authorized user. Format: x-cos-grant-full-control: id=" ",id=" ".
// For authorization to a sub-account,id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>",
// For authorization to a root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>"
void SetXcosGrantFullControl(const std::string& str);

/// Sets the algorithm used by server for encryption. AES256 is supported.
void SetXCosServerSideEncryption(const std::string& str);
```

When the request is executed successfully, the response containing bucket (destination Bucket of multipart upload), key (object name) and uploadId (ID required for the subsequent multipart uploads) is returned.

InitMultiUploadResp contains the following member functions:

```
std::string GetBucket();
std::string GetKey();
std::string GetUploadId();

// Algorithm used by server for encryption
std::string GetXCosServerSideEncryption();
```

Example

```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);
std::string bucket_name = "cpp_sdk_v5-123456789";
std::string object_name = "object_name";

qcloud_cos::InitMultiUploadReq req(bucket_name, object_name);
qcloud_cos::InitMultiUploadResp resp;
qcloud_cos::CosResult result = cos.InitMultiUpload(req, &resp);

std::string upload_id = "";
if (result.IsSucc()) {
```

```
upload_id = resp.GetUploadId();
}
```

Upload Part

Feature description

This API (Upload Part) is used to implement multipart upload after initialization. A file can be split into 10000 chunks at most (minimum is 1) for multipart upload, and the size of each file chunk should be between 1 MB and 5 GB. Parameters partNumber and uploadId are required for Upload Part (partNumber is the file chunk No. Out-of-order upload is supported).

Method prototype

```
CosResult UploadPartData(const UploadPartDataReq& request, UploadPartDataResp* response);
```

Parameters

Parameter	Description
req	UploadPartDataReq, the request for UploadPartData operation.
resp	UploadPartDataResp, the response for UploadPartData operation.

When constructing UploadPartDataReq, you need to specify request's APPID, Bucket, Object, UploadId obtained after the initialization is completed, and the data stream for upload (after calling this API, the caller needs to close the stream).

```
UploadPartDataReq(const std::string& bucket_name,
const std::string& object_name, const std::string& upload_id,
std::istream& in_stream);
```

In addition, file chunk No. is required in the request. This chunk is needed when the multipart upload is completed.

```
void SetPartNumber(uint64_t part_number);
```

UploadPartDataResp contains the following member functions:

```
/// Algorithm used by server for encryption
std::string GetXCosServerSideEncryption();
```

Example

```
// Uploads the first chunk
{
std::fstream is("demo_5M.part1");
qcloud_cos::UploadPartDataReq req(bucket_name, object_name,
```

```
upload_id, is);
req.SetPartNumber(1);
qcloud_cos::UploadPartDataResp resp;
qcloud_cos::CosResult result = cos.UploadPartData(req, &resp);

// After the upload is completed, record the chunk number and Etag returned
if (result.IsSucc()) {
etags.push_back(resp.GetEtag());
part_numbers.push_back(1);
}
is.close();
}

// Uploads the second chunk
{
std::fstream is("demo_5M.part2");
qcloud_cos::UploadPartDataReq req(bucket_name, object_name,
upload_id, is);
req.SetPartNumber(2);
qcloud_cos::UploadPartDataResp resp;
qcloud_cos::CosResult result = cos.UploadPartData(req, &resp);

// After the upload is completed, record the chunk number and Etag returned
if (result.IsSucc()) {
etags.push_back(resp.GetEtag());
part_numbers.push_back(2);
}
is.close();
}
```

Complete Multipart Upload

Feature description

This API (Complete Multipart Upload) is used to complete the entire multipart upload. After you have uploaded all the file chunks using Upload Parts, you can use this API to complete the upload. When using this API, you need to provide the PartNumber and ETag for every chunk in Body, to verify the accuracy of chunks.

Method prototype

```
CosResult CompleteMultiUpload(const CompleteMultiUploadReq& request, CompleteMultiUploadResp* respon
se);
```

Parameters

Parameter	Description
req	CompleteMultiUploadReq, the request for CompleteMultiUpload operation.
resp	CompleteMultiUploadResp, the response for CompleteMultiUpload operation.

When constructing CompleteMultiUploadReq, you need to specify request APPID, Bucket, Object, and UploadId obtained after the initialization.

```
CompleteMultiUploadReq(const std::string& bucket_name,
const std::string& object_name, const std::string& upload_id)
```

In addition, the numbers and ETags of all the uploaded file chunks are also required for the request

```
// When the following methods are called, the numbers should correspond to Etags in sequence on an one-on-one basis.
void SetPartNumbers(const std::vector<uint64_t>& part_numbers);
void SetEtags(const std::vector<std::string>& etags) ;

// Adds part_number and ETag pairs
void AddPartEtagPair(uint64_t part_number, const std::string& etag);

/// Sets the algorithm used by server for encryption. AES256 is supported.
void SetXCosServerSideEncryption(const std::string& str);
```

The response returned for CompleteMultiUploadResp contains Location (domain name of the accessing public network of the created object), Bucket (the destination Bucket for multipart upload), Key (object name) and ETag (MD5 algorithm check value for the merged file). You can call the following member functions to access the response content.

```
std::string GetLocation();
std::string GetKey();
std::string GetBucket();
std::string GetEtag();

/// Algorithm used by server for encryption
std::string GetXCosServerSideEncryption();
```

Example

```
qcloud_cos::CompleteMultiUploadReq req(bucket_name, object_name, upload_id);
qcloud_cos::CompleteMultiUploadResp resp;
req.SetEtags(etags);
req.SetPartNumbers(part_numbers);

qcloud_cos::CosResult result = cos.CompleteMultiUpload(req, &resp);
```

Multipart Upload

Feature description

Multipart Upload encapsulates three steps: initializing multipart upload, executing multipart upload, and completing multipart upload. You only need to specify the file to be uploaded in the request.

Method prototype

```
CosResult MultiUploadObject(const MultiUploadObjectReq& request, MultiUploadObjectResp* response);
```

Parameters

Parameter	Description
req	MultiUploadObjectReq, the request for MultiUploadObject operation.
resp	MultiUploadObjectResp, the response for MultiUploadObject operation.

When constructing MultiUploadObjectReq, you need to specify Bucket, Object, and the path of local file to be uploaded. If the path is not specified, the file with the same name as the Object under the current working path is used by default.

```
MultiUploadObjectReq(const std::string& bucket_name,
const std::string& object_name, const std::string& local_file_path = "");

/// Sets the algorithm used by server for encryption. AES256 is supported.
void SetXCosServerSideEncryption(const std::string& str);
```

- If multipart upload is successful, the response content is the same as that of CompleteMultiUploadResp.
- If multipart upload fails, the response content is same as that of InitMultiUploadResp, UploadPartDataResp or CompleteMultiUploadResp, depending on the failure type. You can call `GetRespTag()` to determine the step where the failure occurs.

```
// Returns Init, Upload, Complete
std::string GetRespTag();

/// Algorithm used by server for encryption
std::string GetXCosServerSideEncryption();
```

Example

```
qcloud_cos::MultiUploadObjectReq req( bucket_name, object_name, "/temp/demo_6G.tmp");
qcloud_cos::MultiUploadObjectResp resp;
qcloud_cos::CosResult result = cos.MultiUploadObject(req, &resp);

if (result.IsSucc()) {
std::cout << resp.GetLocation() << std::endl;
std::cout << resp.GetKey() << std::endl;
std::cout << resp.GetBucket() << std::endl;
std::cout << resp.GetEtag() << std::endl;
} else {
// Determines the specific step where the failure occurs.
std::string resp_tag = resp.GetRespTag();
```

```
if ("Init" == resp_tag) {
// print result
} else if ("Upload" == resp_tag) {
// print result
} else if ("Complete" == resp_tag) {
// print result
}
}
```

Abort Multipart Upload

Feature description

This API (Abort Multipart Upload) is used to abort a multipart upload operation and delete uploaded file chunks. When Abort Multipart Upload is called, a failure is returned for any request that is using Upload Parts.

Method prototype

```
CosResult AbortMultiUpload(const AbortMultiUploadReq& request, AbortMultiUploadResp* response);
```

Parameters

Parameter	Description
req	AbortMultiUploadReq, the request for AbortMultiUpload.
resp	AbortMultiUploadResp, the response for AbortMultiUpload operation.

When constructing AbortMultiUploadReq, you need to specify Bucket, Object and Upload_id.

```
AbortMultiUploadReq(const std::string& bucket_name,
const std::string& object_name, const std::string& upload_id);
```

Call BaseResp's member functions to obtain common header unless otherwise specified.

Example

```
qcloud_cos::AbortMultiUploadReq req(bucket_name, object_name,
upload_id);
qcloud_cos::AbortMultiUploadResp resp;
qcloud_cos::CosResult result = cos.AbortMultiUpload(req, &resp);
```

List Parts

Feature description

This API (List Parts) is used to query the uploaded file chunks in a specific multipart upload, listing all the uploaded chunks under the specified UploadId. For more information about this API, please see List Parts.

Method prototype

```
CosResult ListParts(const ListPartsReq& req, ListPartsResp* resp);
```

Parameters

Parameter	Description
req	ListPartsReq, the request for ListParts operation.
resp	ListPartsResp, the response for ListParts operation.

```
// Constructor: Bucket name, Object name, multipart upload ID
ListPartsReq(const std::string& bucket_name,
const std::string& object_name,
const std::string& upload_id);

// \brief The encoding format of the returned results.
void SetEncodingType(const std::string& encoding_type);

// \brief The maximum number of entries returned at a time. Default is 1,000.
void SetMaxParts(uint64_t max_parts);

// \brief Entries are listed in UTF-8 binary order by default, starting from marker
void SetPartNumberMarker(const std::string& part_number_marker);
```

```
// The destination Bucket for multipart upload
std::string GetBucket();

// The encoding format of returned results.
std::string GetEncodingType();

// Object name
std::string GetKey();

// The ID of current multipart upload.
std::string GetUploadId();

// The information of the initiator of current upload
Initiator GetInitiator();

// The information of the owner of these chunks
Owner GetOwner();

// Entries are listed in UTF-8 binary order by default, starting from marker
uint64_t GetPartNumberMarker();

// Returns information of each chunk
std::vector<Part> GetParts();
```

```
// If the list of returned entries is truncated, NextMarker represents the starting point of the next entry
uint64_t GetNextPartNumberMarker();

// Storage class of the file chunks. Enumerated values: Standard, Standard_IA
std::string GetStorageClass();

// Maximum number of entries returned at a time
uint64_t GetMaxParts();

// Whether the list of returned entries is truncated. Boolean: TRUE, FALSE
bool IsTruncated();
```

Part, Owner and Initiator are composed as follows:

```
struct Initiator {
std::string m_id; // The unique identifier of creator
std::string m_display_name; // User name description of creator
};

struct Owner {
std::string m_id; // The unique identifier of user
std::string m_display_name; // User name description
};

struct Part {
uint64_t m_part_num; // File chunk number
uint64_t m_size; // File chunk size (in bytes)
std::string m_etag; // The MD5 algorithm check value of Object chunk
std::string m_last_modified; // The last modification time of file chunk
};
```

Example

```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);

std::string bucket_name = "cpp_sdk_v5-123456789";
std::string object_name = "test_object";

// uploadId is obtained by calling InitMultiUpload
qcloud_cos::ListPartsReq req(bucket_name, object_name, upload_id);
req.SetMaxParts(1);
req.SetPartNumberMarker("1");
qcloud_cos::ListPartsResp resp;
qcloud_cos::CosResult result = cos.ListParts(req, &resp);

// The call is successful. Call resp's member functions to obtain the response content
if (result.IsSucc()) {
// ...
```

```
} else {
// Failed to delete Object. Call CosResult's member functions to output error information, such as requestID, etc.
}
```

Put Object ACL

Feature description

This API (Put Object ACL) is used to write ACL for an Object. You can import ACL information either by using Header: "x-cos-acl", "x-cos-grant-read", "x-cos-grant-write", "x-cos-grant-full-control", or by using Body in XML format. For more information about this API, please see Put Object ACL.

Method prototype

```
CosResult PutObjectACL(const PutObjectACLReq& req, PutObjectACLResp* resp);
```

Parameters

Parameter	Description
req	PutObjectACLReq, the request for PutObjectACL operation.
resp	PutObjectACLResp, the response for PutObjectACL operation.

```
// Defines the ACL attribute of Object. Valid values: private, public-read-write, public-read.
// Default: private
void SetXCosAcl(const std::string& str);

// Grants read permission to the authorized user. Format: x-cos-grant-read: id=" ",id=" ".
// For authorization to a sub-account,id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>"
// For authorization to a root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>"
void SetXCosGrantRead(const std::string& str);

// Grants write permission to the authorized user. Format: x-cos-grant-write: id=" ",id=" "./
// For authorization to a sub-account,id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>",
// For authorization to a root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>"
void SetXCosGrantWrite(const std::string& str);

// Grants read and write permissions to the authorized user. Format: x-cos-grant-full-control: id=" ",id=" ".
// For authorization to a sub-account,id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>",
// For authorization to a root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>"
void SetXCosGrantFullControl(const std::string& str);

// Object owner ID
void SetOwner(const Owner& owner);

// Sets the information of authorized user and permissions
void SetAccessControlList(const std::vector<Grant>& grants);
```

```
// Adds the authorization information for a single Object
void AddAccessControlList(const Grant& grant);
```

> **Note:**
> APIs such as SetXCosAcl, SetXCosGrantRead, SetXCosGrantWrite, and SetXCosGrantFullControl cannot be used with SetAccessControlList and AddAccessControlList. This is because the former kind of APIs are implemented by setting HTTP Header, while the latter kind of APIs are implemented by adding content in XML format to the Body. You can only use either of the two kinds of APIs. The first kinds of APIs are preferred in SDK.

ACLRule is composed as follows:

```
struct Grantee {
// "type" can be RootAccount or SubAccount.
// If type is RootAccount, you can enter account ID in "uin" or in "uin" of id, or replace uin/<OwnerUin> and uin/<SubUin> with "anyone" (all types of users).
// If type is RootAcount, uin represents root account, and SubAccount represents sub-account.
std::string m_type;
std::string m_id; // qcs::cam::uin/<OwnerUin>:uin/<SubUin>
std::string m_display_name; // Optional
std::string m_uri;
};

struct Grant {
Grantee m_grantee; // Resource information of the authorized user
std::string m_perm; // Indicates the permission granted to the authorized user. Enumerated values: READ, WRITE, and FULL_CONTROL
};
```

Example

```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);

std::string bucket_name = "cpp_sdk_v5-123456789";
std::string object_name = "sevenyou";

// 1. Configures ACL (through Body). You can configure the ACL through either Body or Header. You can only use one of these two methods, otherwise a conflict occurs.
{
qcloud_cos::PutObjectACLReq req(bucket_name, object_name);
qcloud_cos::Owner owner = {"qcs::cam::uin/xxxxx:uin/xxx", "qcs::cam::uin/xxxxxx:uin/xxxxx" };
qcloud_cos::Grant grant;
req.SetOwner(owner);
grant.m_grantee.m_type = "Group";
grant.m_grantee.m_uri = "http://cam.qcloud.com/groups/global/AllUsers";
grant.m_perm = "READ";
```

```
req.AddAccessControlList(grant);

qcloud_cos::PutObjectACLResp resp;
qcloud_cos::CosResult result = cos.PutObjectACL(req, &resp);
// The call is successful. Call resp's member functions to obtain the response content
if (result.IsSucc()) {
// ...
} else {
// Sets the ACL. Call CosResult's member functions to output error information, such as requestID, etc.
}
}

// 2. Configures ACL (through Header). You can configure the ACL through either Body or Header. You can only u
se one of these two methods, otherwise a conflict occurs.
{
qcloud_cos::PutObjectACLReq req(bucket_name, object_name);
req.SetXCosAcl("public-read-write");

qcloud_cos::PutObjectACLResp resp;
qcloud_cos::CosResult result = cos.PutObjectACL(req, &resp);
// The call is successful. Call resp's member functions to obtain the response content
if (result.IsSucc()) {
// ...
} else {
// Sets the ACL. Call CosResult's member functions to output error information, such as requestID, etc.
}
}
```

Get Object ACL

Feature description

This API (Get Object ACL) is used to obtain the ACL (Access Control List) of an object (file). Only the Object owner has the access to this API. For more information about this API, please see Get Object ACL.

Method prototype

```
CosResult GetObjectACL(const DGetObjectACLReq& req, GetObjectACLResp* resp);
```

Parameters

Parameter	Description
req	GetObjectACLReq, the request for GetObjectACL operation.
resp	GetObjectACLResp, the response for GetObjectACL operation.

```
std::string GetOwnerID();
std::string GetOwnerDisplayName();
```

```
std::vector<Grant> GetAccessControlList();
```

Example

```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);

std::string Object_name = "cpp_sdk_v5-123456789";

The bucket_name is required in the constructor of GetObjectACLReq
qcloud_cos::GetObjectACLReq req(Object_name);
qcloud_cos::GetObjectACLResp resp;
qcloud_cos::CosResult result = cos.GetObjectACL(req, &resp);

// The call is successful. Call resp's member functions to obtain the response content
if (result.IsSucc()) {
// ...
} else {
// Failed to obtain the ACL. Call CosResult's member functions to output error information, such as requestID, etc.
}
```

Put Object Copy

Feature description

This API (Put Object Copy) is used to copy a file from source path to the destination path. In the process of copying, file meta-attributes and ACLs can be modified. You can use this API to move or rename a file, modify file attributes and create a copy. The recommended file size is 1MB-5GB. For any file greater than 5 GB, use multipart upload (Upload - Copy). For more information about this API, please see Put Object Copy.

Method prototype

```
CosResult PutObjectCopy(const PutObjectCopyReq& req, PutObjectCopyResp* resp);
```

Parameters

Parameter	Description
req	PutObjectCopyReq, the request for PutObjectCopy operation.
resp	PutObjectCopyResp, the response for PutObjectCopy operation.

```
// The path of source file URL. You can specify the history version with the versionid sub-resource
void SetXCosCopySource(const std::string& str);

// Indicates whether to copy metadata. Enumerated values: Copy, Replaced. Default is Copy.
// If it is marked as Copy, the file is copied directly, with the user metadata in header ignored;
// If it is marked as Replaced, the metadata is modified based on the Header information.
```

```
// If the destination path and the source path are the same (that is, the user attempts to modify the metadata), th
e value must be Replaced
void SetXCosMetadataDirective(const std::string& str);

// The operation is performed if the Object is modified after the specified time, otherwise error code 412 is return
ed.
// It can be used with x-cos-copy-source-If-None-Match. Using it with other conditions can cause a conflict.
void SetXCosCopySourceIfModifiedSince(const std::string& str);

// The action is performed if the Object has not been modified after the specified time, otherwise error code 412 i
s returned.
// It can be used with x-cos-copy-source-If-Match. Using it with other conditions can cause a conflict.
void SetXCosCopySourceIfUnmodifiedSince(const std::string& str);

// The operation is performed if the Etag of Object is the same as the given one, otherwise error code 412 is retur
ned.
// It can be used with x-cos-copy-source-If-Unmodified-Since. Using it with other conditions can cause a conflict.
void SetXCosCopySourceIfMatch(const std::string& str);

// The operation is performed if the Etag of Object is different from the given one, otherwise error code 412 is ret
urned.
// It can be used with x-cos-copy-source-If-Modified-Since. Using it with other conditions can cause a conflict.
void SetXCosCopySourceIfNoneMatch(const std::string& str);

// x-cos-storage-class, which is used to set the storage class of Object. Enumerated values: STANDARD, STANDAR
D_IA
// Default: STANDARD (supported only in South China region)
void SetXCosStorageClass(const std::string& storage_class);

// Defines the ACL attribute of Object. Valid values: private, public-read-write, public-read.
// Default: private
void SetXCosAcl(const std::string& str);

// Grants read permission to the authorized user. Format: x-cos-grant-read: id=" ",id=" ".
// For authorization to a sub-account, id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>"
// For authorization to a root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>"
void SetXCosGrantRead(const std::string& str);

// Grants write permission to the authorized user. Format: x-cos-grant-write: id=" ",id=" "./
//For authorization to a sub-accountid="qcs::cam::uin/<OwnerUin>:uin/<SubUin>",
// For authorization to a root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>"
void SetXCosGrantWrite(const std::string& str);

// Grants read and write permissions to the authorized user. Format: x-cos-grant-full-control: id=" ",id=" ".
// For authorization to a sub-account,id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>",
// For authorization to a root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>"
void SetXCosGrantFullControl(const std::string& str);

// The header information that can be defined by users, which is returned as Object metadata. The size is limited t
o 2 KB.
void SetXCosMeta(const std::string& key, const std::string& value);
```

```
/// Sets the algorithm used by server for encryption. AES256 is supported.
void SetXCosServerSideEncryption(const std::string& str);

// Returns the MD5 algorithm check value for the file. ETag value can be used to check whether the Object conte
nt has changed.
std::string GetEtag();

// Returns the last modification time of the file in GMT format
std::string GetLastModified();

// Returns the version number
std::string GetVersionId();

/// Algorithm used by server for encryption
std::string GetXCosServerSideEncryption();
```

Example

```
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);

std::string bucket_name = "cpp_sdk_v5-123456789";
std::string object_name = "sevenyou";

qcloud_cos::PutObjectCopyReq req(bucket_name, object_name);
req.SetXCosCopySource("sevenyousouthtest-12345656.cn-south.myqcloud.com/sevenyou_source_obj");
qcloud_cos::PutObjectCopyResp resp;
qcloud_cos::CosResult result = cos.PutObjectCopy(req, &resp);
```

Java SDK
Getting Started

Last updated：2018-09-18 22:08:18

Preparation for Development

Related resources

Download the XML Java SDK resources of COS service from XML Java SDK.

Environment dependencies

- The SDK supports JDK 1.7, 1.8 or above.
- For more information on how to install JDK, please see Java Installation and Configuration.

> For more information on the definitions of SecretId, SecretKey, Bucket and other terms and how to obtain them, please see COS Glossary.

Installing SDK

There are two ways to install the SDK: installation using maven and installation using source code.

- Installation using maven
  Add dependencies using pom.xml in maven project, as shown below:

```
<dependency>
<groupId>com.qcloud</groupId>
<artifactId>cos_api</artifactId>
<version>5.4.4</version>
</dependency>
```

- Installation using source code
  Download the source code from XML Java SDK and import it via maven. For example, for eclipse, select **File** -> **Import** -> **maven** -> **Existing Maven Projects**.

Uninstalling SDK

Uninstall the SDK by removing the pom dependencies or source code.

Getting Started

Initializing client cosclient

Set the user authentication information, including appid, the region where the bucket resides, and bucket name

```
// 1 Initialize user authentication information (secretId, secretKey)
COSCredentials cred = new BasicCOSCredentials("AKIDXXXXXXXX", "1A2Z3YYYYYYYYYY");
// 2 Set the region where the bucket resides. For the abbreviations of COS regions, please see https://cloud.tencent.com/document/product/436/6224
ClientConfig clientConfig = new ClientConfig(new Region("ap-beijing-1"));
// 3 Generate the COS client
COSClient cosclient = new COSClient(cred, clientConfig);
// The bucket name entered must be in a format of {name}-{appid}.
String bucketName = "mybucket-1251668577";
```

Uploading files

Upload a local file or input stream with a known length to COS. This is suitable for the upload of small files such as images below 20 MB. The maximum file size supported is 5 GB (inclusive). To upload a file greater than 5 GB, use multipart upload or advanced API. If most of your local files are greater than 20 MB, it is recommended to use an advanced API by referring to the relevant API document. If the object already exists on the COS, it will be overwritten. Because the directory does not exist in the Cloud Object Storage, if you want to create a directory object, see the FAQ in API document.

```
// Simple upload of file. The maximum file size supported is 5 GB. This is suitable for the upload of small files. It is recommended to use this API for the files less than 20 MB.
// To upload a large file, see "Upload Using Advanced API" in the API document.
File localFile = new File("src/test/resources/len5M.txt");
// Specify the object key to be uploaded to COS
// Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name `bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg`, the object key is doc1/pic1.jpg. For more information, please see [Object Key](https://cloud.tencent.com/document/product/436/13324).
String key = "upload_single_demo.txt";
PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName, key, localFile);
PutObjectResult putObjectResult = cosClient.putObject(putObjectRequest);
```

Downloading files

Download files to local machine or obtain the download input streams for downloading files.

```
// Specify the local path to which the file is downloaded
File downFile = new File("src/test/resources/mydown.txt");
// Specify the bucket where the file to be downloaded is located and the object key
GetObjectRequest getObjectRequest = new GetObjectRequest(bucketName, key);
ObjectMetadata downObjectMeta = cosClient.getObject(getObjectRequest, downFile);
```

Deleting files

Delete the files under the specified path on the COS.

```
// Specify the bucket and object key to be deleted
cosClient.deleteObject(bucketName, key);
```

Shutting down client

Shut down cosclient and release the backend thread (manager thread of HTTP connection).

```
// Shut down the client (close the backend thread)
cosClient.shutdown();
```

API Documentation

Last updated：2018-09-26 10:54:08

For a successful COS XML JAVA SDK operation, a specific type for each API is returned. For a failed operation, an exception (CosClientException and CosServiceException) is reported. CosClientException refers to client exceptions, such as network exceptions and request sending failure. CosServiceException contains the reasons why the client request is identified as a failure by the server. For example, you do not have such permission, or the file you want to access does not exist. For more information, please see Exception Types.
The following describes how to use each API in the SDK. For the sake of brevity, subsequent examples only illustrate how to use the API rather than how to capture exceptions.

```
try {
// The bucket name entered must be in a format of {name}-{appid}.
String bucketName = "movie-1251668577";
String key = "abc/def.jpg";
cosclient.deleteObject(bucket, key);
} catch (CosClientException cle) {
// Custom exception handling, such as print exceptions
log.error("del object failed.", cle);
// ...
} catch (CosServiceException cse) {
// Custom exception handling, such as print exceptions
log.error("del object failed.", cse);
// ...
}
```

Bucket API Description

Put Bucket

This API (Put Bucket) is used to create a Bucket. As a limited resource, Bucket is not the same as directory, and the number of files under a Bucket is unlimited, so it is recommended not to create too many Buckets. Generally, you do not need to create a Bucket frequently. You are advised to create a Bucket on the console and work with the Object using SDK.

- **Method prototype**

```
public Bucket createBucket(String bucketName) throws CosClientException, CosServiceException;
```

- **Parameter description**

Parameter Name	Description	Type

Parameter Name	Description	Type
bucketName	The bucket should be named in a format of {name}-{appid}, where name should be comprised of letters, numbers, and dashes.	String

- **Returned value**
  - Successful: Bucket type, including the description of Bucket (bucket name, owner, and creation date).
  - Failed: CosClientException or CosServiceException is thrown when an error (such as authentication failure) occurs. For more information, please see Exception Types. For more information, please see Exception Types.
- **Example**

```
// The bucket name entered must be in a format of {name}-{appid}.
String bucketName = "movie-1251668577";
Bucket bucket = cosClient.createBucket(bucketName);
```

Delete Bucket

This API (Delete Bucket) is used to delete the cleared Buckets.

- **Method prototype**

```
public void deleteBucket(String bucketName) throws CosClientException, CosServiceException;
```

- **Parameter description**

Parameter Name	Description	Type
bucketName	The bucket should be named in a format of {name}-{appid}, where name should be comprised of letters, numbers, and dashes.	String

- **Returned value**
  - Successful: No value is returned.
  - Failed: CosClientException or CosServiceException is thrown when an error (such as authentication failure) occurs. For more information, please see Exception Types.
- **Example**

```
// The bucket name entered must be in a format of {name}-{appid}.
String bucketName = "movie-1251668577";
cosClient.deleteBucket(bucketName);
```

Head Bucket

This API (Head Bucket) is used to query whether a bucket exists.

- **Method prototype**

```
public boolean doesBucketExist(String bucketName)
throws CosClientException, CosServiceException;
```

- **Parameter description**

Parameter Name	Description	Type
bucketName	The bucket should be named in a format of {name}-{appid}, where name should be comprised of letters, numbers, and dashes.	String

- **Returned value**
  - Successful: "true" is returned if the queried bucket exists. Otherwise, "false" is returned.
  - Failed: CosClientException or CosServiceException is thrown when an error (such as authentication failure) occurs. For more information, please see Exception Types.
- **Example**

```
// The bucket name entered must be in a format of {name}-{appid}.
String bucketName = "movie-1251668577";
boolean bucketExistFlag = cosClient.doesBucketExist(bucketName);
```

Get Bucket Location

This API (Get Bucket Location) is used to query the Region where a Bucket resides.

- **Method prototype**

```
public String getBucketLocation(String bucketName)
throws CosClientException, CosServiceException;
```

- **Parameter description**

Parameter Name	Description	Type
bucketName	The bucket should be named in a format of {name}-{appid}, where name should be comprised of letters, numbers, and dashes.	String

- **Returned value**
  - Successful: The Region where a Bucket resides is returned.
  - Failed: CosClientException or CosServiceException is thrown when an error (such as Bucket does not exist) occurs. For more information, please see Exception Types.
- **Example**

```
// The bucket name entered must be in a format of {name}-{appid}.
String location = cosClient.getBucketLocation("movie-1251668577");
```

Get Bucket (List Objects)

This API (Get Bucket) is used to obtain the file list on the COS.

- **Method prototype**

```
public ObjectListing listObjects(ListObjectsRequest listObjectsRequest)
throws CosClientException, CosServiceException;
```

- **Parameter description**

Parameter Name	Description	Type
listObjectsRequest	Request for obtaining a file list ListObjectsRequest	

Request member description:

Request Member	Setting Method	Description	Type
bucketName	Constructor or set method	The bucket should be named in a format of {name}-{appid}, where name should be comprised of letters, numbers, and dashes.	String
prefix	Constructor or set method	Marks members prefixed with "prefix" in the list. By default, no restriction is set. That is, all members under the Bucket are included. Default: ""	String

Request Member	Setting Method	Description	Type
marker	Constructor or the set method	Marks the starting position of the list. It is empty for the first time. Subsequently, the value returned by the previous list is the marker.	String
delimiter	Constructor or set method	Delimiter. It indicates that the path that starts with "prefix" and ends with delimiter for the first time will be returned.	String
maxKeys	Constructor or set method	The maximum number of returned members (less than 1,000). Default: 1,000	Integer

- **Returned value**
  - Successful: Return ObjectListing type, including all members and nextMarker.
  - Failed: CosClientException or CosServiceException is thrown. For more information, please see Exception Types.
- **Example**

```
// The bucket name entered must be in a format of {name}-{appid}.
String bucketName = "movie-1251668577";

// Obtain the bucket members (set delimiter)
ListObjectsRequest listObjectsRequest = new ListObjectsRequest();
listObjectsRequest.setBucketName(bucketName);
// Set a prefix for the list, indicating that all the file keys in the list start with this prefix
listObjectsRequest.setPrefix("");
// Set delimiter to /, indicating the direct members (excluding the recursive submembers) under the directory are obtained
listObjectsRequest.setDelimiter("/");
// Set marker (the first list marker is empty, and subsequent marker is obtained from the previous list)
listObjectsRequest.setMarker("");
// Set the maximum of number of members that can be listed to 100 (if not set, the default is 1,000, and a maximum of 1,000 keys can be listed at a time)
listObjectsRequest.setMaxKeys(100);

ObjectListing objectListing = cosClient.listObjects(listObjectsRequest);
// Obtain the marker for the next list
String nextMarker = objectListing.getNextMarker();
// Check whether the list is completed. If the list is completed, isTruncated is false, otherwise it is true.
boolean isTruncated = objectListing.isTruncated();
List<COSObjectSummary> objectSummaries = objectListing.getObjectSummaries();
for (COSObjectSummary cosObjectSummary : objectSummaries) {
// File path
String key = cosObjectSummary.getKey();
// Obtain file length
```

```
long fileSize = cosObjectSummary.getSize();
// Obtain file ETag
String eTag = cosObjectSummary.getETag();
// Obtain the last modification time
Date lastModified = cosObjectSummary.getLastModified();
// Obtain file storage type
String StorageClassStr = cosObjectSummary.getStorageClass();
}
```

Bucket ACL

This API (Set Bucket ACL) is used to set the access control list of the Bucket.

If Set Bucket ACL is performed, existing permission configuration will be overwritten.

ACL includes a predefined permission policy (CannedAccessControlList) or a custom permission control (AccessControlList). If both of them are set, the predefined policy will be ignored and the custom policy prevails. For more information on permissions, please see the permission section.

- **Method prototype**

```
// Method 1 (Set custom policy)
public void setBucketAcl(String bucketName, AccessControlList acl)
throws CosClientException, CosServiceException;
// Method 2 (Set predefined policy)
public void setBucketAcl(String bucketName, CannedAccessControlList acl) throws CosClientException, CosServi
ceException;
// Method 3 (Encapsulation of the two methods above, including setting of these two policies. If both of them ar
e set, the custom policy prevails.)
public void setBucketAcl(SetBucketAclRequest setBucketAclRequest)
throws CosClientException, CosServiceException;
```

- **Parameter description**

Parameters in Method 3 contain those in Method 1 and 2, so the following takes Method 3 as an example to describe these parameters.

Parameter Name	Description	Type
setBucketAclRequest	Indicates permission configuration request SetBucketAclRequest	

Request member description:

Request Member	Setting Method	Description	Type
bucketName	Constructor or set method	The bucket should be named in a format of {name}-{appid}, where name should be comprised of letters, numbers, and dashes.	String

Request Member	Setting Method	Description	Type
acl	Constructor or set method	Custom permission policy	AccessControlList
cannedAcl	Constructor or set method	Predefined policies, such as public read, public read and write, private read	CannedAccessControlList

- **Returned value**
  - Successful: No value is returned.
  - Failed: CosClientException or CosServiceException is thrown when an error (such as authentication failure) occurs. For more information, please see Exception Types.
- **Example**

```
// The bucket name entered must be in a format of {name}-{appid}.
String bucketName = "movie-1251668577";
// Set a custom ACL
AccessControlList acl = new AccessControlList();
Owner owner = new Owner();
owner.setId("qcs::cam::uin/2779643970:uin/2779643970");
acl.setOwner(owner);
String id = "qcs::cam::uin/2779643970:uin/734505014";
UinGrantee uinGrantee = new UinGrantee("qcs::cam::uin/2779643970:uin/734505014");
uinGrantee.setIdentifier(id);
acl.grantPermission(uinGrantee, Permission.FullControl);
cosclient.setBucketAcl(bucketName, acl);

// Set a predefined ACL
// Set private read and write (New buckets are configured with private read and write by default)
cosclient.setBucketAcl(bucketName, CannedAccessControlList.Private);
// Set public read and private write
cosclient.setBucketAcl(bucketName, CannedAccessControlList.PublicRead);
// Set public read and write
cosclient.setBucketAcl(bucketName, CannedAccessControlList.PublicReadWrite);
```

Get Bucket ACL

This API (Get Bucket ACL) is used to query the access policy ACL of a Bucket.

- **Method prototype**

```
public AccessControlList getBucketAcl(String bucketName)
throws CosClientException, CosServiceException
```

- **Parameter description**

Parameter Name	Description	Type
bucketName	The bucket should be named in a format of {name}-{appid}, where name should be comprised of letters, numbers, and dashes.	String

- **Returned value**
  - Successful: The ACL of a Bucket is returned.
  - Failed: CosClientException or CosServiceException is thrown when an error (such as authentication failure) occurs. For more information, please see Exception Types.
- **Example**

```
// The bucket name entered must be in a format of {name}-{appid}.
String bucketName = "movie-1251668577";
AccessControlList acl = cosclient.getBucketAcl(bucketName);
```

Set Bucket CORS

This API (Set Bucket CORS) is used to set cross-origin access rules for a bucket.

- **Method prototype**

```
public void setBucketCrossOriginConfiguration(String bucketName, BucketCrossOriginConfiguration bucketCrossOriginConfiguration);
```

- **Parameter description**

Parameter Name	Description	Type
bucketName	The bucket should be named in a format of {name}-{appid}, where name should be comprised of letters, numbers, and dashes.	String
bucketCrossOriginConfiguration	The cross-origin access rules set for a bucket BucketCrossOriginConfiguration	

- **Returned value**

- Successful: No value is returned.
- Failed: CosClientException or CosServiceException is thrown when an error (such as authentication failure) occurs. For more information, please see Exception Types.

- **Example**

```
// The bucket name entered must be in a format of {name}-{appid}.
String bucketName = "movie-1251668577";
BucketCrossOriginConfiguration bucketCORS = new BucketCrossOriginConfiguration();
List<CORSRule> corsRules = new ArrayList<>();
CORSRule corsRule = new CORSRule();
// Rule name
corsRule.setId("set-bucket-cors-test");
// Allowed HTTP method
corsRule.setAllowedMethods(AllowedMethods.PUT, AllowedMethods.GET, AllowedMethods.HEAD);
corsRule.setAllowedHeaders("x-cos-grant-full-control");
corsRule.setAllowedOrigins("http://mail.qq.com", "http://www.qq.com",
"http://video.qq.com");
corsRule.setExposedHeaders("x-cos-request-id");
corsRule.setMaxAgeSeconds(60);
corsRules.add(corsRule);
bucketCORS.setRules(corsRules);
cosclient.setBucketCrossOriginConfiguration(bucketName, bucketCORS);
```

Get Bucket CORS

This API (Get Bucket CORS) is used to obtain the cross-origin access rules of a bucket.

- **Method prototype**

```
public BucketCrossOriginConfiguration getBucketCrossOriginConfiguration(String bucketName)
throws CosClientException, CosServiceException;
```

- **Parameter description**

Parameter Name	Description	Type
bucketName	The bucket should be named in a format of {name}-{appid}, where name should be comprised of letters, numbers, and dashes.	String

- **Returned value**

- Successful: The cross-origin rules of a Bucket are returned.

- Failed: CosClientException or CosServiceException is thrown when an error (such as authentication failure) occurs. For more information, please see Exception Types.

- **Example**

```
// The bucket name entered must be in a format of {name}-{appid}.
String bucketName = "movie-1251668577";
BucketCrossOriginConfiguration corsGet = cosclient.getBucketCrossOriginConfiguration(bucketName);
List<CORSRule> corsRules = corsGet.getRules();
for (CORSRule rule : corsRules) {
List<AllowedMethods> allowedMethods = rule.getAllowedMethods();
List<String> allowedHeaders = rule.getAllowedHeaders();
List<String> allowedOrigins = rule.getAllowedOrigins();
List<String> exposedHeaders = rule.getExposedHeaders();
int maxAgeSeconds = rule.getMaxAgeSeconds();
}
```

Delete Bucket CORS

This API (Delete Bucket CORS) is used to delete the cross-origin access rules of a bucket.

- **Method prototype**

```
public void deleteBucketCrossOriginConfiguration(String bucketName)
throws CosClientException, CosServiceException;
```

- **Parameter description**

Parameter Name	Description	Type
bucketName	The bucket should be named in a format of {name}-{appid}, where name should be comprised of letters, numbers, and dashes.	String

- **Returned value**

  - Successful: No value is returned.

  - Failed: CosClientException or CosServiceException is thrown when an error (such as authentication failure) occurs. For more information, please see Exception Types.

- **Example**

```
// The bucket name entered must be in a format of {name}-{appid}.
String bucketName = "movie-1251668577";
```

```
cosclient.deleteBucketCrossOriginConfiguration(bucketName);
```

Set Bucket LifeCycle

This API (Set Bucket LifeCycle) is used to set the lifecycle rules of a bucket.

- **Method prototype**

```
public void setBucketLifecycleConfiguration(String bucketName, BucketLifecycleConfiguration bucketLifecycleConfiguration) throws CosClientException, CosServiceException;
```

- **Parameter description**

Parameter Name	Description	Type
bucketName	The bucket should be named in a format of {name}-{appid}, where name should be comprised of letters, numbers, and dashes.	String

- **Returned value**
  - Successful: No value is returned.
  - Failed: CosClientException or CosServiceException is thrown when an error (such as authentication failure) occurs. For more information, please see Exception Types.
- **Example**

```
List<Rule> rules = new ArrayList<>();
// Rule 1: Delete files whose paths start with hongkong_movie/ after 30 days
Rule deletePrefixRule = new Rule();
deletePrefixRule.setId("delete prefix xxxy after 30 days");
deletePrefixRule
.setFilter(new LifecycleFilter(newLifecyclePrefixPredicate("hongkong_movie/")));
// Delete files that have been uploaded or modified for 30 days
deletePrefixRule.setExpirationInDays(30);
// Set rules to active
deletePrefixRule.setStatus(BucketLifecycleConfiguration.ENABLED);

// Rule 2: Put files into low frequency after 20 days and delete them after one year
Rule standardIaRule = new Rule();
standardIaRule.setId("standard_ia transition");
standardIaRule.setFilter(new LifecycleFilter(new LifecyclePrefixPredicate("standard_ia/")));
List<Transition> standardIaTransitions = new ArrayList<>();
Transition standardTransition = new Transition();
standardTransition.setDays(20);
```

```
standardTransition.setStorageClass(StorageClass.Standard_IA.toString());
standardIaTransitions.add(standardTransition);
standardIaRule.setTransitions(standardIaTransitions);
standardIaRule.setStatus(BucketLifecycleConfiguration.ENABLED);
standardIaRule.setExpirationInDays(365);

// Add two rules to the policy set
rules.add(deletePrefixRule);
rules.add(standardIaRule);

// Generate bucketLifecycleConfiguration
BucketLifecycleConfiguration bucketLifecycleConfiguration =
new BucketLifecycleConfiguration();
bucketLifecycleConfiguration.setRules(rules);

// The bucket name entered must be in a format of {name}-{appid}.
String bucketName = "movie-1251668577";
SetBucketLifecycleConfigurationRequest setBucketLifecycleConfigurationRequest =
new SetBucketLifecycleConfigurationRequest(bucketName, bucketLifecycleConfiguration);

// Set the lifecycle
cosclient.setBucketLifecycleConfiguration(setBucketLifecycleConfigurationRequest);
```

Get Bucket LifeCycle

This API (Get Bucket LifeCycle) is used to obtain the lifecycle rules of a bucket.

- **Method prototype**

```
public BucketLifecycleConfiguration getBucketLifecycleConfiguration(String bucketName)
throws CosClientException, CosServiceException;
```

- **Parameter description**

Parameter Name	Description	Type
bucketName	The bucket should be named in a format of {name}-{appid}, where name should be comprised of letters, numbers, and dashes.	String

- **Returned value**
  - Successful: Return BucketLifecycleConfiguration type, including the lifecycle rules of the bucket.
  - Failed: CosClientException or CosServiceException is thrown when an error (such as authentication failure) occurs. For more information, please see Exception Types.

- **Example**

```
// The bucket name entered must be in a format of {name}-{appid}.
String bucketName = "movie-1251668577";
BucketLifecycleConfiguration queryLifeCycleRet =
cosclient.getBucketLifecycleConfiguration(bucketName);
List<Rule> ruleLists = queryLifeCycleRet.getRules();
```

Delete Bucket LifeCycle

This API (Delete Bucket LifeCycle) is used to delete the lifecycle rules of the cleared bucket.

- **Method prototype**

```
public void deleteBucketLifecycleConfiguration(String bucketName)
throws CosClientException, CosServiceException;
```

- **Parameter description**

Parameter Name	Description	Type
bucketName	The bucket should be named in a format of {name}-{appid}, where name should be comprised of letters, numbers, and dashes.	String

- **Returned value**
  - Successful: No value is returned.
  - Failed: CosClientException or CosServiceException is thrown when an error (such as authentication failure) occurs. For more information, please see Exception Types.
- **Example**

```
// The bucket name entered must be in a format of {name}-{appid}.
String bucketName = "movie-1251668577";
cosclient.deleteBucketLifecycleConfiguration(bucketName);
```

Object API Description

PUT Object (Upload Object)

Upload a local file or input stream with a known length to COS. This is suitable for the upload of small files such as images below 20 MB. The maximum file size supported is 5 GB (inclusive). To upload a file greater than 5 GB, use multipart upload. If the object already exists on the COS, it will be overwritten.

- **Method prototype**

```
// Method 1: Upload a local file to COS
public PutObjectResult putObject(String bucketName, String key, File file)
throws CosClientException, CosServiceException;
// Method 2: Upload an input stream to COS
public PutObjectResult putObject(String bucketName, String key, InputStream input,
ObjectMetadata metadata) throws CosClientException, CosServiceException;
// Method 3: Encapsulate the two methods above to support more fine-grained parameter control, such as content-type and content-disposition
public PutObjectResult putObject(PutObjectRequest putObjectRequest)
throws CosClientException, CosServiceException;
```

- **Parameter description**

Parameter Name	Description	Type
putObjectRequest	File upload request	PutObjectRequest

Request member description:

Request Member	Setting Method	Description	Type
bucketName	Constructor or set method	The bucket should be named in a format of {name}-{appid}, where name should be comprised of letters, numbers, and dashes.	String
key	Constructor or set method	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name `bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg`, the object key is doc1/pic1.jpg. For more information, please see Object Key	String
file	Constructor or set method	Local file	File
input	Constructor or set method	Input stream	InputStream

Request Member	Setting Method	Description	Type
metadata	Constructor or set method	Meta information of a file	ObjectMetadata

- **Returned value**
  - Successful: PutObjectResult, including the file ETag and other information.
  - Failed: CosClientException or CosServiceException is thrown when an error (such as authentication failure) occurs. For more information, please see Exception Types.
- **Example**

```
// The bucket name entered must be in a format of {name}-{appid}.
String bucketName = "movie-1251668577";
// Method 1: Upload a local file
File localFile = new File("/data/test.txt");
String key = "aaa.txt";
PutObjectResult putObjectResult = cosClient.putObject(bucketName, key, localFile);
// Obtain file ETag

// Method 2: Upload an input stream (The length of a input stream must be known in advance. Otherwise, it may cause OOM)
FileInputStream fileInputStream = new FileInputStream(localFile);
ObjectMetadata objectMetadata = new ObjectMetadata();
// Set the length of an input stream to 500
objectMetadata.setContentLength(500);
// Set Content type. Default is application/octet-stream
objectMetadata.setContentType("application/pdf");
PutObjectResult putObjectResult = cosClient.putObject(bucketName, key, fileInputStream, objectMetadata);
String etag = putObjectResult.getETag();
// Close the input stream...

// Method 3: Provide more fine-grained control. Common settings are as follows
// 1. storage-class storage type, including standard (default), low frequency, and nearline
// 2. content-type. For local file upload, the files are mapped based on the suffix by default. For example, the jpg file is mapped //as image/jpeg
// For input stream upload, the default is application/octet-stream
// 3. Set permissions when uploading (or by calling API set object ACL)
File localFile = new File("/data/dog.jpg");
String key = "mypic.jpg";
PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName, key, file);
// Set the storage type to low frequency
putObjectRequest.setStorageClass(StorageClass.Standard_IA);
```

```
// Set custom attributes (such as content-type, content-disposition)
ObjectMetadata objectMetadata = new ObjectMetadata();
// Set Content type. Default is application/octet-stream
objectMetadata.setContentType("image/jpeg");
putObjectRequest.setMetadata(objectMetadata);
PutObjectResult putObjectResult = cosClient.putObject(putObjectRequest);
String etag = putObjectResult.getETag(); // Obtain file ETag
```

- **Multipart upload**

Multipart upload is to split a file into multiple parts for upload. A maximum of 40 TB of parts can be uploaded concurrently.

The steps of multipart upload are as follows:

1. Initialize multipart upload, and obtain uploadid. (initiateMultipartUpload)
2. Upload data in parts (concurrently). (uploadPart)
3. Complete multipart upload. (completeMultipartUpload)

In addition, you can also obtain the uploaded parts (listParts) or terminate multipart upload (abortMultipartUpload). Multipart upload has complicated steps and high requirements for use, so it is recommended to use the following advanced and encapsulated API for upload.

- **Method prototype**

```
// Method 1: Initialize multipart upload
public InitiateMultipartUploadResult initiateMultipartUpload(
InitiateMultipartUploadRequest request) throws CosClientException, CosServiceException;
// Method 2: Upload data in parts
public UploadPartResult uploadPart(UploadPartRequest uploadPartRequest)
throws CosClientException, CosServiceException;
// Method 3: Complete multipart upload
public CompleteMultipartUploadResult completeMultipartUpload(
CompleteMultipartUploadRequest request) throws CosClientException, CosServiceException;
// Method 4: List uploaded parts
public PartListing listParts(ListPartsRequest request)
throws CosClientException, CosServiceException;
// Method 5: Terminate multipart upload
public void abortMultipartUpload(AbortMultipartUploadRequest request)
throws CosClientException, CosServiceException;
```

- **Returned value**
  - **Method 1 (initiateMultipartUpload)**
    - Successful: Return InitiateMultipartUploadResult type, including the upload ID required for subsequent multipart upload.

- Failed: CosClientException or CosServiceException is thrown when an error (such as authentication failure) occurs. For more information, please see Exception Types.

- **Method 2 (uploadPart)**
  - Successful: Return UploadPartResult, including the Etag and partNumber of the part.
  - Failed: CosClientException or CosServiceException is thrown when an error (such as authentication failure) occurs. For more information, please see Exception Types.
- **Method 3 (completeMultipartUpload)**
  - Successful: Return CompleteMultipartUploadResult, including the Etag of the entire file.
  - Failed: CosClientException or CosServiceException is thrown when an error (such as authentication failure) occurs. For more information, please see Exception Types.
- **Method 4 (listParts)**
  - Successful: Return PartListing, including the ETag and No. of each part as well as the starting marker of the next list.
  - Failed: CosClientException or CosServiceException is thrown when an error (such as authentication failure) occurs. For more information, please see Exception Types.
- **Method 5 (abortMultipartUpload)**
  - Successful: No value is returned.
  - Failed: CosClientException or CosServiceException is thrown when an error (such as authentication failure) occurs. For more information, please see Exception Types.

- **Example**

```
// The bucket name entered must be in a format of {name}-{appid}.
String bucketName = "movie-1251668577";
// Initialize multipart upload
InitiateMultipartUploadRequest initRequest =
new InitiateMultipartUploadRequest(bucketName, key);
InitiateMultipartUploadResult initResponse = cosClient.initiateMultipartUpload(initRequest);
String uploadId = initResponse.getUploadId()
// Upload parts. A maximum of 1,000 parts can be uploaded concurrently. The size of each part should be betwee
n 1 MB and 5 GB.
// Set the size of each part to 4 MB. If there are a total of n parts, the size of part 1 to part n-1 is the same, and the
```

```
last part is less than or equal to the size of previous parts.
List<PartETag> partETags = new ArrayList<PartETag>();
int partNumber = 1;
int partSize = 4 * 1024 * 1024;
// partStream represents the input stream of the part's data. The length of input stream is partSize.
UploadPartRequest uploadRequest = new UploadPartRequest().withBucketName(bucketName).
withUploadId(uploadId).withKey(key).withPartNumber(partNumber).
withInputStream(partStream).withPartSize(partSize);
UploadPartResult uploadPartResult = cosClient.uploadPart(uploadRequest);
String eTag = uploadPartResult.getETag(); // Obtain the Etag of the part
partETags.add(new PartETag(partNumber, eTag)); // partETags records Etag information of all uploaded parts
partETags.add(new PartETag(partNumber, eTag)); // Upload parts with the partNumber of 2 to n

// Complete multipart upload.
CompleteMultipartUploadRequest compRequest = new CompleteMultipartUploadRequest(bucketName, key,
uploadId, partETags);
CompleteMultipartUploadResult result = cosClient.completeMultipartUpload(compRequest);

// ListPart is used to obtain the information of the uploaded part based on uploadId before the multipart upload
is completed or aborted. It can be used to construct partEtags.
ListPartsRequest listPartsRequest = new ListPartsRequest(bucket, key, uploadId);
do {
partListing = cosclient.listParts(listPartsRequest);
for (PartSummary partSummary : partListing.getParts()) {
partETags.add(new PartETag(partSummary.getPartNumber(), partSummary.getETag()));
}
listPartsRequest.setPartNumberMarker(partListing.getNextPartNumberMarker());
} while (partListing.isTruncated());

// abortMultipartUpload is used to abort an uncompleted multipart upload
AbortMultipartUploadRequest abortMultipartUploadRequest =
new AbortMultipartUploadRequest(bucket, key, uploadId);
cosclient.abortMultipartUpload(abortMultipartUploadRequest);
```

Get Object

Download files to local machine or obtain the download input streams for downloading files.

- **Method prototype**

```
// The bucket name entered must be in a format of {name}-{appid}.
String bucketName = "movie-1251668577";
// Method 1: Download the file and get the input stream
public COSObject getObject(GetObjectRequest getObjectRequest)
throws CosClientException, CosServiceException;
// Method 2: Download the file locally
public ObjectMetadata getObject(GetObjectRequest getObjectRequest, File destinationFile)
throws CosClientException, CosServiceException;
```

- **Parameter description**

Parameter Name	Description	Type
getObjectRequest	File download request GetObjectRequest	
destinationFile	File stored locally	File

Request member description:

Request Member	Setting Method	Description	Type
bucketName	Constructor or set method	The bucket should be named in a format of {name}-{appid}, where name should be comprised of letters, numbers, and dashes.	String
key	Constructor or set method	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name `bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg` , the object key is doc1/pic1.jpg. For more information, please see Object Key	String
range	Set method	The range of download	Long[]

- **Returned value**
  - **Method 1 (Get the input stream of the downloaded file)**
    - Successful: Return COSObject type, including the input stream and file attributes.
    - Failed: CosClientException or CosServiceException is thrown when an error (such as authentication failure) occurs. For more information, please see Exception Types.
  - **Method 2 (Download the file locally)**
    - Successful: Return the file attribute objectMetadata, including the file's custom header, content-type and other attributes.
    - Failed: CosClientException or CosServiceException is thrown when an error (such as authentication failure) occurs. For more information, please see Exception Types.
- **Example**

```
// The bucket name entered must be in a format of {name}-{appid}.
String bucketName = "movie-1251668577";
String key = "abc/def.jpg";
// Method 1: Get the input stream of the downloaded file
```

```
GetObjectRequest getObjectRequest = new GetObjectRequest(bucketName, key);
COSObject cosObject = cosClient.getObject(getObjectRequest);
COSObjectInputStream cosObjectInput = cosObject.getObjectContent();

// Method 2: Download the file locally
File downFile = new File("src/test/resources/mydown.txt");
GetObjectRequest getObjectRequest = new GetObjectRequest(bucketName, key);
ObjectMetadata downObjectMeta = cosClient.getObject(getObjectRequest, downFile);
```

Delete Object

This API (Delete Object) is used to delete files on COS.

- **Method prototype**

```
// Delete files
public void deleteObject(String bucketName, String key)
throws CosClientException, CosServiceException;
```

- **Parameter description**

Parameter Name	Description	Type	
bucketName	The bucket should be named in a format of {name}-{appid}, where name should be comprised of letters, numbers, and dashes.	String	
key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name `bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg` , the object key is doc1/pic1.jpg. For more information, please see Object Key	String	.

- **Returned value**
  - Successful: No value is returned.
  - Failed: CosClientException or CosServiceException is thrown when an error (such as authentication failure) occurs. For more information, please see Exception Types.
- **Example**

```
// Delete files on COS
// The bucket name entered must be in a format of {name}-{appid}.
String bucketName = "movie-1251668577";
String key = "abc/def.jpg";
cosclient.deleteObject(bucket, key);
```

Head Object

This API (Head Object) is used to get the attributes of a file on COS.

- **Method prototype**

```
// Obtain file attributes
public ObjectMetadata getObjectMetadata(String bucketName, String key)
throws CosClientException, CosServiceException;
```

- **Parameter description**

Parameter Name	Description	Type	
bucketName	The bucket should be named in a format of {name}-{appid}, where name should be comprised of letters, numbers, and dashes.	String	
key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg , the object key is doc1/pic1.jpg. For more information, please see Object Key	String	.

- **Returned value**
  - Successful: No value is returned.
  - Failed: CosClientException or CosServiceException is thrown when an error (such as authentication failure) occurs. For more information, please see Exception Types.
- **Example**

```
// Obtain the attributes of a file on COS
// The bucket name entered must be in a format of {name}-{appid}.
String bucketName = "movie-1251668577";
String key = "abc/def.jpg";
ObjectMetadata objectMetadata = cosclient.getObjectMetadata(bucketName, key);
```

Put Object Copy

This API (Put Object Copy) is used to copy the Object to a new path or bucket. It supports cross-region/account/bucket copy, and needs the read permission to the source file and the write permission to the destination file. The maximum allowed file size is 5 GB. To copy files larger than 5 GB, use the advanced API.

- **Method prototype**

```
// Obtain file attributes
public CopyObjectResult copyObject(CopyObjectRequest copyObjectRequest)
throws CosClientException, CosServiceException
```

- **Parameter description**

Parameter Name	Description	Type
copyObjectRequest	File copy request	CopyObjectRequest

Request member description:

Parameter Name	Description	Type
sourceBucketRegion	Region of the source Bucket. Default: same with the region of the current clientconfig, which represents an intra-region copy	String
sourceBucketName	Source Bucket name. The bucket should be named in a format of {name}-{appid}, where name should be comprised of letters, numbers, and dashes.	String
sourceKey	Source object key. Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name `bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg`, the object key is doc1/pic1.jpg. For more information, please see Object Key	String
sourceVersionId	Version ID of the source file with multiple versions. Default: The latest version of the source file	String
destinationBucketName	Destination Bucket name. The bucket should be named in a format of {name}-{appid}, where name should be comprised of letters, numbers, and dashes.	String
destinationKey	Destination object key. Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name `bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg`, the object key is doc1/pic1.jpg. For more information, please see Object Key	String
storageClass	The storage type of the copied destination file (standard, low frequency, nearline). Default: Standard	String

- **Returned value**
  - Successful: Return CopyObjectResult, including Etag and other information of the new file.

- Failed: CosClientException or CosServiceException is thrown when an error (such as authentication failure) occurs. For more information, please see Exception Types.

- **Example**

```
// Intra-region copy with the same account
// Source bucket. The bucket entered must be in a format of {name}-{appid}
String srcBucketName = "srcBucket-1251668577";
// The source file to copy
String srcKey = "aaa/bbb.txt";
// Destination bucket. The bucket entered must be in a format of {name}-{appid}
String destBucketName = "destBucket-1251668577";
// The destination file to copy
String destKey = "ccc/ddd.txt";
CopyObjectRequest copyObjectRequest = new CopyObjectRequest(srcBucketName, srcKey, destBucketName, des
tKey);
CopyObjectResult copyObjectResult = cosclient.copyObject(copyObjectRequest);

// Cross-account and cross-region copy (The read permission to source file and the write permission to destinatio
n file are required)
String srcBucketNameOfDiffAppid = "srcBucket-125100000";
Region srcBucketRegion = new Region("ap-shanghai");
copyObjectRequest = new CopyObjectRequest(srcBucketRegion, srcBucketNameOfDiffAppid, srcKey, destBucket
Name, destKey);
copyObjectResult = cosclient.copyObject(copyObjectRequest);
```

Set Object ACL

This API (Set Object ACL) is used to set the Access Control List of the Object. If Set Bucket ACL is performed, existing permission configuration will be overwritten.
The object inherits the bucket permission by default. COS only supports setting 1,000 ACL rules for an account (appid), so it is recommended to set ACL only for objects with inconsistent bucket permission to prevent the number of permission from exceeding the threshold. The function of unlimited ACL is under development.

ACL includes a predefined permission policy (CannedAccessControlList) or a custom permission control (AccessControlList). If both of them are set, the predefined policy will be ignored and the custom policy prevails. For more information on permissions, please see the permission section.

- **Method prototype**

```
// Method 1 (Set custom policy)
public void setObjectAcl(String bucketName, String key, AccessControlList acl)
throws CosClientException, CosServiceException
Method 2 (Set predefined policy)
public void setObjectAcl(String bucketName, String key, CannedAccessControlList acl)
throws CosClientException, CosServiceException
Method 3 (Encapsulation of the two methods above, including setting of these two policies. If both of them are
```

```
set, the custom policy prevails.)
public void setObjectAcl(SetObjectAclRequest setObjectAclRequest)
throws CosClientException, CosServiceException;
```

- **Parameter description**
  - Parameters in **Method 3** contain those in Method 1 and 2, so the following takes Method 3 as an example to describe these parameters.

Parameter Name	Description	Type
SetObjectAclRequest	Request type	setObjectAclRequest

Request member description:

Request Member	Setting Method	Description	Type
bucketName	Constructor or set method	The bucket should be named in a format of {name}-{appid}, where name should be comprised of letters, numbers, and dashes.	String
key	Constructor or set method	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name `bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg` , the object key is doc1/pic1.jpg. For more information, please see Object Key	String
acl	Constructor or set method	Custom permission policy	AccessControlList
cannedAcl	Constructor or set method	Predefined policies, such as public read, public read and write, private read	CannedAccessControlList

- **Returned value**
  - Successful: No value is returned.
  - Failed: CosClientException or CosServiceException is thrown when an error (such as authentication failure) occurs. For more information, please see Exception Types.
- **Example**

```
// The bucket name entered must be in a format of {name}-{appid}.
String bucketName = "movie-1251668577";
String key = "abc/def.jpg";
// Set a custom ACL
AccessControlList acl = new AccessControlList();
Owner owner = new Owner();
owner.setId("qcs::cam::uin/2779643970:uin/2779643970");
acl.setOwner(owner);
String id = "qcs::cam::uin/2779643970:uin/734505014";
UinGrantee uinGrantee = new UinGrantee("qcs::cam::uin/2779643970:uin/734505014");
uinGrantee.setIdentifier(id);
acl.grantPermission(uinGrantee, Permission.FullControl);
cosclient.setObjectAcl(buckeName, key, acl);

// Set a predefined ACL
// Set private read and write (Object integrates permissions of Bucket by default)
cosclient.setObjectAcl(buckeName, key, CannedAccessControlList.Private);
// Set public read and private write
cosclient.setObjectAcl(buckeName, key, CannedAccessControlList.PublicRead);
// Set public read and write
cosclient.setObjectAcl(buckeName, key, CannedAccessControlList.PublicReadWrite);
```

Get Object ACL

This API (Get Object ACL) is used to query the ACL of an Object.

- **Method prototype**

```
public AccessControlList getObjectAcl(String bucketName, String key)
throws CosClientException, CosServiceException;
```

- **Parameter description**

Parameter Name	Description	Type	
bucketName	The bucket should be named in a format of {name}-{appid}, where name should be comprised of letters, numbers, and dashes.	String	
key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg , the object key is doc1/pic1.jpg. For more information, please see Object Key	String	.

- **Returned value**

  - Successful: Return the ACL to which the Object resides.

- Failed: CosClientException or CosServiceException is thrown when an error (such as authentication failure) occurs. For more information, please see Exception Types.

- **Example**

```
// The bucket name entered must be in a format of {name}-{appid}.
String bucketName = "movie-1251668577";
String key = "abc/def.jpg";
AccessControlList acl = cosclient.getObjectAcl(bucketName, key);
```

Generating a Pre-Signed URL

In COSClient, you can construct or generate a pre-signed URL that can be distributed to clients for download or upload.

- **Method prototype**

```
// Construct a pre-signed URL
public URL generatePresignedUrl(GeneratePresignedUrlRequest req) throws CosClientException
```

- **Parameter description**

Parameter Name	Description	Type
req	Pre-signed request	GeneratePresignedUrlRequest

Request member description:

Request Member	Setting Method	Description	Type
method	Constructor or set method	HTTP method. Available values: PUT, GET, and DELETE	HttpMethodName
bucketName	Constructor or set method	The bucket should be named in a format of {name}-{appid}, where name should be comprised of letters, numbers, and dashes.	String

Request Member	Setting Method	Description	Type
key	Constructor or set method	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name `bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg`, the object key is doc1/pic1.jpg. For more information, please see Object Key	String
expiration	set method	Signature expiration time	Date
contentType	set method	Content-Type in the request that needs a signature	String
contentMd5	set method	Content-Md5 in the request that needs a signature	String
responseHeaders	set method	The returned http header to be overridden in the request to download a signature	ResponseHeaderOverrides

- **Returned value**

URL

- **Example**

Example 1. Generate a signed download URL. The sample code is as follows:

```
// Generate a download URL
// The bucket name entered must be in a format of {name}-{appid}.
String bucketName = "mybucket-1251668577";
String key = "aaa.txt";
GeneratePresignedUrlRequest req =
new GeneratePresignedUrlRequest(bucketName, key, HttpMethodName.GET);
// Set the signature expiration time (optional). If it is not set, the signature expiration time (5 minutes) in ClientCo
nfig is used by default.
// The set signature will expire in half an hour
Date expirationDate = new Date(System.currentTimeMillis() + 30L * 60L * 1000L);
req.setExpiration(expirationDate);
URL downloadUrl = cosclient.generatePresignedUrl(req);
String downloadUrlStr = downloadUrl.toString();
```

Example 2. Generate a signed download URL and set to override the public headers (such as content-type and content-language) to be returned. The sample code is as follows:

```
// The bucket should be named in a format of {name}-{appid}, where name should be comprised of letters, numb
ers, and dashes.
```

```
String bucketName = "mybucket-1251668577";
String key = "aaa.txt";
GeneratePresignedUrlRequest req =
new GeneratePresignedUrlRequest(bucketName, key, HttpMethodName.GET);
// Set the http header returned during download
ResponseHeaderOverrides responseHeaders = new ResponseHeaderOverrides();
String responseContentType = "image/x-icon";
String responseContentLanguage = "zh-CN";
String responseContentDispositon = "filename=\"abc.txt\"";
String responseCacheControl = "no-cache";
String cacheExpireStr =
DateUtils.formatRFC822Date(new Date(System.currentTimeMillis() + 24L * 3600L * 1000L));
responseHeaders.setContentType(responseContentType);
responseHeaders.setContentLanguage(responseContentLanguage);
responseHeaders.setContentDisposition(responseContentDispositon);
responseHeaders.setCacheControl(responseCacheControl);
responseHeaders.setExpires(cacheExpireStr);
req.setResponseHeaders(responseHeaders);
// Set the signature expiration time (optional). If it is not set, the signature expiration time (5 minutes) in ClientCo
nfig is used by default.
// The set signature will expire in half an hour
Date expirationDate = new Date(System.currentTimeMillis() + 30L * 60L * 1000L);
req.setExpiration(expirationDate);
URL url = cosclient.generatePresignedUrl(req);
```

Example 3. Generate public buckets (anonymous and readable) that do not need signed URLs. The sample code is as follows:

```
// To generate an anonymous request signature, you need to reinitialize an anonymous COSClient
// 1. Initialize a user's identity information. For an anonymous identity, ak sk does not need to be input
COSCredentials cred = new AnonymousCOSCredentials();
// 2. Set the region where the bucket resides. For the abbreviations of COS regions, please see https://cloud.tence
nt.com/document/product/436/6224
ClientConfig clientConfig = new ClientConfig(new Region("ap-beijing-1"));
// 3. Generate the COS client
COSClient cosclient = new COSClient(cred, clientConfig);
// The bucket name must contain appid
String bucketName = "mybucket-1251668577";

String key = "aaa.txt";
GeneratePresignedUrlRequest req =
new GeneratePresignedUrlRequest(bucketName, key, HttpMethodName.GET);
URL url = cosclient.generatePresignedUrl(req);

System.out.println(url.toString());

cosclient.shutdown();
```

Example 4. Generate some pre-signed upload URLs that can be directly distributed to the client for file uploads. The sample code is as follows:

```
// The bucket name entered must be in a format of {name}-{appid}.
String bucketName = "mybucket-1251668577";

String key = "aaa.txt";
// Set the signature expiration time (optional). If it is not set, the signature expiration time (5 minutes) in ClientCo
nfig is used by default.
// The set signature will expire in half an hour
Date expirationTime = new Date(System.currentTimeMillis() + 30L * 60L * 1000L);
URL url = cosclient.generatePresignedUrl(bucketName, key, expirationTime, HttpMethodName.PUT);
```

Generating Signature

COSSigner provides a method to construct a COS signature which can be distributed to mobile SDKs for upload and download.
The signature path matches the key to be operated after distribution.

- **Method prototype**

```
// Construct COS signature
public String buildAuthorizationStr(HttpMethodName methodName, String resouce_path,
COSCredentials cred, Date expiredTime);

// Construct COS signature
// Compared with first method, the second method provides additional signatures for some HTTP Headers and all
parameters in the entered URL.
// It is used for more complicated signature control. The generated signature must also carry the corresponding h
eader and param for upload and download operations.
public String buildAuthorizationStr(HttpMethodName methodName, String resouce_path,
Map<String, String> headerMap, Map<String, String> paramMap, COSCredentials cred,
Date expiredTime);
```

- **Parameter description**

Parameter Name	Description	Type
methodName	HTTP request method. Available values: PUT, GET, DELETE, HEAD, and POST	HttpMethodName
resouce_path	The path that needs a signature, starting with "/". Same as the key for file upload.	HttpMethodName
cred	Key information	COSCredentials
expiredTime	Expiration time	Date

Parameter Name	Description	Type
headerMap	The HTTP Header map that needs a signature. Perform the signature only for entered Content-Type, Content-Md5, and header starting with "x"	Map
paramMap	URL Param map that needs a signature	Map

- **Returned value**

String

- **Example**

Example 1 Generate a upload signature

```
// The bucket name entered must be in a format of {name}-{appid}.
String bucketName = "mybucket-1251668577";
COSCredentials cred = new BasicCOSCredentials("AKIDXXXXXXXX", "1A2Z3YYYYYYYYYY");
COSSigner signer = new COSSigner();
// Set the expiration time as 1 hour
Date expiredTime = new Date(System.currentTimeMillis() + 3600L * 1000L);
// The key that needs signature. The generated signature can only be used for uploading for the corresponding k
ey.
String key = "/aaa.txt";
String sign = signer.buildAuthorizationStr(HttpMethodName.PUT, key, cred, expiredTime);
```

Example 2 Generate a download signature

```
// The bucket name entered must be in a format of {name}-{appid}.
String bucketName = "mybucket-1251668577";
COSCredentials cred = new BasicCOSCredentials("AKIDXXXXXXXX", "1A2Z3YYYYYYYYYY");
COSSigner signer = new COSSigner();
// Set the expiration time as 1 hour
Date expiredTime = new Date(System.currentTimeMillis() + 3600L * 1000L);
// The key that needs signature. The generated signature can only be used for downloading for the correspondin
g key.
String key = "/aaa.txt";
String sign = signer.buildAuthorizationStr(HttpMethodName.GET, key, cred, expiredTime);
```

Example 3 Generate a deletion signature

```
// The bucket name entered must be in a format of {name}-{appid}.
String bucketName = "mybucket-1251668577";
COSCredentials cred = new BasicCOSCredentials("AKIDXXXXXXXX", "1A2Z3YYYYYYYYYY");
COSSigner signer = new COSSigner();
// Set the expiration time as 1 hour
Date expiredTime = new Date(System.currentTimeMillis() + 3600L * 1000L);
// The key that needs signature. The generated signature can only be used for deletion for the corresponding ke
```

```
y.
String key = "/aaa.txt";
String sign = signer.buildAuthorizationStr(HttpMethodName.DELETE, key, cred, expiredTime);
```

File Upload via Advanced API (Recommended)

The advanced API encapsulates upload and download APIs using the TransferManger type. It has a thread pool inside to accept users' upload and download requests, so users can choose to submit tasks asynchronously.

```
ExecutorService threadPool = Executors.newFixedThreadPool(32);
// Input a threadpool. If no thread pool is input, TransferManager will generate a single-thread pool by default.
TransferManager transferManager = new TransferManager(cosclient, threadPool);
// ..... (submit upload/download requests as described below)
// Close TransferManger
transferManager.shutdownNow();
```

Uploading File

The upload API automatically selects simple upload or multipart upload based on the size of users' files, making it easy for users to use. Besides, you do not need to care about each step of the multipart upload.

- **Method prototype**

```
// Upload files
public Upload upload(final PutObjectRequest putObjectRequest)
throws CosServiceException, CosClientException;
```

- **Parameter description**

Parameter Name	Description	Type
putObjectRequest	File upload request	PutObjectRequest

Request member description:

Request Member	Setting Method	Description	Type
bucketName	Constructor or set method	The bucket should be named in a format of {name}-{appid}, where name should be comprised of letters, numbers, and dashes.	String

Request Member	Setting Method	Description	Type
key	Constructor or set method	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name `bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg` , the object key is doc1/pic1.jpg. For more information, please see Object Key	String
file	Constructor or set method	Local file	File
input	Constructor or set method	Input stream	InputStream
metadata	Constructor or set method	Meta information of a file	ObjectMetadata

- **Returned value**
  - Successful: Return Upload. You can query whether the upload is completed, or wait until the upload finishes synchronously.
  - Failed: CosClientException or CosServiceException is thrown when an error (such as authentication failure) occurs. For more information, please see Exception Types.
- **Example**

```
// The bucket name entered must be in a format of {name}-{appid}.
String bucketName = "movie-1251668577";
String key = "/mypic.jpg";
File localFile = new File("/data/dog.jpg");
PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName, key, file);
// Local file upload
Upload upload = transferManager.upload(putObjectRequest);
// Wait for the transfer to finish (call waitForCompletion if you want to wait for the upload to finish synchronousl
y)
UploadResult uploadResult = upload.waitForUploadResult();
```

Downloading File

Download files on COS locally.

- **Method prototype**

```
// Download files
public Download download(final GetObjectRequest GetObjectRequest, final File file);
```

- **Parameter description**

Parameter Name	Description	Type
getObjectRequest	File download request GetObjectRequest	
file	File to be downloaded locally	File

Request member description:

Request Member	Setting Method	Description	Type
bucketName	Constructor or set method	The bucket should be named in a format of {name}-{appid}, where name should be comprised of letters, numbers, and dashes.	String
key	Constructor or set method	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name `bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg` , the object key is doc1/pic1.jpg. For more information, please see Object Key	String
range	Set method	The range of download	Long[]

- **Returned value**
  - Successful: Return Download. You can query whether the download is completed, or wait until the download finishes synchronously.
  - Failed: CosClientException or CosServiceException is thrown when an error (such as authentication failure) occurs. For more information, please see Exception Types.
- **Example**

```
// The bucket name entered must be in a format of {name}-{appid}.
String bucketName = "movie-1251668577";
String key = "/mypic.jpg";
File localDownFile = new File("/data/dog.jpg");
GetObjectRequest getObjectRequest = new GetObjectRequest(bucket, key);
// Download files
Download download = transferManager.download(getObjectRequest, localDownFile);
// Wait for the transfer to finish (call waitForCompletion if you want to wait for the upload to finish synchronousl
```

```
y)
download.waitForCompletion();
```

Copying File

The copy API automatic selects copy or multipart copy based on file size. Users do not need to care about the size of the file to be copied.

- **Method prototype**

```
// Upload files
public Copy copy(final CopyObjectRequest copyObjectRequest);
```

- **Parameter description**

Parameter Name	Description	Type
copyObjectRequest	File copy request	CopyObjectRequest

Request member description:

Parameter Name	Description	Type	
sourceBucketRegion	Region of the source Bucket. Default: same with the region of the current clientconfig, which represents an intra-region copy	String	
sourceBucketName	Source bucket. The bucket entered must be in a format of {name}-{appid}	String	
sourceKey	Source object key. Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name `bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg` , the object key is doc1/pic1.jpg. For more information, please see Object Key	String	.
sourceVersionId	Version ID of the source file with multiple versions. Default: The latest version of the source file	String	
destinationBucketName	Destination bucket. The bucket entered must be in a format of {name}-{appid}	String	
destinationKey	Destination object key. Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name `bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg` , the object key is doc1/pic1.jpg. For more information, please see Object Key	String	.
storageClass	The storage type of the copied destination file (standard, low frequency, nearline). Default: Standard	String	

- **Returned value**
  - Successful: Return Copy. You can query whether the copy is completed, or wait until the copy finishes synchronously.
  - Failed: CosClientException or CosServiceException is thrown when an error (such as authentication failure) occurs. For more information, please see Exception Types.
- **Example**

```
// The region of bucket to copy. Cross-region copy is supported.
Region srcBucketRegion = new Region("ap-shanghai");
// Source bucket. The bucket entered must be in a format of {name}-{appid}
String srcBucketName = "srcBucket-1251668577";
// The source file to copy
String srcKey = "aaa/bbb.txt";
// Destination bucket. The bucket entered must be in a format of {name}-{appid}
String destBucketName = "destBucket-1251668577";
// The destination file to copy
String destKey = "ccc/ddd.txt";

// Generate srcCOSClient to get source file information
COSClient srcCOSClient = new COSClient(cred, new ClientConfig(srcBucketRegion));
CopyObjectRequest copyObjectRequest = new CopyObjectRequest(srcBucketRegion, srcBucketName,
srcKey, destBucketName, destKey);
try {
Copy copy = transferManager.copy(copyObjectRequest, srcCOSClient, null);
// Return an asynchronous result copy. You can synchronously call waitForCopyResult to wait for copy to end. If s
uccessful, CopyResult is returned, and an exception will be thrown if failed.
CopyResult copyResult = copy.waitForCopyResult();
} catch (CosServiceException e) {
e.printStackTrace();
} catch (CosClientException e) {
e.printStackTrace();
} catch (InterruptedException e) {
e.printStackTrace();
}
```

Permission Configuration

Permissions to access files on COS can be configured and obtained through SET/GET Object ACL and SET/GET Bucket ACL. The mainly used two types are AccessControlList and CannedAccessList. The following is the detailed description: The object inherits the bucket permission by default. COS only supports setting 1,000 ACL rules for an account (appid), so it is recommended to set ACL only for objects with inconsistent bucket permission to prevent the number of permission from exceeding the threshold. The function of unlimited ACL is under development.

AccessControlList

Custom access control policy is used to set policy control to a user.

Members of AccessControlList type

Member Name	Description	Type
List<Grant>	Contains information of all users to be authorized	Array
owner	The owner of the Object or Owner	Owner class

Members of Grant type

Member Name	Description	Type
grantee	Identity information of an authorized user	Grantee
permission	Authorized permission information (such as readable, writable, readable & writable)	Permission

Members of Owner type

Member Name	Description	Type

Permission | Identity information of an owner | String |
| displayname | Owner's name (same as id) | String |

- **Example**

```
// The identity information in the permission must be in the required format. The format for the root account and subaccount is as follows:
// Both root_uin and sub_uin below must be valid QQ numbers
// The root account qcs::cam::uin/<root_uin>:uin/<root_uin> indicates that the root_uin is granted to the root account (that is, the first and second uin are the same)
// For example, qcs::cam::uin/2779643970:uin/2779643970
// The sub-account qcs::cam::uin/<root_uin>:uin/<sub_uin> indicates that the sub_uin is granted to the root_uin.
// For example, qcs::cam::uin/2779643970:uin/73001122

AccessControlList acl = new AccessControlList();
Owner owner = new Owner();
// Set the owner information. Owner can only be a root account.
owner.setId("qcs::cam::uin/2779643970:uin/2779643970");
acl.setOwner(owner);

// Grant the root account 73410000 the read and write permissions
UinGrantee uinGrantee1 = new UinGrantee("qcs::cam::uin/73410000:uin/73410000");
```

```
acl.grantPermission(uinGrantee1, Permission.FullControl);
// Grant the sub-account 72300000 of 2779643970 the read permission
UinGrantee uinGrantee2 = new UinGrantee("qcs::cam::uin/2779643970:uin/72300000");
acl.grantPermission(uinGrantee2, Permission.Read);
// Grant the sub-account 7234444 of 2779643970 the write permission
UinGrantee uinGrantee3 = new UinGrantee("qcs::cam::uin/7234444:uin/7234444");
acl.grantPermission(uinGrantee3, Permission.Write);

// Set object's ACL
cosclient.setObjectAcl(bucket, key, acl);
```

CannedAccessControlList

CannedAccessControlList represents a preset policy for everyone. It is an enumeration type with the following enumerated values.

Enumerated Value	Description
Private	Private read and write (only owner can read and write)
PublicRead	Public read and private write (owner can read and write, and other users can read)
PublicReadWrite	Public read and write (everyone can read and write)

Client Encryption

The Java SDK supports client encryption feature that encrypts files before uploading, and decrypts them for downloading. Client encryption supports symmetric AES and asymmetric RSA encryption.
The symmetry and asymmetry here are only used to encrypt the generated random keys. AES256 is always used to encrypt file data symmetrically.
Client encryption is suitable for users who store sensitive data. client encryption may sacrifice certain upload speed, and the SDK may use serial method for multipart upload.

Preparation for client encryption

Aes256 is used in client encryption to encrypt data. By default, the earlier versions, such as JDK6 - JDK8, do not support 256-bit encryption. If an exception `java.security.InvalidKeyException: Illegal key size or default parameters` is reported during runtime, we must supplement Oracle's JCE unlimited permission file and deploy it in the JRE environment. Download the corresponding files based on the current JDK version and decompress and save them in the jre/lib/security directory under JAVA_HOME.

1. JDK6 JCE supplement package
2. JDK7 JCE supplement package
3. JDK8 JCE supplement package

Upload encryption process

1. Before uploading a file object, we randomly generate a symmetric encryption key. The random key is encrypted by the symmetric or asymmetric key provided by the user, and the encrypted result is encoded with base64 and stored in the meta information of the object.
2. When the file object is uploaded, AES256 algorithm is used to encrypt the file object in memory.

Download decryption process

1. Obtain the necessary encryption information in the meta information of the file, and decrypt the information decoded with base64 using the user key to obtain the key of the encrypted data

2. Use the key to decrypt the downloaded input stream using AES256 to obtain the decrypted file input stream.

3. **Example**

Use symmetric AES256 encryption to generate a random key example. For more information on the complete sample code, please see Complete Example of Symmetric Client Key Encryption

```
// Initialize user authentication information (secretId, secretKey)
COSCredentials cred = new BasicCOSCredentials("AKIDXXXXXXXXXXXXXXX",
"YYZZZZZZZZZZZZZZZZZ");
// Set the region where the bucket resides. For the abbreviations of COS regions, please see https://www..com/do
cument/product/436/6224
ClientConfig clientConfig = new ClientConfig(new Region("ap-beijing-1"));

// Load the key saved in the file. If it does not exist, use buildAndSaveSymmetricKey to generate the key first.
// buildAndSaveSymmetricKey();
SecretKey symKey = loadSymmetricAESKey();

EncryptionMaterials encryptionMaterials = new EncryptionMaterials(symKey);
// Use the AES/GCM mode and store the encrypted information in the meta information of the file.
CryptoConfiguration cryptoConf = new CryptoConfiguration(CryptoMode.AuthenticatedEncryption)
.withStorageMode(CryptoStorageMode.ObjectMetadata);

// Generate encryption client (EncryptionClient). The COSEncryptionClient is a subclass of COSClient, and all APIs
supported by COSClient are also available.
// EncryptionClient overwrites the COSClient upload and download logic. It will perform the encryption operation
internally. The other operations execution logic is consistent with that of the COSClient.
COSEncryptionClient cosEncryptionClient =
new COSEncryptionClient(new COSStaticCredentialsProvider(cred),
new StaticEncryptionMaterialsProvider(encryptionMaterials), clientConfig,
cryptoConf);

// Upload files
// Here is an example of putObject. For advanced API upload, use the COSEncryptionClient object when generati
ng TransferManager.
String bucketName = "mybucket-1251668577";
String key = "xxx/yyy/zzz.txt";
File localFile = new File("src/test/resources/plain.txt");
```

```
PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName, key, localFile);
cosEncryptionClient.putObject(putObjectRequest);
```

Use symmetric RSA encryption to generate a random key example. For more information on the complete sample code, please see Complete Example of Symmetric Client Key Encryption

```
// Initialize user authentication information (secretId, secretKey)
COSCredentials cred = new BasicCOSCredentials("AKIDXXXXXXXXXXXXXXXXXXX",
"YYZZZZZZZZZZZZZZZZZ");
// Set the region where the bucket resides. For the abbreviations of COS regions, please see https://cloud.tencent.
com/document/product/436/6224
ClientConfig clientConfig = new ClientConfig(new Region("ap-beijing-1"));


// Load the key saved in the file. If it does not exist, use buildAndSaveAsymKeyPair to generate the key first.
buildAndSaveAsymKeyPair();
KeyPair asymKeyPair = loadAsymKeyPair();

EncryptionMaterials encryptionMaterials = new EncryptionMaterials(asymKeyPair);
// Use the AES/GCM mode and store the encrypted information in the meta information of the file.
CryptoConfiguration cryptoConf = new CryptoConfiguration(CryptoMode.AuthenticatedEncryption)
.withStorageMode(CryptoStorageMode.ObjectMetadata);

// Generate encryption client (EncryptionClient). The COSEncryptionClient is a subclass of COSClient, and all APIs
supported by COSClient are also available.
// EncryptionClient overwrites the COSClient upload and download logic. It will perform the encryption operation
internally. The other operations execution logic is consistent with that of the COSClient.
COSEncryptionClient cosEncryptionClient =
new COSEncryptionClient(new COSStaticCredentialsProvider(cred),
new StaticEncryptionMaterialsProvider(encryptionMaterials), clientConfig,
cryptoConf);

// Upload files
// Here is an example of putObject. For advanced API upload, use the COSEncryptionClient object when generati
ng TransferManager.
String bucketName = "mybucket-1251668577";
String key = "xxx/yyy/zzz.txt";
File localFile = new File("src/test/resources/plain.txt");
PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName, key, localFile);
cosEncryptionClient.putObject(putObjectRequest);
```

Exception Description

When the SDK fails, the exceptions thrown are all RuntimeExcpetion. The common SDK exceptions are CosClientException, CosServiceException, and IllegalArgumentException.

CosClientException

Client exceptions refer to server interaction failures caused by unexpected client issues such as failure to connect to the server, failure to parse the data returned by the server, and the occurrence of I/O exception when reading a local file. Inherited from RuntimeException, CosClientException has no custom member variables, and is used in the same way as RuntimeException.

CosServiceException

CosServiceException service exception refers to scenarios in which interaction is completed but the operation failed. For example, the client accesses a bucket that does not exist, delete a file that does not exist, or does not have the permission to perform an operation, or the server failed. CosServiceException contains the status code returned by the server, requestid, error details, and so on. After an exception is captured, it is recommended to print the entire exception. The exception contains the necessary troubleshooting factors. Member variables of exception are described as follows:

request Member	Description	Type
requestId	Request ID to specify a request. It is very important for troubleshooting.	String
traceId	ID for troubleshooting	String
statusCode	Status code of the response. 4xx represents the request failure caused by the client, and 5xx represents the failure caused by the server exception For more information, please see [COS Error Message] (https://cloud.tencent.com/document/product/436/7730)	String
errorType	Enumeration type, indicating the type of exception (Client, Service, and Unknown)	ErrorType
errorCode	Error Code returned by body when request fails. For more information, please see COS Error Message	String
errorMessage	Error Message returned by body when request fails. For more information, please see COS Error Message	String

FAQ

1. Why does java.lang.NoSuchMethodError appear when I run the SDK?
   A JAR packet conflict may occur. For example, the JAR package for http in the user's project does not have method A, but the SDK-dependent JAR package has method A. The loading order goes wrong. The http library in the user project is loaded. When the SDK is running, the NoSuchMethodError exception is thrown. Solution: Change the version of the package that causes NoSuchMethodError in the contained project to the version of the corresponding library in the pom.xml in the SDK.
2. Upload using SDK is very slow,Logs frequently display IOException ?
   Causes and solutions:
   a. Check whether you were accessing COS over the public network. For COS access in the same region, it is recommended to use the private network. (IP address range 10,100,169 is resolved from the private network domain name.For more information on COS domains, please see Available Regions for COS). If the public network is used,

check whether the outbound bandwidth is small or whether other programs occupy bandwidth resources.
b. Ensure that the log level in the production environment is not debug. INFO log is recommended. For information on log configuration of log4j, please see log4j log configuration template.
c. The speed of simple upload can reach 10 MB. When you use an advanced API and 32 concurrency level, the speed can reach 60 MB. If your speed is far less than these two value. Refer to a and b.
d. If warn log displays IOException, ignore it. The SDK will retry. If it fails after multiple attempts of retries, the log displays IOException, which may be caused by too slow speed. For the cause, please see a and b.

3. How do I create a directory using the SDK?
Files and directories in COS are objects, and directories are objects ending with "/". When you create a file, you do not need to create a directory. For example, if you create a file with an object key of xxx/yyy/zzz.txt, just set the key to xxx/yyy/zzz.txt instead of creating an xxx/yyy/ object. Separate directories with "/" to display the hierarchy on the console. However, these directory objects may not exist. If you want to create a directory object, use the following sample code.

```
String bucketName = "mybucket-125166000";=
String key = "xxx/yyy/";
//A directory object is an empty file ending with "/". Upload a byte stream with a length of 0.
InputStream input = new ByteArrayInputStream(new byte[0]);
ObjectMetadata objectMetadata = new ObjectMetadata();
objectMetadata.setContentLength(0);

PutObjectRequest putObjectRequest =
new PutObjectRequest(bucketName, key, input, objectMetadata);
PutObjectResult putObjectResult = cosclient.putObject(putObjectRequest);
```

JavaScript SDK Getting Started

Last updated：2018-08-24 10:07:18

Preparations for Development

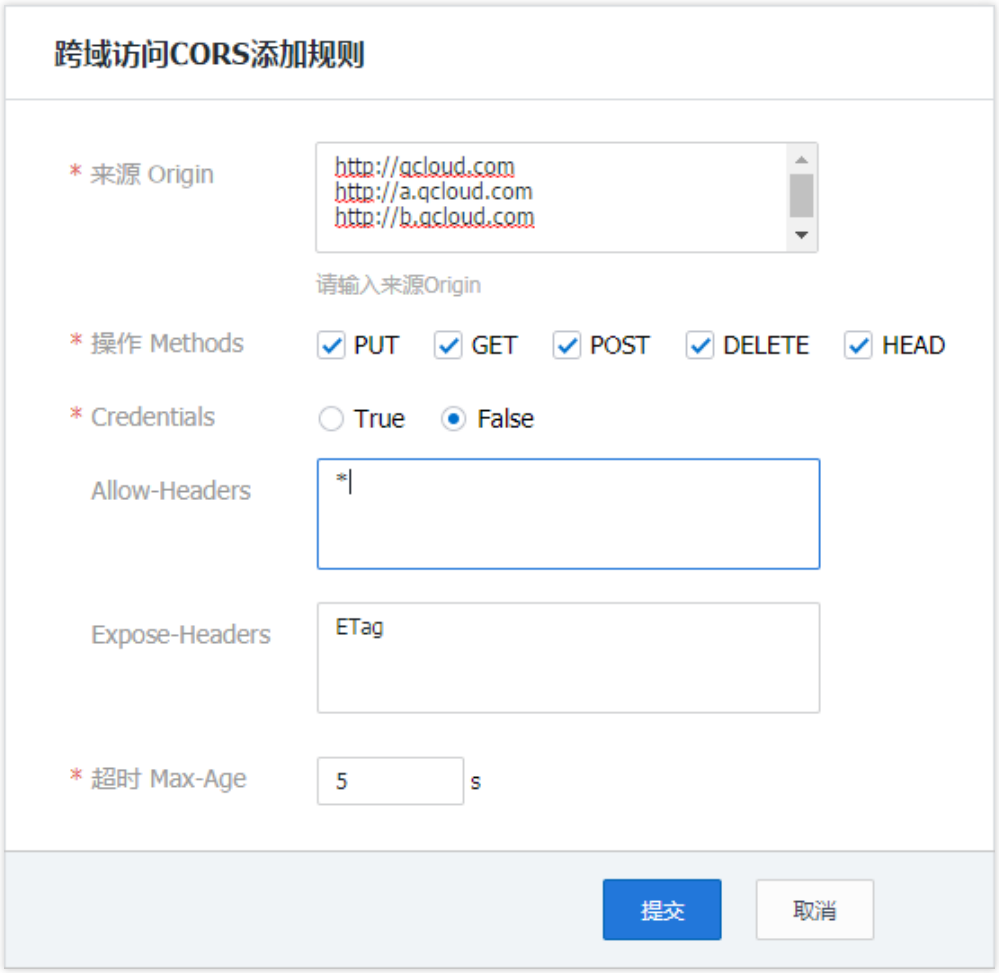
Obtain SDK

Download XML JS SDK resources for COS service on Github from: tencentyun/cos-js-sdk-v5.

Download Demo from XML JS SDK Demo.

Preparations for development

1. First, JS SDK requires the browser to support basic HTML5 features in order to support uploading files using ajax and computing md5 values of files.
2. Go to the COS Console to create a bucket and obtain Bucket (bucket name) and Region (region name).
3. Go to Key Management on the console to obtain SecretId and SecretKey of your project.
4. Configure CORS rules, as shown below:



For more information on the definitions of SecretId, SecretKey, Bucket and other terms and how to obtain them, please see COS Glossary.

Getting Started

Computing signature

If the signature computing is implemented at frontend, SecretId and SecretKey can be exposed. For this reason, the signature computing is performed at backend. The frontend obtains the signature computing result via ajax. Add a permission verification at backend for your website during the deployment.For other languages, please see the relevant XML SDK.

Example for upload

1. Create test.html, fill in the following code, and modify the Bucket and Region in it.
2. Deploy the backend signature service and modify the signature service address in getAuthorization.
3. Place test.html on the Web server, then visit the page in the browser to test the file upload.

```
<input id="file-selector" type="file">
<script src="dist/cos-js-sdk-v5.min.js"></script>
<script>
var Bucket = 'test-1250000000';
var Region = 'ap-guangzhou';

// Initialize the instance
var cos = new COS({
getAuthorization: function (options, callback) {
// Obtain the signature asynchronously
$.get('../server/sts.php', {
bucket: options.Bucket,
region: options.Region,
}, function (data) {
callback({
TmpSecretId: data.TmpSecretId,
TmpSecretKey: data.TmpSecretKey,
XCosSecurityToken: data.XCosSecurityToken,
ExpiredTime: data.ExpiredTime,
});
});
}
});

// Listen on the selected file
document.getElementById('file-selector').onchange = function () {

var file = this.files[0];
```

```
if (!file) return;

// Upload the file in multipart
cos.sliceUploadFile({
Bucket: Bucket,
Region: Region,
Key: file.name,
Body: file,
}, function (err, data) {
console.log(err, data);
});

};
</script>
```

Introducing Webpack

In the scenarios where webpack is supported, it can be introduced as a module using npm.

```
npm i cos-js-sdk-v5 --save
```

Other Documents and Examples

1. For more examples, please see XML JavaScript SDK Demo.
2. For the complete API documentation, please see XML JavaScript SDK API documentation.

API Documentation

Last updated：2018-07-20 18:35:03

This article provides a detailed description of the APIs for the JavaScript SDK.

URL to JavaScript SDK github: tencentyun/cos-js-sdk-v5

In codes described in the following sections, "COS" indicates the class of SDK, and "cos" the SDK instance.

For more information on the definitions of SecretId, SecretKey, Bucket, Region and other terms and how to obtain them, please see COS Glossary.

The " - " preceding the parameter name indicates a "sub-parameter".

Constructor

new COS({})

When being referenced directly with a script tag, the SDK occupies the global variable name "COS". By using the constructor of this "COS", you can create an SDK instance.

Use Case

Create a COS SDK instance:

- Format 1 (recommended): Use the temporary key format:

```
var cos = new COS({
// Required
getAuthorization: function (options, callback) {
$.get('http://example.com/server/sts.php', {
bucket: options.Bucket,
region: options.Region,
}, function (data) {
callback({
TmpSecretId: data.TmpSecretId,
TmpSecretKey: data.TmpSecretKey,
XCosSecurityToken: data.XCosSecurityToken,
ExpiredTime: data.ExpiredTime,
});
});
}
});
```

- Format 2: Request a signature from the backend if it is required for every request from the frontend.

```
var cos = new COS({
// Required
getAuthorization: function (options, callback) {
$.get('http://example.com/server/auth.php', {
method: options.Method,
pathname: '/' + options.Key,
}, function (Authorization) {
callback({
Authorization: Authorization
// XCosSecurityToken: data.XCosSecurityToken, // If the signature is calculated with the temporary key, XCosSecurityToken is required.
});
});
},
// Optional
FileParallelLimit: 3, // Limits the number of concurrent file uploads
ChunkParallelLimit: 3, // Limits the number of concurrent multipart uploads under a single file
ProgressInterval: 1000, // Limits the intervals between onProgress callbacks for upload
});
```

- Format 3: A fixed key is used and the signature calculated at the frontend (To protect the key from leakage, we recommend that you use it for debugging only).

```
var cos = new COS({
SecretId: 'AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
SecretKey: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
});
```

Parameters of the constructor

Parameter Name	Description	Type	Required
SecretId	Indicates the user's SecretId	String	No
SecretKey	Indicates the user's SecretKey. To protect the key from leakage, we recommend that you use it for debugging at the frontend only.	String	No
FileParallelLimit	Indicates the number of concurrent file uploads under one instance, which defaults to 3.	Number	No
ChunkParallelLimit	Indicates the number of concurrent multipart uploads under one file, which defaults to 3.	Number	No
ChunkSize	Indicates the size of each part for multipart upload, which defaults to 1048576 (1MB)	Number	No

Parameter Name	Description	Type	Required
ProgressInterval	Indicates the frequency at which onProgress, the callback method for upload progress, is called back, which is in ms and defaults to 1000	Number	No
Protocol	Indicates a custom request protocol. Options include `https:` and `http:`. `http:` is used if the page is identified to be an `http:` one. Otherwise, `https:` is used.	String	No
getAthorization	Indicates the callback method for acquiring signature. This parameter is required if no SecretId or SecretKey is available.	Function	No

Details on the getAuthorization callback function (use Format 1)

```
getAuthorization: function(options, callback) { ... }
```

Callback parameters for getAuthorization:

Parameter Name	Description	Type
options	Parameter objects required to acquire the temporary key	Function
- Bucket	Bucket name. The bucket name entered must be in a format of {name}-{appid}.	String
- Region	The region where the Bucket resides. For enumerated values, please see [Bucket Region].(https://cloud.tencent.com/document/product/436/6224)	String
callback	Indicates the postback method after the temporary key is acquired	Function

Callback posts back an object after the temporary key is acquired. Here is a list of the attributes of this object:

Attribute Name	Parameter Description	Type	Required
TmpSecretId	Indicates the tmpSecretId for the acquired temporary key	String	Yes
TmpSecretKey	Indicates the tmpSecretKey for the acquired temporary key	String	No
XCosSecurityToken	Indicates the sessionToken for the acquired temporary key, which corresponds to the x-cos-security-token field of the header	String	No
ExpiredTime	Indicates the expiredTime for the acquired temporary key, which means the timeout length	String	No

Details on the getAuthorization callback function (use Format 2)

```
getAuthorization: function(options, callback) { ... }
```

Callback parameters for getAuthorization:

Parameter Name	Description	Type	Required
options	Parameter objects required to acquire the signature	Function	
- Method	Indicates the method for the current request	Function	No
- Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	No
- Query	The query parameter object for the current request, in a format of {key: 'val'}	Object	No
- Headers	The header parameter object for the current request, in a format of {key: 'val'}	Function	No
callback	Indicates the callback after the temporary key is acquired	Function	No

After the calculation is completed with getAuthorization, callback posts back a signature string or an object:
Posting back the signature string posts back a string value, requesting Authorization, the authentication header credential field to be used.
Here is a list of the attributes of the posted back object:

Attribute Name	Parameter Description	Type	Required
Authorization	Indicates the Authorization information for the acquired temporary key	String	Yes
XCosSecurityToken	Indicates the sessionToken for the acquired temporary key, which corresponds to the x-cos-security-token field of the header	String	No

Acquire authentication credentials

You have three ways to acquire the authentication credentials for your instance by passing in different parameters during instantiation:

1. Pass in SecretId and SecretKey. This allows the signature, every time it is required, to be calculated within the instance.
2. Pass in the getAuthorization callback. This allows the signature, every time it is required, to be calculated and returned to the instance by this callback.
3. Pass in the getSTS callback. This allows the temporary key, every time it is required, to be returned to the instance by this callback and the signature, every time it is required, to be calculated with the temporary key within the instance.

Static Methods

COS.getAuthorization

For COS XML API requests, the authentication credential Authorization is required for all operations on private resources to identify whether the current request is valid.

There are two ways to use authentication credentials:

1. Use them in header parameters, with the field name of "authorization".
2. Use them in URL parameters, with the field name of "sign".

The COS.getAuthorization method is used to calculate authentication credentials, i.e., the signature information used to verify the validity of requests.

> **Note**:
> We recommend that you use this method for debugging at the frontend only. Calculating the signature at the frontend is not recommended for launching projects, because this may cause the key to be leaked.

Use Case

Acquire the authentication credentials for uploading files:

```
var Authorization = COS.getAuthorization({
SecretId: 'AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
SecretKey: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
Method: 'get',
Key: 'a.jpg',
Expires: 60,
Query: {},
Headers: {}
});
```

Parameters

Parameter Name	Description	Type	Required
SecretId	Indicates the user's SecretId	String	Yes
SecretKey	Indicates the user's SecretKey	String	Yes
Method	Indicates the operation method, such as get, post, delete, head, and other HTTP methods	String	Yes
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. **If the requested operation is on a file, this parameter indicates the file name and is required**. If the operation is on a bucket, it is left empty.	String	No
Expires	Indicates the length signature timeout in seconds, which defaults to 900 seconds	Number	No

Parameter Name	Description	Type	Required
Query	Indicates the requested query parameter object	Object	No
Headers	Indicates the requested header parameter object	Object	No

Returned Result

The calculated authentication credential string "authorization" is returned.

Tool-based Methods

Get Auth

The cos.getAuth method is equivalent to the edition of COS.getAuthorization mounted to the instance. The difference is that cos.getAuth does not require passing in SecretId and SecretKey, but uses the method to acquire authentication credentials for the object itself instead.

Use Case

```
var authorization = cos.getAuth({
Method: 'get',
Key: '1.jpg'
});
```

Parameters

Parameter Name	Description	Type	Required
Method	Indicates the operation method, such as get, post, delete, head, and other HTTP methods	String	Yes
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. **If the requested operation is on a file, this parameter indicates the file name and is required**. If the operation is on a bucket, it is left empty.	String	No
Expires	Indicates the length signature timeout in seconds, which defaults to 900 seconds	Number	No
Query	Indicates the requested query parameter object	Object	No
Headers	Indicates the requested header parameter object	Object	No

Returned Result

The calculated authentication credential string "authorization" is returned.

Get Object Url

Use Case

// Acquire an unsigned object URL

```
var url = cos.getObjectUrl({
Key: '1.jpg',
Sign: false
});
```

// Acquire a signed object URL

```
cos.getObjectUrl({
Key: '1.jpg',
Sign: true
}, function (err, data) {
console.log(err || data.Url);
});
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. **If the requested operation is on a file, this parameter indicates the file name and is required**. If the operation is on a bucket, it is left empty.	String	Yes
Sign	Whether to return a signed URL	Boolean	No
Method	Indicates the operation method, such as get, post, delete, head, and other HTTP methods, which defaults to "get".	String	No
Query	Indicates the query parameter object involved in signature calculation	Object	No
Headers	Indicates the header parameter object involved in signature calculation	Object	No

Returned Result

A string is returned in either of the two ways:

1. If signature calculation can be performed synchronously, (for example, SecretId and SecretKey are passed in during instantiation), a signed URL is returned by default.
2. Otherwise, an unsigned URL is returned.

Callback Function Description

```
function(err, data) { ... }
```

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
- Url	Indicates the calculated URL	String

Bucket Operations

Head Bucket

Feature description

The Head Bucket request is used to determine whether the Bucket and the permission to access the Bucket exist. The same permission applies to Head and Read. HTTP status code 200 will be returned if the Bucket exists, 403 if there is no permission, and 404 if the Bucket does not exist.

Use Case

```
cos.headBucket({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
}, function(err, data) {
console.log(err || data);
});
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes

Callback Function Description

```
function(err, data) { ... }
```

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object

Get Bucket

Feature description

Get Bucket request is identical to List Object request. It is used to list partial or all of the Objects under the Bucket. The caller of this API requires Read permission for the Bucket.

Use Case

List all the files under the "a" directory

```
cos.getBucket({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
Prefix: 'a/', /* Not required */
}, function(err, data) {
console.log(err || data);
});
```

List files under the "a" directory, without deep traversal

```
cos.getBucket({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou' /* Required */
Prefix: 'a/', /* Not required*/
Delimiter: '/', /* Not required*/
}, function(err, data) {
console.log(err || data);
});
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Prefix	Indicates the prefix match, which is used to specify the prefix address of the returned file	String	No
Delimiter	Delimiter is a sign. If Prefix exists, the same paths between Prefix and delimiter are grouped as the same type and defined as Common Prefix, and then all Common Prefixes are listed. If Prefix does not exist, the listing process starts from the beginning of the path.	String	No
Marker	Entries are listed using UTF-8 binary order by default, starting from the marker	String	No
MaxKeys	Maximum number of entries returned at a time. Default is 1,000	String	No
EncodingType	Indicates the encoding method of the returned value. Available value: url	String	No

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
- headers	Indicates the header information returned for a request	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- CommonPrefixes	The same paths between Prefix and delimiter are grouped as the same type and defined as Common Prefix	Array
- - Prefix	Indicates a single Common prefix	String
- - Name	Provides the information of Bucket	String

Parameter Name	Description	Type
- Prefix	Indicates the prefix match, which is used to specify the prefix address of the returned file	String
- Marker	Entries are listed using UTF-8 binary order by default, starting from the marker	String
- MaxKeys	Maximum number of entries of the result returned for response request each time	String
- IsTruncated	Indicates whether the returned entry is truncated. String value: 'true' or 'false'	String
- NextMarker	If the returned entry is truncated, NextMarker represents the starting point of the next entry	String
- Encoding-Type	Encoding type of Delimiter, which is used for Marker, Prefix, NextMarker, and Key	String
- Contents	Metadata information	Array
- - ETag	The MD-5 algorithm check value of the file, such as `"22ca88419e2ed4721c23807c678adbe4c08a7880"` . **Be sure to enclose the value in double quotation marks**	String
- - Size	Indicates the file size (in bytes)	String
- - Key	Object name	String
- - LastModified	Indicates the time when Object was last modified, such as 2017-06-23T12:33:27.000Z	String
- - Owner	Information of the Bucket owner	Object
- ID	AppID of the Bucket	String
- StorageClass	The storage level of Object. Enumerated values: STANDARD, STANDARD_IA	String

Delete Bucket

Feature description

Delete Bucket API request is used to delete a Bucket under a specified account. The Bucket must be empty before it can be deleted. The Bucket can be deleted only if its content is removed. The HTTP status code 200 or 204 is returned upon a successful deletion.

Use Case

Call Delete Bucket:

```
cos.deleteBucket({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou' /* Required */
}, function(err, data) {
```

```
console.log(err || data);
});
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object

Get Bucket ACL

Feature description

This API (Get Bucket ACL) is used to obtain the ACL (access control list) of the Bucket. Only the Bucket owner has the access to this API.

Use Case

```
cos.getBucketAcl({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou' /* Required */
```

```
}, function(err, data) {
console.log(err || data);
});
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object
- Owner	Information of the Bucket owner	Object
- - DisplayName	Name of the Bucket owner	String
- - ID	ID of the Bucket owner. Format: qcs::cam::uin/<OwnerUin>:uin/<SubUin>; in case of root account, <OwnerUin> and <SubUin> are of the same value	String
- Grants	Lists information of authorized users and their permissions	Array
- - Permission	Indicates the permissions granted to authorized users. Enumerated values: READ, WRITE, and FULL_CONTROL	String

Parameter Name	Description	Type
- - Grantee	Indicates information of authorized users. "type" can be RootAccount and Subaccount. In case of RootAccount, ID is specified as root account. In case of Subaccount, ID is specified as sub-account	Object
- - - DisplayName	Indicates the username	String
- - - ID	Indicates the user ID. In case of root account, format: qcs::cam::uin/<OwnerUin>:uin/<OwnerUin> or qcs::cam::anyone:anyone (all users). In case of sub-account, format: qcs::cam::uin/<OwnerUin>:uin/<SubUin>	String

Put Bucket ACL

Feature description

The API Put Bucket ACL is used to write ACL for a Bucket. You can import ACL information either by using Header: "x-cos-acl", "x-cos-grant-read", "x-cos-grant-write", "x-cos-grant-full-control", or by using body in XML format.

Use Case

Set Public Read for Buckets

```
cos.putBucketAcl({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
ACL: 'public-read'
}, function(err, data) {
console.log(err || data);
});
```

Grants a user the permission to read and write buckets.

```
cos.putBucketAcl({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
GrantFullControl: 'id="qcs::cam::uin/1001:uin/1001",id="qcs::cam::uin/1002:uin/1002"' // 1001 represents a uin
}, function(err, data) {
console.log(err || data);
});
```

Grants a user the permission to read and write buckets.

```
cos.putBucketAcl({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
GrantFullControl: 'id="qcs::cam::uin/1001:uin/1001",id="qcs::cam::uin/1002:uin/1002"' // 1001 represents a uin
}, function(err, data) {
console.log(err || data);
});
```

Changes Bucket permissions using AccessControlPolicy

```
cos.putBucketAcl({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
AccessControlPolicy: {
"Owner": { // AccessControlPolicy must contain "owner".
"ID": 'qcs::cam::uin/459000000:uin/459000000' // 459000000 represents the QQ number of the Bucket user.
},
"Grants": [{
"Grantee": {
"ID": "qcs::cam::uin/10002:uin/10002", // 10002 repsents a QQ number
},
"Permission": "WRITE"
}]
}
}, function(err, data) {
console.log(err || data);
});
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
ACL	Defines the ACL attribute of Object. Valid values: private, public-read, public-read-write. Default value is private.	String	No
GrantRead	Grants read permission to the authorized user. Format: id=" ",id=" ". For authorization to a subaccount, id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>"; for authorization to the root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>". For example: 'id="qcs::cam::uin/123:uin/123", id="qcs::cam::uin/123:uin/456"'	String	No

Parameter Name	Description	Type	Required
GrantWrite	Grants write permission to the authorized user. Format: id=" ",id=" ". For authorization to a subaccount, id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>"; for authorization to the root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>". For example: 'id="qcs::cam::uin/123:uin/123", id="qcs::cam::uin/123:uin/456"'	String	No
GrantFullControl	Grants read and write permissions to the authorized user. Format: id=" ",id=" ". For authorization to a subaccount, id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>"; for authorization to the root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>". For example: 'id="qcs::cam::uin/123:uin/123", id="qcs::cam::uin/123:uin/456"'	String	No
AccessControlPolicy	Provides all configuration information of cross-origin resource sharing	Object	No
- Owner	Indicates the object representing the Bucket owner	Object	No
- - ID	Indicates the string representing the user ID, for example, qcs::cam::uin/1001:uin/1001, where 1001 represents a uin	Object	No
- Grants	Provides all configuration information of cross-origin resource sharing	Object	No
- - Permission	Provides all configuration information of cross-origin resource sharing. Options include READ, WRITE, FULL_CONTROL, READ_ACP, and WRITE_ACP.	String	No
- - Grantee	Provides all configuration information of cross-origin resource sharing	Array	No
- - - ID	Indicates the string representing the user ID, for example, qcs::cam::uin/1001:uin/1001, where 1001 represents a uin	String	No
- - - DisplayName	Indicates the string representing the username, which is usually entered as a string matching the user ID.	String	No

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object

Get Bucket CORS

Feature description

This API (Get Bucket CORS) is used by the Bucket owner to configure cross-origin resource sharing on a bucket. (Cross-origin Resource Sharing (CORS) is a W3C standard.) By default, the Bucket owner has the permission of this API and can grant it to others.

Use Case

Call Get Bucket CORS:

```
cos.getBucketCors({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
}, function(err, data) {
console.log(err || data);
});
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
- CORSRules	Provides all configuration information of cross-origin resource sharing	Array
- - AllowedMethods	Allowed HTTP operations. Enumerated values: GET, PUT, HEAD, POST, DELETE.	Array
- - AllowedOrigins	Allowed access source. The wildcard "*" is supported. Format: protocol://domain name[:port], for example, http://www.qq.com	Array
- - AllowedHeaders	When an OPTIONS request is sent, notifies the server about which custom HTTP request headers are allowed for subsequent requests. Wildcard "*" is supported.	Array
- - ExposeHeaders	Sets the custom header information that can be received by the browser from the server end.	Array
- - MaxAgeSeconds	Sets the validity period of the results obtained by OPTIONS	String
- - ID	Sets rule ID	String

Put Bucket CORS

Note:

1. To change the cross-origin access configuration , make sure the Bucket provides cross-origin support. Go to the console to make cross-origin access configuration . For more information, please see Development Environment.
2. Make sure that changing cross-origin access configuration does not affect the cross-origin requests under the current origin.

Feature description

The API Put Bucket CORS is used to set cross-origin resource sharing permission for your Bucket. You can do so by importing configuration files of XML format (file size limit: 64 KB). By default, the Bucket owner has the permission of this API and can grant it to others.

Use Case

Call Put Bucket CORS:

```
cos.putBucketCors({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
CORSRules: [{
"AllowedOrigin": ["*"],
"AllowedMethod": ["GET", "POST", "PUT", "DELETE", "HEAD"],
"AllowedHeader": ["*"],
"ExposeHeader": ["ETag", "x-cos-acl", "x-cos-version-id", "x-cos-delete-marker", "x-cos-server-side-encryption"],
"MaxAgeSeconds": "5"
}]
}, function(err, data) {
console.log(err || data);
});
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
CORSRules	Provides all configuration information of cross-origin resource sharing	Array	No
- ID	Sets rule ID (optional)	String	No
- AllowedMethods	Allowed HTTP operations. Enumerated values: GET, PUT, HEAD, POST, DELETE.	Array	Yes
- AllowedOrigins	Allowed access source. The wildcard "*" is supported. Format: protocol://domain name[:port], for example, `http://www.qq.com`	Array	Yes
- AllowedHeaders	When an OPTIONS request is sent, notifies the server about which custom HTTP request headers are allowed for subsequent requests. Wildcard "*" is supported.	Array	No
- ExposeHeaders	Configures the custom header information that can be received by the browser from the server end.	Array	No
- MaxAgeSeconds	Sets the validity period of the results obtained by OPTIONS	String	No

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object

Delete Bucket CORS

Note:

1. Deleting the `cross-origin access configuration` information of the current Bucket may cause all the cross-origin requests to fail. Please proceed with caution.
2. This method is not recommend for use at the browser end.

Feature description

Delete Bucket CORS API request is used to delete configuration information of cross-domain access.

Use Case

Call Delete Bucket CORS:

```
cos.deleteBucketCors({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
}, function(err, data) {
console.log(err || data);
});
```

Parameters

Parameter Name	Description	Type	Required

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object

Get Bucket Location

Feature description

This API (Get Bucket Location) is used to obtain the information of the region where the Bucket resides. This GET operation returns the region where the Bucket resides via the location sub-resource. Only the Bucket owner is allowed to operate on this API.

Use Case

Call Get Bucket Location:

```
cos.getBucketLocation({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
}, function(err, data) {
```

```
console.log(err || data);
});
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object
- LocationConstraint	The region where the Bucket resides. For enumerated values, please see [Bucket Region].(https://cloud.tencent.com/document/product/436/6224)	String

Object Operations

Head Object

Feature description

The Head Object request is used to obtain the metadata of the corresponding Object. The same permission applies to Head and Get.

Use Case

Call Head Object:

```
cos.headObject({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
Key: '1.jpg', /* Required */
}, function(err, data) {
console.log(err || data);
});
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	Yes
IfModifiedSince	If Object is modified after the specified time, the Object meta information is returned, otherwise 304 is returned	String	No

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200 and 304. If the change has not been modified since the specified time, 304 is returned.	Number

Parameter Name	Description	Type
- headers	Indicates the header information returned for a request	Object
- x-cos-object-type	Indicates whether the Object is appendable for upload. Enumerated values: normal or appendable	String
- x-cos-storage-class	The storage level of Object. Enumerated values: STANDARD, STANDARD_IA	String
- x-cos-meta- *	User-defined meta information	String
- NotModified	Indicates whether the Object is left unmodified since the specified time	Boolean

Get Object

Feature description

Get Object API request is used to download one file (Object) in Bucket of COS to the local computer. This action requires that the user has the read permission for the target Object or the read permission for the target Object has been made available for everyone (public-read).

Use Case

Call Get Object:

```
cos.getObject({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
Key: '1.jpg', /* Required */
}, function(err, data) {
console.log(err || data.Body);
});
```

Specify the range to obtain file content

```
cos.getObject({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
Key: '1.jpg', /* Required */
Range: 'bytes=1-3', /* Not required */
}, function(err, data) {
console.log(err || data.Body);
});
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	Yes
ResponseContentType	Sets the Content-Type parameter in the response header	String	No
ResponseContentLanguage	Sets the Content-Language parameter in the returned header	String	No
ResponseExpires	Sets the Content-Expires parameter in the returned header	String	No
ResponseCacheControl	Sets the Cache-Control in the returned header	String	No
ResponseContentDisposition	Sets the Content-Disposition parameter in the returned header	String	No
ResponseContentEncoding	Sets the Content-Encoding parameter in the returned header	String	No
Range	The specified range of file download defined in RFC 2616 (in bytes), such as Range: 'bytes=1-3'	String	No
IfModifiedSince	If Object is modified after the specified time, the Object meta information is returned, otherwise 304 is returned	String	No
IfUnmodifiedSince	The file content is returned if the file has been modified at or before the specified time. If not, 412 (precondition failed) is returned	String	No
IfMatch	The file is returned if Etag is the same as the specified value. If not, 412 (precondition failed) is returned	String	No
IfNoneMatch	The file is returned if Etag is different from the specified value. Otherwise 304 is returned (not modified)	String	No

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 304, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object
- x-cos-object-type	Indicates whether the Object is appendable for upload. Enumerated values: normal or appendable	String
- x-cos-storage-class	The storage level of Object. Enumerated values: STANDARD, STANDARD_IA. **Note: If this header is not returned, it means the file is at a STANDARD storage level**.	String
- x-cos-meta- *	User-defined metadata	String
- NotModified	If IfModifiedSince is used for request and the file is not modified, the value is true. Otherwise, it is false.	Boolean
- Body	The returned file conten, which is in string format by default	String

Put Object

Feature description

Put Object request allows you to upload a local file (Object) to the specified Bucket. This action requires that the user has the WRITE permission for the Bucket.

Notes:

1. Key (file name) should not end with `/`. Otherwise, it will be identified as a folder.
2. A maximum of 1,000 ACL policies are allowed under a single Bucket. Therefore, set the ACL permission for no more than 999 files under one Bucket.

Use Case

Call Put Object to upload files:

```
cos.putObject({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
Key: '1.jpg', /* Required */
StorageClass: 'STANDARD',
Body: file, // Upload file objects
onProgress: function(progressData) {
console.log(JSON.stringify(progressData));
}
}, function(err, data) {
console.log(err || data);
});
```

Upload strings as file content:

```
cos.putObject({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
Key: '1.jpg', /* Required */
Body: 'hello!',
}, function(err, data) {
console.log(err || data);
});
```

Upload strings as file content:

```
cos.putObject({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
Key: '1.jpg', /* Required */
Body: 'hello!',
}, function(err, data) {
console.log(err || data);
});
```

Create a directory

```
cos.putObject({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
Key: 'a/', /* Required */
Body: '',
}, function(err, data) {
console.log(err || data);
});
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	Yes
CacheControl	Cache-Control, the caching policy defined in RFC 2616 and saved as Object metadata	String	No
ContentDisposition	The file name defined in RFC 2616, which will be saved as Object metadata.	String	No
ContentEncoding	The encoding format defined in RFC 2616, which is saved as Object metadata	String	No
ContentLength	HTTP request content length defined in RFC 2616 (in bytes)	String	No
ContentType	The content type (MIME) defined in RFC 2616, which is saved as Object metadata	String	No
Expect	If Expect: 100-continue is used, the request content will not be sent until the receipt of response from server	String	No
Expires	The expiration time defined in RFC 2616, which is saved as Object metadata	String	No
ACL	Defines the ACL attribute of Object. Valid values: private, public-read, public-read-write. Default value is private.	String	No
GrantRead	Grants read permission to the authorized user. Format: id="",id="". For authorization to a subaccount, id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>"; for authorization to the root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>". For example: 'id="qcs::cam::uin/123:uin/123", id="qcs::cam::uin/123:uin/456"'	String	No
GrantWrite	Grants write permission to the authorized user. Format: id="",id="". For authorization to a subaccount, id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>"; for authorization to the root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>". For example: 'id="qcs::cam::uin/123:uin/123", id="qcs::cam::uin/123:uin/456"'	String	No

Parameter Name	Description	Type	Required
GrantFullControl	Grants read and write permissions to the authorized user. Format: id=" ",id=" ". For authorization to a subaccount, id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>"; for authorization to the root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>". For example: 'id="qcs::cam::uin/123:uin/123", id="qcs::cam::uin/123:uin/456"'	String	No
StorageClass	Sets the storage level of Object. Enumerated values: STANDARD, STANDARD_IA. Default value: STANDARD	String	No
x-cos-meta- *	The header information allowed to be defined by users , which will be returned as Object metadata. The size is limited to 2K.	String	No
Body	The content of the file uploaded, which can be `strings` , `File objects` or `Blob objects`	String \ File\ Blob	No
onProgress	Callback function for progress. Below is a list of attributes for the progress callback response object (progressData)	Function	No
progressData.loaded	Indicates the size of the downloaded portion of the file in bytes	Number	No
progressData.total	Indicates the size of the entire file in bytes	Number	No
progressData.speed	Indicates the download speed in bytes/s	Number	No
progressData.percent	The percentage of file downloaded in decimal, for example, 0.5 for 50%	Number	No

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object

Parameter Name	Description	Type
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object
- ETag	Returns the MD5 algorithm check value for the file. The ETag value can be used to check whether the Object is corrupted in the upload process. **Note: The ETag value must be enclosed in double quotation marks, such as "09cba091df696af91549de27b8e7d0f6"**	String

Delete Object

Feature description

Delete Object API request is used to delete one file (Object) in Bucket of COS. This action requires that the user has the WRITE permission for the Bucket.

Use Case

Call Delete Object:

```
cos.deleteObject({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
Key: '1.jpg' /* Required */
}, function(err, data) {
console.log(err || data);
});
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	Yes

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 204, 403 and 404. **If the deletion is successful or the file does not exist, 204 or 200 is returned. If the specified Bucket is not found, 404 is returned.**	Number
- headers	Indicates the header information returned for a request	Object

Options Object

Feature description

This API (Options Object) is used to implement a pre-request for cross-origin access configuration. This is how it works. Before any cross-origin request is sent, an OPTIONS request, along with the specific source origin, HTTP method and header information, to COS to determine whether a true cross-origin request can be sent. When the CORS configuration does not exist, 403 Forbidden is returned for the request.
Use the Put Bucket CORS API to enable CORS support for the Bucket.

Use Case

Call Options Object:

```
cos.optionsObject({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
Key: '1.jpg', /* Required */
Origin: 'https://www.qq.com', /* Required */
AccessControlRequestMethod: 'PUT', /* Required */
AccessControlRequestHeaders: 'origin,accept,content-type' /* Not required */
}, function(err, data) {
console.log(err || data);
});
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	Yes
Origin	Simulates the origin from which the request for cross-origin access is sent	String	Yes
AccessControlRequestMethod	Simulates the HTTP method of the request for cross-origin access	String	Yes
AccessControlRequestHeaders	Simulates the origin from which the request for cross-origin access is sent	String	Yes

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
- headers	Indicates the header information returned for a request	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- AccessControlAllowOrigin	Simulates the name of the origin from which the request for cross-origin access is sent, separated by commas. If the origin is not allowed, the header will not be returned. For example: *	String

Parameter Name	Description	Type
- AccessControlAllowMethods	Simulates the HTTP method of the request for cross-origin access, separated by commas. If the method is not allowed, the header will not be returned. Examples include PUT, GET, POST, DELETE, and HEAD	String
- AccessControlAllowHeaders	Simulates the header of the request for cross-origin access, separated by commas. If the simulation of any request header is not allowed, the header will not be returned. Examples include accept, content-type, origin, and authorization	String
- AccessControlExposeHeaders	Returned headers supported by cross-origin request, separated by commas. For example: ETag	String
- AccessControlMaxAge	Sets the validity period of the results obtained by OPTIONS for example: 3600	String
- OptionsForbidden	Indicates whether an OPTIONS request is forbidden. If the HTTP status code 403 is returned, the value is true	Boolean

Get Object ACL

Feature description

The API Get Object ACL is used to obtain access permission of an Object under a Bucket. Only the Bucket owner is allowed to perform the action.

Use Case

Call Get Object ACL:

```
cos.getObjectAcl({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
Key: '1.jpg', /* Required */
}, function(err, data) {
console.log(err || data);
});
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes

Parameter Name	Description	Type	Required
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	Yes

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object
- Owner	Owner of the identified resources	Object
- ID	ID of the Object owner. Format: qcs::cam::uin/<OwnerUin>:uin/lt;SubUin>; in case of root account, <OwnerUin> and <SubUin> are of the same value	String
- DisplayName	Name of the Object owner	String
- Grants	Lists information of authorized users and their permissions	Array
- Permission	Indicates the permissions granted to authorized users. Enumerated values: READ, WRITE, and FULL_CONTROL	String
- Grantee	Indicates information of authorized users. "type" can be RootAccount and Subaccount. In case of RootAccount, ID is specified as root account. In case of Subaccount, ID is specified as sub-account	Object
- DisplayName	Indicates the username	String

Parameter Name	Description	Type
- ID	Indicates the user ID. In case of root account, format: qcs::cam::uin/<OwnerUin>:uin/<OwnerUin> or qcs::cam::anyone:anyone (all users). In case of sub-account, format: qcs::cam::uin/<OwnerUin>:uin/<SubUin>	String

Put Object ACL

Feature description

This API (Put Object ACL) is used to configure ACL for an Object in a Bucket.
The number of ACL policies under a single Bucket is limited to 1000. Therefore, under a single Bucket, up to 999 files can be set with ACL permissions.

Use Case

Call Put Object ACL to modify file permissions:

```
cos.putObjectAcl({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
Key: '1.jpg', /* Required */
ACL: 'public-read', /* Not required */
}, function(err, data) {
console.log(err || data);
});
```

Grants a user the permission to read and write files.

```
cos.putObjectAcl({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
Key: '1.jpg', /* Required */
GrantFullControl: 'id="qcs::cam::uin/1001:uin/1001",id="qcs::cam::uin/1002:uin/1002"' // 1001 represents a uin
}, function(err, data) {
console.log(err || data);
});
```

Changes Bucket permissions using AccessControlPolicy

```
cos.putObjectAcl({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
Key: '1.jpg', /* Required */
AccessControlPolicy: {
"Owner": { // AccessControlPolicy must contain "owner".
"ID": 'qcs::cam::uin/459000000:uin/459000000' // 459000000 represents the QQ number of the Bucket user.
},
```

```
"Grants": [{
"Grantee": {
"ID": "qcs::cam::uin/10002:uin/10002", // 10002 repsents a QQ number
},
"Permission": "WRITE"
}]
}
}, function(err, data) {
console.log(err || data);
});
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	Yes
ACL	Defines the ACL attribute of Object. Valid values: private, public-read, public-read-write. Default value is private.	String	No
GrantRead	Grants read permission to the authorized user. Format: id=" ",id=" ". For authorization to a subaccount, id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>"; for authorization to the root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>". For example: 'id="qcs::cam::uin/123:uin/123", id="qcs::cam::uin/123:uin/456"'	String	No
GrantWrite	Grants write permission to the authorized user. Format: id=" ",id=" ". For authorization to a subaccount, id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>"; for authorization to the root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>". For example: 'id="qcs::cam::uin/123:uin/123", id="qcs::cam::uin/123:uin/456"'	String	No
GrantFullControl	Grants read and write permissions to the authorized user. Format: id=" ",id=" ". For authorization to a subaccount, id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>"; for authorization to the root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>". For example: 'id="qcs::cam::uin/123:uin/123", id="qcs::cam::uin/123:uin/456"'	String	No

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 204, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object

Delete Multiple Object

Feature description

This API (Delete Multiple Object) is used to delete files in batches in specific Bucket. A maximum of 1,000 Objects are allowed to be deleted in batches at a time. COS provides two modes for returned results: Verbose and Quiet. Verbose mode returns the result of deletion of each Object, while Quiet mode only returns the information of the Objects with an error.

Use Case

Deleting multiple files:

```
cos.deleteMultipleObject({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
Objects: [
{Key: '1.jpg'},
{Key: '2.zip'},
]
}, function(err, data) {
console.log(err || data);
});
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	Yes
Quiet	Boolean. Indicates whether the Quiet mode is enabled. True means Quiet mode is enabled, and False means Verbose mode is enabled. The default is False	Boolean	No
Objects	The file list to be deleted	Array	Yes

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 204, 403, and 404.	Number
- headers	Indicates the header information returned for a request	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 204, 403, and 404.	Number
- headers	Indicates the header information returned for a request	Object
- Deleted	Indicates the information of Object that has been deleted successfully	Array
- Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String
- Error	Indicates the information of Object that fail to be deleted	Array
- Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String

Parameter Name	Description	Type
- Code	Error code for failed deletion	String
- Message	Message indicating the deletion error	String

Put Object Copy

Feature description

This API (Put Object Copy) is used to copy a file from source path to the destination path. The recommended file size is 1 MB-5 GB. For any file above 5 GB, please use multipart upload (Upload - Copy). In the process of copying, file meta-attributes and ACLs can be modified. Users can use this API to move or rename a file, modify file attributes and create a copy.

Use Case

Call Put Object ACL:

```
cos.putObjectCopy({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
Key: '1.jpg', /* Required */
CopySource: 'test1.cos.ap-guangzhou.myqcloud.com/2.jpg', /* Required */
}, function(err, data) {
console.log(err || data);
});
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	Yes
CopySource	The path of source file URL. You can specify the history version with the versionid sub-resource	String	Yes
ACL	Defines the ACL attribute of Object. Valid values: private, public-read, public-read-write. Default value is private.	String	No

Parameter Name	Description	Type	Required
GrantRead	Grants read permission to the authorized user. Format: id=" ",id=" ". For authorization to a subaccount, id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>"; for authorization to the root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>". For example: 'id="qcs::cam::uin/123:uin/123", id="qcs::cam::uin/123:uin/456"'	String	No
GrantWrite	Grants write permission to the authorized user. Format: id=" ",id=" ". For authorization to a subaccount, id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>"; for authorization to the root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>". For example: 'id="qcs::cam::uin/123:uin/123", id="qcs::cam::uin/123:uin/456"'	String	No
GrantFullControl	Grants read and write permissions to the authorized user. Format: id=" ",id=" ". For authorization to a subaccount, id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>"; for authorization to the root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>". For example: 'id="qcs::cam::uin/123:uin/123", id="qcs::cam::uin/123:uin/456"'	String	No
MetadataDirective	Indicates whether to copy metadata. Enumerated values: Copy, Replaced. Default is Copy. If it is marked as Copy, the copying action will be performed directly, with the user metadata in the Header ignored; if it is marked as Replaced, the metadata will be modified based on the Header information. **If the destination path and the source path are the same, that is, the user attempts to modify the metadata, the value must be Replaced**	String	No
CopySourceIfModifiedSince	The action is performed if the Object has been modified since the specified time, otherwise error code 412 is returned. **It can be used with CopySourceIfNoneMatch. Using it with other conditions can cause a conflict.**	String	No
CopySourceIfUnmodifiedSince	The action is performed if the Object has not been modified since the specified time, otherwise error code 412 is returned. **It can be used with CopySourceIfMatch. Using it with other conditions can cause a conflict.**	String	No

Parameter Name	Description	Type	Required
CopySourceIfMatch	The action is performed if the Etag of Object is the same as the given one, otherwise error code 412 is returned. **It can be used with CopySourceIfUnmodifiedSince. Using it with other conditions can cause a conflict.**	String	No
CopySourceIfNoneMatch	The action is performed if the Etag of Object is different from the given one, otherwise error code 412 is returned. **It can be used with CopySourceIfModifiedSince. Using it with other conditions can cause a conflict.**	String	No
StorageClass	Indicates the storage class. Enumerated values: Standard, Standard_IA; the default is Standard	String	No
x-cos-meta- *	Indicates other custom file headers	String	No
CacheControl	Specifies the instructions that all caching mechanisms must obey throughout the request/response chain	String	No
ContentDisposition	Indicates an extension of MIME protocol. The MIME protocol instructs the MIME user agent how to display attached files	String	No
ContentEncoding	A pair of header fields used in HTTP to negotiate the "encoding format for text transmission"	String	No
ContentType	HTTP request content type defined in RFC 2616 (MIME), for example: `text/plain`	String	No
Expect	Indicates the specific server behavior requested	String	No
Expires	Response expiration date and time	String	No

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object

Parameter Name	Description	Type
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object
- ETag	The MD-5 algorithm check value of the file, such as `"22ca88419e2ed4721c23807c678adbe4c08a7880"` . **Be sure to enclose the value in double quotation marks**	String
- LastModified	Indicates the time when Object was last modified, such as 2017-06-23T12:33:27.000Z	String

Multipart Upload Operations

Initiate Multipart Upload

Feature description

This API (Initiate Multipart Upload) is used to initialize multipart upload. After the request is executed successfully, Upload ID is returned for the subsequent Upload Part requests.

Use Case

Call Initiate Multipart Upload:

```
cos.multipartInit({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
Key: '1.jpg', /* Required */
}, function(err, data) {
console.log(err || data);
});
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	Yes

Parameter Name	Description	Type	Required
CacheControl	Cache-Control, the caching policy defined in RFC 2616 and saved as Object metadata	String	No
ContentDisposition	The file name defined in RFC 2616, which will be saved as Object metadata.	String	No
ContentEncoding	The encoding format defined in RFC 2616, which is saved as Object metadata	String	No
ContentType	The content type (MIME) defined in RFC 2616, which is saved as Object metadata	String	No
Expires	The expiration time defined in RFC 2616, which is saved as Object metadata	String	No
ACL	Defines the ACL attribute of Object. Valid values: private, public-read, public-read-write. Default value is private.	String	No
GrantRead	Grants read permission to the authorized user. Format: id=" ",id=" ". For authorization to a subaccount, id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>"; for authorization to the root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>". For example: 'id="qcs::cam::uin/123:uin/123", id="qcs::cam::uin/123:uin/456"'	String	No
GrantWrite	Grants write permission to the authorized user. Format: id=" ",id=" ". For authorization to a subaccount, id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>"; for authorization to the root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>". For example: 'id="qcs::cam::uin/123:uin/123", id="qcs::cam::uin/123:uin/456"'	String	No
GrantFullControl	Grants read and write permissions to the authorized user. Format: id=" ",id=" ". For authorization to a subaccount, id="qcs::cam::uin/<OwnerUin>:uin/<SubUin>"; for authorization to the root account, id="qcs::cam::uin/<OwnerUin>:uin/<OwnerUin>". For example: 'id="qcs::cam::uin/123:uin/123", id="qcs::cam::uin/123:uin/456"'	String	No
StorageClass	Set the storage level of Object. Enumerated values: STANDARD, STANDARD_IA. Default: STANDARD	String	No
x-cos-meta- *	The header information allowed to be defined by users , which will be returned as Object metadata. The size is limited to 2 KB.	String	No

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
Bucket	The target Bucket for multipart upload	String
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String
UploadId	ID used in subsequent uploads	String

Upload Part

Feature description

Upload Part request is used to implement the multipart upload after initialization. The allowed number of parts is limited to 10,000, and the size of part should be between 1 MB and 5 GB.
You can obtain an uploadid when you use the API "Initiate Multipart Upload" to initiate multipart upload. This ID exclusively identifies this multipart data, and the relative position of this multipart in the entire file. Upload Part should be used with partNumber and uploadId. partNumber is the part No. and supports out-of-order upload.
If the uploadId and partNumber are the same, the parts uploaded later will overwrite the parts uploaded earlier. A 404 error "NoSuchUpload" will be returned if the uploadId does not exist.

Use Case

Call Upload Part:

```
cos.multipartUpload({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
Key: '1.jpg', /* Required */
}, function(err, data) {
console.log(err || data);
});
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	Yes
ContentLength	HTTP request content length defined in RFC 2616 (in bytes)	String	Yes
PartNumber	Part No.	String	Yes
UploadId	Upload task No.	String	Yes
Body	The content of the uploaded part, which can be `string` , `File object` or `Blob object`	String \ File \ Blob	Yes
Expect	When `Expect: 100-continue` is used, the request content will not be sent until the receipt of response from server	String	No
ContentMD5	Indicates the 128-bit content MD5 check value encoded using Base64, defined in RFC 1864. This header is used to check whether the file content has changed	String	No

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number

Parameter Name	Description	Type
- headers	Indicates the header information returned for a request	Object
- ETag	The MD-5 algorithm check value of the file, such as `"22ca88419e2ed4721c23807c678adbe4c08a7880"` . **Be sure to enclose the value in double quotation marks**	String

Complete Multipart Upload

Feature description

This API (Complete Multipart Upload) is used to complete the entire multipart upload. You must use this API to complete the multipart upload operation of the entire file when you have uploaded all parts using Upload Parts. When using this API, you need to provide the PartNumber and ETag for every part in request Body, to verify the accuracy of parts.
The merging of parts is required and takes several minutes, thus COS returns status code 200 immediately when the merging process begins. During merging, COS may returns blank information periodically to keep the connection active, until the merging process completes, upon which the COS will return the content of the merged parts in Body.

- When this API is called, "400 EntityTooSmall" is returned if the uploaded part is smaller than 1 MB.
- "400 InvalidPart" is returned if the numbers of uploaded parts are discontinuous.
- "400 InvalidPartOrder" is returned if the part information entries in the request Body are not sorted in ascending order according to their numbers.
- "404 NoSuchUpload" is returned if the UploadId does not exist when this API is called.

Use Case

Call Complete Multipart Upload:

```
cos.multipartComplete({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
Key: '1.zip', /* Required */
UploadId: '1521389146c60e7e198202e4e6670c5c78ea5d1c60ad62f1862f47294ec0fb8c6b7f3528a2', /* Required */
Parts: [
{PartNumber: '1', ETag: '"0cce40bdbaf2fa0ff204c20fc965dd3f"'},
]
}, function(err, data) {
console.log(err || data);
});
```

Parameters

Parameter Name	Description	Type	Required

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	Yes
UploadId	Upload task No.	String	Yes
Parts	Used to describe the block information list in this block upload	Array	Yes
PartNumber	Part No.	String	Yes
ETag	The MD5 algorithm check value of the multipart, such as `"22ca88419e2ed4721c23807c678adbe4c08a7880"` . **Be sure to enclose the value in double quotation marks**	String	Yes

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object
- Location	Domain name for public network access of the created Object	String
- Bucket	The target Bucket for multipart upload	String
- Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String

Parameter Name	Description	Type
- ETag	The MD5 algorithm check value of the merged file, such as `"22ca88419e2ed4721c23807c678adbe4c08a7880"` . **Be sure to enclose the value in double quotation marks**	String

List Parts

Feature description

This API (List Parts) is used to query the uploaded file chunks in a specific multipart upload, listing all the uploaded chunks under the specified UploadId.

Use Case

Call List Parts:

```
cos.multipartListPart({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
Key: '1.jpg', /* Required */
UploadId: '1521389146c60e7e198202e4e6670c5c78ea5d1c60ad62f1862f47294ec0fb8c6b7f3528a2', /* Required */
}, function(err, data) {
console.log(err || data);
});
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	Yes
UploadId	The ID of current multipart upload. You can obtain an uploadid when you use the API "Initiate Multipart Upload" to initiate multipart upload. This ID exclusively identifies this multipart data, and the relative position of this multipart in the entire file.	String	Yes
EncodingType	Indicates the encoding method of the returned value.	String	No
MaxParts	Maximum number of entries returned at a time. Default is 1,000	String	No
PartNumberMarker	Entries are listed using UTF-8 binary order by default, starting from the marker	String	No

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object
- Bucket	The target Bucket for multipart upload	String
- Encoding-type	The encoding method of the returned value	String
- Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String
- UploadId	Indicates the ID of current multipart upload.	String
- Initiator	Indicates the information of the initiator of current upload	Object
- - DisplayName	Indicates the name of the initiator of the upload	String
- - ID	Indicates the ID of the initiator of the upload. Format: qcs::cam::uin/<OwnerUin>:uin/<SubUin>; in case of root account, <OwnerUin> and <SubUin> are of the same value	String
- Owner	Indicates the information of the owner of these parts	Object
- - DisplayName	Name of the Bucket owner	String
- - ID	Indicates the ID of the Bucket owner, typically the user's UIN	String
- StorageClass	The storage class of uploaded parts. Enumerated values: Standard, Standard_IA	String

Parameter Name	Description	Type
- PartNumberMarker	Entries are listed using UTF-8 binary order by default, starting from the marker	String
- NextPartNumberMarker	If the returned entry is truncated, NextMarker represents the starting point of the next entry	String
- MaxParts	Maximum number of entries returned at a time	String
- IsTruncated	Indicates whether the returned entry is truncated. Value: 'true' or 'false'	String
- Part	Indicates the list of information about parts	Array
- - PartNumber	Indicates the part No.	String
- - LastModified	Indicates the time when the part was last modified	String
- - ETag	Indicates the MD5 algorithm check value of the part	String
- - Size	Indicates the part size (in bytes)	String

Abort Multipart Upload

Feature description

This API (Abort Multipart Upload) is used to abort a multipart upload operation and delete uploaded file chunks. When Abort Multipart Upload is called, the Upload Parts returns failure to any request that is using the Upload Parts. "404 NoSuchUpload" is returned if the UploadID does not exist.

It is recommended that you complete multipart upload in time or abort the upload operation for the reason that parts that have been uploaded but not aborted can take up storage, incurring cost.

Use Case

Call Abort Multipart Upload:

```
cos.multipartAbort({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
Key: '1.zip', /* Required */
UploadId: '1521389146c60e7e198202e4e6670c5c78ea5d1c60ad62f1862f47294ec0fb8c6b7f3528a2' /* Required */
}, function(err, data) {
console.log(err || data);
});
```

Parameters

Parameter Name	Description	Type	Required

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	Yes
UploadId	The ID of current multipart upload. You can obtain an uploadid when you use the API "Initiate Multipart Upload" to initiate multipart upload. This ID exclusively identifies this multipart data, and the relative position of this multipart in the entire file.	String	Yes

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object

List Multipart Uploads

Feature description

This API (List Multiparts Uploads) is used to query multipart upload operations that are still in process. Up to 1000 such operations can be listed each time.

Use Case

Gets an unfinished UploadId list with a prefix of 1.zip

```
cos.multipartList({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
Prefix: '1.zip', /* Not required */
}, function(err, data) {
console.log(err || data);
});
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Delimiter	Delimiter is a sign. Objects that contain the same string between the prefix, if specified, and the first occurrence of the delimiter after the prefix are grouped under a single result element: common prefix. If Prefix does not exist, the listing process starts from the beginning of the path.	String	No
EncodingType	Indicates the encoding format of the returned value. Valid value: url	String	No
Prefix	The returned Object key must be prefixed with Prefix. Note that the returned key will still contain Prefix when querying with prefix	String	No
MaxUploads	Set the maximum number of multipart returned. Valid values: 1-1,000. Default: 1,000	String	No
KeyMarker	Used together with upload-id-marker. • If upload-id-marker is not specified, entries whose ObjectNames are in front of key-marker (according to alphabetical order) will be listed. • If upload-id-marker is specified, entries whose ObjectNames are in front of key-marker (according to alphabetical order) will be listed, and entries whose ObjectNames are equal to key-marker and UploadIDs are in front of upload-id-marker (according to alphabetical order) will also be listed.	String	No

Parameter Name	Description	Type	Required
UploadIdMarker	Used together with key-marker. • If key-marker is not specified, upload-id-marker will be ignored. • If key-marker is specified, entries whose ObjectNames are in front of key-marker (according to alphabetical order) will be listed, and entries whose ObjectNames are equal to key-marker and UploadIDs are in front of upload-id-marker (according to alphabetical order) will also be listed.	String	No

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object
- Bucket	The target Bucket for multipart upload	String
- Encoding-Type	Indicates the encoding format of the returned value. Valid value: url	String
- KeyMarker	Entries will be listed starting from this key value	String
- UploadIdMarker	Entries will be listed starting from this UploadId value	String
- NextKeyMarker	If the returned entry is truncated, NextKeyMarker represents the starting point of the next entry	String
- NextUploadIdMarker	If the returned entry is truncated, UploadId represents the starting point of the next entry	String

Parameter Name	Description	Type
- MaxUploads	Sets the maximum number of multipart returned. Valid values: 1-1,000	String
- IsTruncated	Indicates whether the returned entry is truncated. Value: 'true' or 'false'	String
- Delimiter	Delimiter is a sign. Objects that contain the same string between the prefix, if specified, and the first occurrence of the delimiter after the prefix are grouped under a single result element: common prefix. If Prefix does not exist, the listing process starts from the beginning of the path.	String
- Prefix	The returned Object key must be prefixed with Prefix. Note that the returned key will still contain Prefix when querying with prefix	String
- CommonPrefixs	The same paths between Prefix and delimiter are grouped as the same type and defined as Common Prefix	Array
- Prefix	Displays detailed CommonPrefixs	String
- Upload	A collection of information about Upload	Array
- - Key	Object name	String
- - UploadId	Indicates the ID of current multipart upload	String
- StorageClass	Indicates the storage class of uploaded parts. Enumerated values: Standard, Standard_IA	String
- Initiator	Indicates the information of the initiator of current upload	Object
- - DisplayName	Indicates the name of the initiator of the upload	String
- - ID	ID of the initiator of the upload. Format: qcs::cam::uin/<OwnerUin>:uin/<SubUin>; in case of root account, <OwnerUin> and <SubUin> are of the same value	String
- Owner	Indicates the information of the owner of these parts	Object
- - DisplayName	Name of the Bucket owner	String
- - ID	ID of the Bucket owner. Format: qcs::cam::uin/<OwnerUin>:uin/<SubUin>; in case of root account, <OwnerUin> and <SubUin> are of the same value	String
- Initiated	Indicates the starting time for multipart upload	String

Multipart Upload/Replication Task

This kind of method is the encapsulation of the above native method, realizes the whole process of multipart uploading/replication, supports concurrent multipart uploading/replication, resuming upload from breakpoint, and the cancellation, suspension and restart of upload tasks.

Slice Upload File

Feature description

This API (Slice Upload File) is used to upload a file in parts.

Use Case

Call Slice Upload File:

```
cos.sliceUploadFile({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
Key: '1.zip', /* Required */
Body: file, /* Required */
TaskReady: function(taskId) { /* Not required */
console.log(taskId);
},
onHashProgress: function (progressData) { /* Not required */
console.log(JSON.stringify(progressData));
},
onProgress: function (progressData) { /* Not required */
console.log(JSON.stringify(progressData));
}
}, function(err, data) {
console.log(err || data);
});
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	Yes
Body	The content of the file uploaded, which can be File objects or Blob objects	File \ Blob	Yes
SliceSize	Indicates the size of part	String	No
AsyncLimit	Indicates the number of concurrent parts	String	No
StorageClass	The storage level of Object. Enumerated values: STANDARD, STANDARD_IA	String	No

Parameter Name	Description	Type	Required
TaskReady	Indicates the callback function when the upload task is created, returns a taskId that uniquely identifies the upload task, which can be used to cancel (cancelTask), pause (pauseTask), and restart (restartTask) the upload task.	Function	No
- taskId	Indicates the number of task to be uploaded	String	No
onHashProgress	Indicates the progress callback function to calculate a file's MD5 value. The callback parameter is the progress object progressData	Function	No
- progressData.loaded	Indicates the size of the verified portion of the file in bytes	Number	No
- progressData.total	Indicates the size of the entire file in bytes	Number	No
- progressData.speed	Indicates the verification speed in bytes/s	Number	No
- progressData.percent	The percentage of file verified in decimal, for example, 0.5 for 50%	Number	No
onProgress	Indicates the progress callback function of the uploaded file. The callback parameter is the progress object progressData	Function	No
- progressData.loaded	Indicates the size of the uploaded portion of the file in bytes	Number	No
- progressData.total	Indicates the size of the entire file in bytes	Number	No
- progressData.speed	Indicates the upload speed in bytes/s	Number	No
- progressData.percent	The percentage of file uploaded in decimal, for example, 0.5 for 50%	Number	No

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number

Parameter Name	Description	Type
- headers	Indicates the header information returned for a request	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object
- Location	Domain name for public network access of the created Object	String
- Bucket	The target Bucket for multipart upload	String
- Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String
- ETag	The MD5 algorithm check value of the merged file, such as `"22ca88419e2ed4721c23807c678adbe4c08a7880"` . **Be sure to enclose the value in double quotation marks**	String

Cancel Task

Feature description

This API (Cancel Task) is used to cancel the multipart upload task based on the taskid.

Use Case

Call Cancel Task.

```
var taskId = 'xxxxx'; /* Required */
cos.cancelTask(taskId);
```

Parameters

Parameter Name	Description	Type	Required
taskId	Indicates the number of the file upload task. When the sliceUploadFile method is called, its TaskReady callback returns the taskId of the upload task.	String	Yes

Pause Task

Feature description

This API (Pause Task) is used to pause the multipart upload task based on the taskid.

Use Case

Call Pause Task.

```
var taskId = 'xxxxx'; /* Required */
cos.pauseTask(taskId);
```

Parameters

Parameter Name	Description	Type	Required
taskId	Indicates the number of the file upload task. When the sliceUploadFile method is called, its TaskReady callback returns the taskId of the upload task.	String	Yes

Restart Task

Feature description

This API (Get Service) is used to restart the upload task based on the taskid, including the upload task that the user manually stopped (stop by calling pauseTask) or stopped because of an upload error.

Use Case

Call Restart Task:

```
var taskId = 'xxxxx'; /* Required */
cos.restartTask(taskId);
```

Parameters

Parameter Name	Description	Type	Required
taskId	Indicates the number of the file upload task. When the sliceUploadFile method is called, its TaskReady callback returns the taskId of the upload task.	String	Yes

Slice Copy File

Feature description

This API (Slice Copy File) is used to copy a file from the source path to the destination path through multipart copy. In the process of copying, file meta-attributes and ACLs can be modified. Users can use this API to move or rename a file, modify file attributes and create a copy.

Method prototype

Call Slice Copy File:

```
cos.sliceCopyFile({
Bucket: 'test-1250000000', /* Required */
Region: 'ap-guangzhou', /* Required */
Key: '1.zip', /* Required */
CopySource: 'test1.cos.ap-guangzhou.myqcloud.com/2.zip', /* Required */
onProgress:function (progressData) { /* Not required */
console.log(JSON.stringify(progressData));
}
},function (err,data) {
console.log(err || data);
});
```

Operation parameter description

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	Yes
CopySource	The path of source file URL. You can specify the history version with the versionid sub-resource	String	Yes
ChunkSize	Indicates the size of each part for multipart copy, which defaults to 1048576 (1 MB)	Number	No
SliceSize	Indicates the file size for multipart copy, which defaults to 5 GB	Number	No
onProgress	Indicates the progress callback function of the uploaded file. The callback parameter is the progress object progressData	Function	No
- progressData.loaded	Indicates the size of the uploaded portion of the file in bytes	Number	No
- progressData.total	Indicates the size of the entire file in bytes	Number	No
- progressData.speed	Indicates the upload speed in bytes/s	Number	No
- progressData.percent	The percentage of file uploaded in decimal, for example, 0.5 for 50%	Number	No

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. [Error Code Document] Object	
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
- statusCode	Indicates the HTTP status code returned for a request, such as 200, 403, and 404	Number
- headers	Indicates the header information returned for a request	Object
- Location	Domain name for public network access of the created Object	String
- Bucket	The target Bucket for multipart upload	String
- Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String
- ETag	The MD5 algorithm check value of the merged file, such as "22ca88419e2ed4721c23807c678adbe4c08a7880". Be sure to enclose the value in double quotation marks	String

Node.js SDK
Getting Started

Last updated：2018-07-20 18:02:39

Preparations for Development

Obtain SDK

Download XML JS SDK resources for COS service on Github from: tencentyun/cos-nodejs-sdk-v5 .

Download Demo from XML Node.js SDK Demo .

Introduce npm

Before development, you need to install environment dependencies: npm address

```
npm i cos-nodejs-sdk-v5 --save
```

Development environment

1. The use of SDK requires that your operating environment includes nodejs and npm. Nodejs 7.0 or above is recommended.
2. After installing npm, be sure to install npm install once under SDK's decompressed directory (install dependency package).
3. Go to Key Management on the console to obtain SecretId and SecretKey of your project.

> For more information on the definitions of SecretId, SecretKey, Bucket and other terms and how to obtain them, please see COS Glossary.

Getting Started

1. Go to the COS Console to create a bucket and obtain Bucket (bucket name) and Region (region name).
2. Go to Key Management on the console to obtain SecretId and SecretKey of your project.
3. Modify the SecretId, SecretKey, Bucket and Region to test the file upload by referring to the following code.

```
// Introduce the module
var COS = require('cos-nodejs-sdk-v5');
// Create an instance
var cos = new COS({
AppId: '1250000000',
SecretId: 'AKIDxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
```

```
SecretKey: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
});
// Multipart upload
cos.sliceUploadFile({
Bucket: 'test',
Region: 'ap-guangzhou',
Key: '1.zip',
FilePath: './1.zip'
}, function (err, data) {
console.log(err, data);
});
```

Related Documents

1. For more examples, please see XML Node.js SDK Demo.
2. For the complete API documentation, please see XML Node.js SDK API documentation.

API Documentation

Last updated：2018-07-25 18:03:56

For more information on the definitions of SecretId, SecretKey, Bucket, and other terms and how to obtain them, please see COS Glossary.

URL to Node.js SDK github: tencentyun/cos-nodejs-sdk-v5

Service Operation

Get Service

Feature description

This API (Get Service) is used to obtain the list of all Buckets under the current account. This API requires Authorization signature for verification and can only obtain the Bucket list under the account to which the AccessID in signature belongs.

Method prototype

Call Get Service:

```
cos.getService(params, function(err, data) {
if(err) {
console.log(err);
} else {
console.log(data);
}
});
```

Operation parameter description

- **No special parameters**

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
Owner	Provides the information of the Bucket owner	Object
uin	UIN of the Bucket owner	String
Buckets	Provides the information of the Bucket list returned this time	Array
Name	Bucket name	String
CreateDate	Date on which the Bucket was created. It takes on an ISO8601 format.	String

Bucket Operations

Head Bucket

Feature description

This API (Head Bucket) is used to determine whether the Bucket and the permission to access the Bucket exist. Head and Read have the same permission.

Method prototype

Call Head Bucket:

```
var params = {
Bucket : 'STRING_VALUE', /* Required */
Region : 'STRING_VALUE' /* Required */
};

cos.headBucket(params, function(err, data) {
if(err) {
console.log(err);
} else {
console.log(data);
}
});
```

Operation parameter description

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
BucketExist	Indicates whether a Bucket exists	Boolean
BucketAuth	Indicates whether a user has the Bucket permission	Boolean

Get Bucket

Feature description

This API (Get Bucket) is equivalent to List Object. It is used to list some or all of the Objects under the Bucket. Read permission is required to initiate this request.

Method prototype

Call Get Bucket:

```
var params = {
Bucket : 'STRING_VALUE', /* Required */
Region : 'STRING_VALUE', /* Required */
Prefix : 'STRING_VALUE', /* Not required */
Delimiter : 'STRING_VALUE', /* Not required */
Marker : 'STRING_VALUE', /* Not required */
MaxKeys : 'STRING_VALUE', /* Not required */
EncodingType : 'STRING_VALUE', /* Not required */
};

cos.getBucket(params, function(err, data) {
```

```
if(err) {
console.log(err);
} else {
console.log(data);
}
});
```

Operation parameter description

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Prefix	Indicates the prefix match, which is used to specify the prefix address of the returned object key	String	No
Delimiter	Delimiter is a sign. If Prefix exists, the same paths between Prefix and delimiter are grouped as the same type and defined as Common Prefix, and then all Common Prefixes are listed. If Prefix does not exist, the listing process starts from the beginning of the path	String	No
Marker	Entries are listed using UTF-8 binary order by default, starting from the marker	String	No
MaxKeys	Maximum number of entries returned at a time. Default is 1,000	String	No
EncodingType	Indicates the encoding method of the returned value.	String	No

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
CommonPrefixes	The same paths between Prefix and delimiter are grouped as the same type and defined as Common Prefix	Array

Parameter Name	Description	Type	
Prefix	Indicates the prefix match, which is used to specify the prefix address of the returned object key	String	No
Name	Bucket name	String	
Prefix	Prefix of an object key	String	No
Marker	Entries are listed using UTF-8 binary order by default, starting from the marker	String	
MaxKeys	Maximum number of entries returned at a time	String	
IsTruncated	Indicates whether the returned entry is truncated. Value: 'true' or 'false'	String	
NextMarker	If the returned entry is truncated, NextMarker represents the starting point of the next entry	String	
Encoding-Type	Encoding type of Delimiter, which is used for Marker, Prefix, NextMarker, and Key	String	
Contents	Metadata information	Array	
ETag	The SHA-1 algorithm check value of the file	String	
Size	File size (in bytes)	String	
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	
LastModified	Indicates the time when Object was last modified	String	
Owner	Information of the Bucket owner	Object	
ID	Bucket Owner's UID	String	

Put Bucket

Feature description

This API (Put Bucket) is used to create a Bucket under the specified account.

Method prototype

Call Put Bucket:

```
var params = {
Bucket : 'STRING_VALUE', /* Required */
Region : 'STRING_VALUE', /* Required */
ACL : 'STRING_VALUE', /* Not required */
GrantRead : 'STRING_VALUE', /* Not required */
```

```
GrantWrite : 'STRING_VALUE', /* Not required */
GrantFullControl : 'STRING_VALUE' /* Not required */
};

cos.putBucket(params, function(err, data) {
if(err) {
console.log(err);
} else {
console.log(data);
}
});
```

Operation parameter description

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
ACL	Allows users to define file permissions. Valid values: private, public-read, and public-read-write. Default value is private.	String	No
GrantRead	Grants read permission to the authorized user. Format: x-cos-grant-read: uin=" ",uin=" ". For authorization to a sub-account, uin="RootAcountID/SubAccountID". For authorization to the root account, uin="RootAcountID".	String	No
GrantWrite	Grants write permission to the authorized user. Format: x-cos-grant-write: uin=" ",uin=" ". For authorization to a sub-account, uin="RootAcountID/SubAccountID". For authorization to the root account, uin="RootAcountID".	String	No
GrantFullControl	Grants read and write permissions to the authorized user. Format: x-cos-grant-full-control: uin=" ",uin=" ". For authorization to a sub-account, uin="RootAcountID/SubAccountID". For authorization to the root account, uin="RootAcountID".	String	No

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
Location	Operation address of Bucket after it is created successfully	String

Delete Bucket

Feature description

This API (Delete Bucket) is used to delete a Bucket under the specified account. The Bucket must be empty before it can be deleted.

Method prototype

Call Delete Bucket:

```
var params = {
Bucket : 'STRING_VALUE', /* Required */
Region : 'STRING_VALUE' /* Required */
};

cos.deleteBucket(params, function(err, data) {
if(err) {
console.log(err);
} else {
console.log(data);
}
});
```

Operation parameter description

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
DeleteBucketSuccess	Indicates whether the deletion is successful	Boolean

Get Bucket ACL

Feature description

This API is used to read the ACL of a Bucket. Only the Bucket owner is allowed to perform the operation.

Method prototype

Call Get Bucket ACL:

```
var params = {
Bucket : 'STRING_VALUE', /* Required */
Region : 'STRING_VALUE' /* Required */
};

cos.getBucketAcl(params, function(err, data) {
if(err) {
console.log(err);
} else {
console.log(data);
}
});
```

Operation parameter description

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
Owner	Indicates the owner of the resource	Object
uin	QQ number of the user	String
AccessControlList	Information of the authorized user and permissions	Object
Grant	Specific authorization information	Array
Permission	Permission information. Enumerated values: READ, WRITE, and FULL_CONTROL	String
Grantee	Resource information of the authorized user	Object
uin	QQ number of the authorized user or 'anonymous'	String
Subacount	QQ number of the sub-account	String

Put Bucket ACL

Feature description

This API is used to write ACL for a Bucket. Importing a new ACL using Put Bucket ACL operation will overwrite the existing ACL. Only the owner is allowed to perform the operation.

Method prototype

Call Put Bucket ACL:

```
var params = {
Bucket : 'STRING_VALUE', /* Required */
Region : 'STRING_VALUE', /* Required */
ACL : 'STRING_VALUE', /* Not required */
GrantRead : 'STRING_VALUE', /* Not required */
GrantWrite : 'STRING_VALUE', /* Not required */
GrantFullControl : 'STRING_VALUE' /* Not required */
};

cos.putBucketAcl(params, function(err, data) {
if(err) {
console.log(err);
} else {
console.log(data);
```

```
}
});
```

Operation parameter description

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
ACL	Allows users to define file permissions. Valid values: private and public-read. Default value: private.	String	No
GrantRead	Grants read permission to the authorized user. Format: x-cos-grant-read: uin=" ",uin=" ". For authorization to a sub-account, uin="RootAcountID/SubAccountID". For authorization to the root account, uin="RootAcountID".	String	No
GrantWrite	Grants write permission to the authorized user. Format: x-cos-grant-write: uin=" ",uin=" ". For authorization to a sub-account, uin="RootAcountID/SubAccountID". For authorization to the root account, uin="RootAcountID".	String	No
GrantFullControl	Grants read and write permissions to the authorized user. Format: x-cos-grant-full-control: uin=" ",uin=" ". For authorization to a sub-account, uin="RootAcountID/SubAccountID". For authorization to the root account, uin="RootAcountID".	String	No

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object

Parameter Name	Description	Type
BucketGrantSuccess	Whether authorization is successful	Boolean

Get Bucket CORS

Feature description

This API (Get Bucket CORS) is used to read cross-origin access configurations.

Method prototype

Call Get Bucket CORS:

```
var params = {
Bucket : 'STRING_VALUE', /* Required */
Region : 'STRING_VALUE' /* Required */
};

cos.getBucketCors(params, function(err, data) {
if(err) {
console.log(err);
} else {
console.log(data);
}
});
```

Operation parameter description

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object

Parameter Name	Description	Type
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
CORSRule	A collection of configurations	Array
AllowedMethod	Allowed HTTP operations. Enumerated values: Get, Put, Head, Post, and Delete	Array
AllowedOrigin	Allowed access sources. The wildcard "*" is supported.	Array
AllowedHeader	When an OPTIONS request is sent, notifies the server about which custom HTTP request headers are allowed for subsequent requests.	Array
ExposeHeader	Sets the custom header information that can be received by the browser from the server end	Array
MaxAgeSeconds	Sets the validity period of the results obtained by OPTIONS	String
ID	Rule name	String

Put Bucket CORS

Feature description

This API (Put Bucket CORS) is used to read and write cross-origin access configurations.

Method prototype

Call Put Bucket CORS:

```
var params = {
Bucket : 'STRING_VALUE', /* Required */
Region : 'STRING_VALUE', /* Required */
CORSRules : [
{
ID : 'STRING_VALUE', /* Not required */
AllowedMethods: [ /* Required */
'STRING_VALUE',
...
],
AllowedOrigins: [ /* Required */
'STRING_VALUE',
...
],
AllowedHeaders: [ /* Not required */
'STRING_VALUE',
...
],
ExposeHeaders: [ /* Not required */
```

```
'STRING_VALUE',
...
],
MaxAgeSeconds: 'STRING_VALUE' /* Not required */
},
....
]
};

cos.putBucketCors(params, function(err, data) {
if(err) {
console.log(err);
} else {
console.log(data);
}
});
```

Operation parameter description

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
CORSRules	A collection of cross-origin rules	Array	No
ID	Rule name	String	No
AllowedMethods	Allowed HTTP operations. Enumerated values: Get, Put, Head, Post, and Delete	Array	Yes
AllowedOrigins	Allowed access sources. The wildcard "*" is supported. The protocol, port and domain name must be consistent.	Array	Yes
AllowedHeaders	When an OPTIONS request is sent, notifies the server about which custom HTTP request headers are allowed for subsequent requests. Wildcard "*" is supported.	Array	No
ExposeHeaders	Sets the custom header information that can be received by the browser from the server end	Array	No
MaxAgeSeconds	Sets the validity period of the results obtained by OPTIONS	String	No

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
PutBucketCorsSucesss	Whether Bucket CORS is configured successfully	Boolean

Delete Bucket CORS

Feature description

This API (Delete Bucket CORS) is used to delete cross-origin access configurations.

Method prototype

Call Delete Bucket CORS:

```
var params = {
Bucket : 'STRING_VALUE', /* Required */
Region : 'STRING_VALUE' /* Required */
};

cos.deleteBucketCors(params, function(err, data) {
if(err) {
console.log(err);
} else {
console.log(data);
}
});
```

Operation parameter description

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
DeleteBucketCorsSuccess	Whether Bucket CORS is deleted successfully	Boolean

Get Bucket Location

Feature description

This API (Get Bucket Location) is used to obtain the region information of the Bucket. Only the Bucket owner is allowed to read the information.

Method prototype

Call Get Bucket Location:

```
var params = {
Bucket : 'STRING_VALUE', /* Required */
Region : 'STRING_VALUE' /* Required */
};

cos.getBucketLocation(params, function(err, data) {
if(err) {
console.log(err);
} else {
console.log(data);
}
});
```

Operation parameter description

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
LocationConstraint	Region where the Bucket resides. Enumerated values: china-east, china-south, china-north, china-west, and singapore	String

Get Bucket Tagging

Feature description

This API (Get Bucket Tagging) is used to obtain tags of a specified Bucket.

Method prototype

Call Get Bucket Tagging:

```
var params = {
Bucket : 'STRING_VALUE', /* Required */
Region : 'STRING_VALUE' /* Required */
};

cos.getBucketTagging(params, function(err, data) {
if(err) {
console.log(err);
} else {
console.log(data);
}
});
```

Operation parameter description

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
Tags	A collection of Bucket tags	Array
Key	Type name of Tag	String
Value	Tag's value	String

Put Bucket Tagging

Feature description

This API (Put Bucket Tagging) is used to tag a specified Bucket.

Operation parameter description

Call Put Bucket Tagging:

```
var params = {
Bucket : 'STRING_VALUE', /* Required */
Region : 'STRING_VALUE', /* Required */
Tags : [
{
Key : 'key1', /* Required */
Value : 'value1' /* Required */
},
...
]
};

cos.putBucketTagging(params, function(err, data) {
if(err) {
console.log(err);
} else {
console.log(data);
}
});
```

Operation parameter description

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Tags	A collection of Bucket tags	Array	Yes
Key	Type name of Tag	String	Yes
Value	Tag's value	String	Yes

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
PutBucketTaggingSuccess	Whether Tag is configured successfully	Boolean

Delete Bucket Tagging

Feature description

This API (Delete Bucket Tagging) is used to delete tags of a specified Bucket.

Method prototype

Call Delete Bucket Tagging:

```
var params = {
Bucket : 'STRING_VALUE', /* Required */
Region : 'STRING_VALUE' /* Required */
};

cos.deleteBucketTagging(params, function(err, data) {
if(err) {
console.log(err);
} else {
```

```
console.log(data);
}
});
```

Operation parameter description

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
DeleteBucketTaggingSuccess	Whether Bucket tag is deleted successfully	Boolean

Object Operations

Head Object

Feature description

This API (Head Object) is used to get the metadata of an Object. It has the same permissions as Get Object.

Method prototype

Call Head Object:

```
var params = {
Bucket : 'STRING_VALUE', /* Required */
Region : 'STRING_VALUE', /* Required */
Key : 'STRING_VALUE', /* Required */
IfModifiedSince : 'STRING_VALUE' /* Not required */
```

```
};

cos.headObject(params, function(err, data) {
if(err) {
console.log(err);
} else {
console.log(data);
}
});
```

Operation parameter description

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	Yes
IfModifiedSince	If Object is modified after the specified time, the Object meta information is returned	String	No

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
x-cos-object-type	Indicates whether the Object is appendable for upload. Enumerated values: normal or appendable	String
x-cos-storage-class	The storage level of Object. Enumerated values: Standard, and Standard_IA	String
x-cos-meta-*	User-defined metadata	String

Parameter Name	Description	Type
NotModified	If IfModifiedSince is used for request and the file is not modified, the value is true	Boolean

Get Object

Feature description

This API (Delete Object) is used to download a file (Object) locally. This operation requires that the user have the read permission for the target Object or the read permission for the target Object be available for everyone (public-read).

Method prototype

Call Get Object:

```
var params = {
Bucket : 'STRING_VALUE', /* Required */
Region : 'STRING_VALUE', /* Required */
Key : 'STRING_VALUE', /* Required */
ResponseContentType : 'STRING_VALUE', /* Not required */
ResponseContentLanguage : 'STRING_VALUE', /* Not required */
ResponseExpires : 'STRING_VALUE', /* Not required */
ResponseCacheControl : 'STRING_VALUE', /* Not required */
ResponseContentDisposition : 'STRING_VALUE', /* Not required */
ResponseContentEncoding : 'STRING_VALUE', /* Not required */
Range : 'STRING_VALUE', /* Not required */
IfModifiedSince : 'STRING_VALUE', /* Not required */
Output : 'STRING_VALUE' || 'WRITE_STRING' /* Required */
};

cos.getObject(params, function(err, data) {
if(err) {
console.log(err);
} else {
console.log(data);
}
});
```

Operation parameter description

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes

Parameter Name	Description	Type	Required
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	Yes
ResponseContentType	Sets the Content-Type parameter in the response header	String	No
ResponseContentLanguage	Sets the Content-Language parameter in the response header	String	No
ResponseExpires	Sets the Content-Expires parameter in the response header	String	No
ResponseCacheControl	Sets the Cache-Control parameter in the response header	String	No
ResponseContentDisposition	Sets the Content-Disposition parameter in the response header	String	No
ResponseContentEncoding	Sets the Content-Encoding parameter in the response header	String	No
Range	The specified range of file download defined in RFC 2616 (in bytes)	String	No
IfModifiedSince	The file content is returned if the file has been modified after the specified time	String	No
Output	A file path that needs outputting or a write stream	String / WriteStream	Yes

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
x-cos-object-type	Indicates whether the Object is appendable for upload. Enumerated values: normal or appendable	String

Parameter Name	Description	Type
x-cos-storage-class	The storage level of Object. Enumerated values: Standard, and Standard_IA	String
x-cos-meta-*	User-defined metadata	String
NotModified	If IfModifiedSince is used for request and the file is not modified, the value is true	Boolean

Put Object

Feature description

This API (Put Object) is used to upload a file (Object) to the specified Bucket.

> **Note:** Key (File name) cannot end with `/`. Otherwise, it will be identified as a folder.

Method prototype

Call Put Object:

```
cos.putObject({
Bucket : 'STRING_VALUE', /* Required */
Region : 'STRING_VALUE', /* Required */
Key : 'STRING_VALUE', /* Required */
Body: fs.createReadStream('./a.zip'), /* Required */
onProgress: function (progressData) {
console.log(progressData);
},
}, function(err, data) {
if(err) {
console.log(err);
} else {
console.log(data);
}
});
```

Operation parameter description

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes

Parameter Name	Description	Type	Required
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	Yes
CacheControl	The caching policy defined in RFC 2616, which is saved as Object metadata	String	No
ContentDisposition	The file name defined in RFC 2616, which is saved as Object metadata	String	No
ContentEncoding	The encoding format defined in RFC 2616, which is saved as Object metadata	String	No
ContentLength	HTTP request content length defined in RFC 2616 (in bytes)	String	Yes
ContentType	The content type (MIME) defined in RFC 2616, which is saved as Object metadata	String	No
Expect	If Expect: 100-continue is used, the request content will not be sent until the receipt of response from server	String	No
Expires	The expiration time defined in RFC 2616, which is saved as Object metadata	String	No
ContentSha1	The 160-bit content SHA-1 algorithm check value defined in RFC 3174	String	No
ACL	Allows users to define file permission. Valid values: private, and public-read	String	No
GrantRead	Grants read permission to the authorized user. Format: x-cos-grant-read: uin=" ",uin=" ". For authorization to a sub-account, uin="RootAcountID/SubAccountID". For authorization to the root account, uin="RootAcountID"	String	No
GrantWrite	Grants write permission to the authorized user. Format: x-cos-grant-write: uin=" ",uin=" ". For authorization to a sub-account, uin="RootAcountID/SubAccountID". For authorization to the root account, uin="RootAcountID"	String	No
GrantFullControl	Grants read and write permissions to the authorized user. Format: x-cos-grant-full-control: uin=" ",uin=" ". For authorization to a sub-account, uin="RootAcountID/SubAccountID". For authorization to the root account, uin="RootAcountID"	String	No
x-cos-meta- *	The header information that can be defined by users, which is returned as Object metadata. The size is limited to 2K.	String	No

Parameter Name	Description	Type	Required
Body	Input file path or file stream	String/ Stream	Yes
onProgress	Progress callback function. Callback is an object which includes progress information	Function	No

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
ETag	Returns the MD5 algorithm check value of the file. ETag value can be used to check whether the Object content has changed	String

Delete Object

Feature description

This API (Delete Object) is used to delete a file (Object).

Method prototype

Call Delete Object:

```
var params = {
Bucket : 'STRING_VALUE', /* Required */
Region : 'STRING_VALUE', /* Required */
Key : 'STRING_VALUE' /* Required */
};

cos.deleteObject(params, function(err, data) {
if(err) {
console.log(err);
} else {
console.log(data);
}
});
```

Operation parameter description

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	Yes

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
DeleteObjectSuccess	Whether a file is deleted successfully	Boolean
BucketNotFound	If the specific Bucket is not found, the value is true	Boolean

Options Object

Feature description

This API (Options Object) is used to implement a pre-request for cross-origin access. That is , an OPTIONS request is sent to the server to verify whether cross-origin operations are possible.

Method prototype

Call Options Object:

```
var params = {
Bucket : 'STRING_VALUE', /* Required */
Region : 'STRING_VALUE', /* Required */
Key : 'STRING_VALUE', /* Required */
Origin : 'STRING_VALUE', /* Required */
AccessControlRequestMethod : 'STRING_VALUE', /* Required */
```

```
AccessControlRequestHeaders : 'STRING_VALUE' /* Not required */
};

cos.optionsObject(params, function(err, data) {
if(err) {
console.log(err);
} else {
console.log(data);
}
});
```

Operation parameter description

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	Yes
Origin	Simulates the origin from which the request for cross-origin access is sent	String	Yes
AccessControlRequestMethod	Simulates the HTTP method of the request for cross-origin access	String	Yes
AccessControlRequestHeaders	Simulates the header of the request for cross-origin access	String	No

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object

Parameter Name	Description	Type
AccessControlAllowOrigin	Simulates the origin from which the request for cross-origin access is sent. If the origin is not allowed, the header will not be returned	String
AccessControlAllowMethods	Simulates the HTTP method of the request for cross-origin access. If the method is not allowed, the header will not be returned	String
AccessControlAllowHeaders	Simulates the header of the request for cross-origin access. If the simulation of any request header is not allowed, the header will not be returned	String
AccessControlExposeHeaders	Returned headers supported by cross-origin request, separated by commas	String
AccessControlMaxAge	Sets the validity period of the results obtained by OPTIONS	String

Get Object ACL

Feature description

This API (Get Object ACL) is used to read the Object ACL. Only the Object owner is allowed to perform the operation.

Method prototype

Call Get Object ACL:

```
var params = {
Bucket : 'STRING_VALUE', /* Required */
Region : 'STRING_VALUE', /* Required */
Key : 'STRING_VALUE' /* Required */
};

cos.getObjectAcl(params, function(err, data) {
if(err) {
console.log(err);
} else {
console.log(data);
}
});
```

Operation parameter description

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes

Parameter Name	Description	Type	Required
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	Yes

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
Owner	Indicates the owner of the resource	Object
uin	QQ number of the user	String
AccessControlList	Information of the authorized user and permissions	Object
Grant	Specific authorization information	Array
Permission	Permission information. Enumerated values: READ, WRITE, and FULL_CONTROL	String
Grantee	Resource information of the authorized user	Object
uin	QQ number of the authorized user or 'anonymous'	String
Subacount	QQ number of the sub-account	String

Put Object ACL

Feature description

This API (Put Object ACL) is used to write ACL for an Object.

Method prototype

Call Put Object ACL:

```
var params = {
Bucket : 'STRING_VALUE', /* Required */
Region : 'STRING_VALUE', /* Required */
Key : 'STRING_VALUE', /* Required */
```

```
ACL : 'STRING_VALUE', /* Not required */
GrantRead : 'STRING_VALUE', /* Not required */
GrantWrite : 'STRING_VALUE', /* Not required */
GrantFullControl : 'STRING_VALUE' /* Not required */
};

cos.putObjectAcl(params, function(err, data) {
if(err) {
console.log(err);
} else {
console.log(data);
}
});
```

Operation parameter description

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	Yes
ACL	Allows users to define file permissions. Valid values: private, public-read, and public-read-write. Default value is private.	String	No
GrantRead	Grants read permission to the authorized user. Format: x-cos-grant-read: uin=" ",uin=" ". For authorization to a sub-account, uin="RootAcountID/SubAccountID". For authorization to the root account, uin="RootAcountID".	String	No
GrantWrite	Grants write permission to the authorized user. Format: x-cos-grant-write: uin=" ",uin=" ". For authorization to a sub-account, uin="RootAcountID/SubAccountID". For authorization to the root account, uin="RootAcountID".	String	No
GrantFullControl	Grants read and write permissions to the authorized user. Format: x-cos-grant-full-control: uin=" ",uin=" ". For authorization to a sub-account, uin="RootAcountID/SubAccountID".For authorization to the root account, uin="RootAcountID".	String	No

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object

Delete Multiple Object

Feature description

This API (Delete Multiple Object) is used for batch deletion of files. A maximum of 1,000 files can be deleted at a time. COS provides two modes for returned results: Verbose and Quiet. Verbose mode will return the result of deletion of each Object, while Quiet mode only returns the information of the Objects with an error.

Method prototype

Call Delete Multiple Object:

```
var params = {
Bucket : 'STRING_VALUE', /* Required */
Region : 'STRING_VALUE', /* Required */
Quiet : 'BOOLEAN_VALUE', /* Not required */
Objects : [
{
Key : 'STRING_VALUE' /* Required */
}
]
};

cos.deleteMultipleObject(params, function(err, data) {
if(err) {
console.log(err);
} else {
console.log(data);
}
});
```

Operation parameter description

Parameter Name	Description	Type	Required

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Quiet	Boolean. Indicates whether the Quiet mode is enabled. True means Quiet mode is enabled, and False means Verbose mode is enabled. Default is False.	Boolean	No
Objects	List of files to be deleted	Array	No
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	Yes

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
Deleted	Indicates the information of Object that has been deleted successfully	Array
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String
Error	Indicates the information of Object that failed to be deleted	Array
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String
Code	Error code for failed deletion	String
Message	Message indicating the deletion error	String

Multipart Upload Operations

Initiate Multipart Upload

Feature description

This API (Initiate Multipart Upload) is used to initialize multipart upload. After the request is executed successfully, Upload ID is returned for the subsequent Upload Part requests.

Method prototype

Call Initiate Multipart Upload:

```
cos.multipartInit({
Bucket : 'STRING_VALUE', /* Required */
Region : 'STRING_VALUE', /* Required */
Key : 'STRING_VALUE', /* Required */
}, function(err, data) {
if(err) {
console.log(err);
} else {
console.log(data);
}
});
```

Operation parameter description

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	Yes
CacheControl	The caching policy defined in RFC 2616, which is saved as Object metadata	String	No
ContentDisposition	The file name defined in RFC 2616, which is saved as Object metadata	String	No
ContentEncoding	The encoding format defined in RFC 2616, which is saved as Object metadata	String	No
ContentType	The content type (MIME) defined in RFC 2616, which is saved as Object metadata	String	No
Expires	The expiration time defined in RFC 2616, which is saved as Object metadata	String	No
ACL	Allows users to define file permission. Valid values: private and public-read	String	No

Parameter Name	Description	Type	Required
GrantRead	Grants read permission to the authorized user. Format: x-cos-grant-read: uin=" ",uin=" ". For authorization to a sub-account, uin="RootAcountID/SubAccountID". For authorization to the root account, uin="RootAcountID"	String	No
GrantWrite	Grants write permission to the authorized user. Format: x-cos-grant-write: uin=" ",uin=" ". For authorization to a sub-account, uin="RootAcountID/SubAccountID". For authorization to the root account, uin="RootAcountID"	String	No
GrantFullControl	Grants read and write permissions to the authorized user. Format: x-cos-grant-full-control: uin=" ",uin=" ". For authorization to a sub-account, uin="RootAcountID/SubAccountID". For authorization to the root account, uin="RootAcountID"	String	No
StorageClass	Sets the storage level of Object. Enumerated values: Standard and Standard_IA. Default is Standard (this is only supported for South China region)	String	No
x-cos-meta- *	The header information that can be defined by users, which is returned as Object metadata. The size is limited to 2K.	String	No

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
Bucket	The target Bucket for multipart upload	String
Key	Object name	String
UploadId	ID used in subsequent uploads	String

Upload Part

Feature description

This API (Upload Part) is used to implement multipart upload after initialization. A file can be split into 10000 chunks at most (minimum is 1) for multipart upload, and the size of each file chunk should be between 1 MB and 5 GB. Upload Part should be used with partNumber and uploadID. partNumber is the part No. and supports out-of-order upload.

Method prototype

Call Upload Part:

```
var params = {
Bucket : 'STRING_VALUE', /* Required */
Region : 'STRING_VALUE', /* Required */
Key : 'STRING_VALUE', /* Required */
ContentLength : 'STRING_VALUE', /* Required */
Expect : 'STRING_VALUE', /* Not required */
ContentSha1 : 'STRING_VALUE', /* Not required */
PartNumber : 'STRING_VALUE', /* Required */
UploadId : 'STRING_VALUE', /* Required */
};

cos.multipartUpload(params, function(err, data) {
if(err) {
console.log(err);
} else {
console.log(data);
}
});
```

Operation parameter description

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	Yes
ContentLength	HTTP request content length defined in RFC 2616 (in bytes)	String	Yes
Expect	If Expect: 100-continue is used, the request content will not be sent until the receipt of response from server	String	No
ContentSha1	The 160-bit content SHA-1 algorithm check value defined in RFC 3174	String	No

Parameter Name	Description	Type	Required
PartNumber	Part No.	String	Yes
UploadId	Upload task No.	String	Yes

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
ETag	ETag value of a part, which is sha1 check value	String

Complete Multipart Upload

Feature description

This API (Complete Multipart Upload) is used to complete the entire multipart upload. After you have uploaded all the file chunks using Upload Parts, you can use this API to complete the upload. When using this API, you need to provide the PartNumber and ETag for every chunk in Body, to verify the accuracy of chunks.

Method prototype

Call Complete Multipart Upload:

```
var params = {
Bucket : 'STRING_VALUE', /* Required */
Region : 'STRING_VALUE', /* Required */
Key : 'STRING_VALUE', /* Required */
UploadId : 'STRING_VALUE', /* Required */
Parts : [
{
PartNumber : 'STRING_VALUE', /* Required */
ETag : 'STRING_VALUE' /* Required */
},
...
]
};
```

```
cos.multipartComplete(params, function(err, data) {
if(err) {
console.log(err);
} else {
console.log(data);
}
});
```

Operation parameter description

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	Yes
UploadId	Upload task No.	String	Yes
Parts	ETag information of a part	Array	Yes
PartNumber	Part No.	String	Yes
ETag	ETag value of a part, which is sha1 check value. It must be enclosed in double quotes, such as: "3a0f1fd698c235af9cf098cb74aa25bc".	String	Yes

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
Location	Domain name for public network access of the created Object	String
Bucket	The target Bucket for multipart upload	String
Key	Object name	String

Parameter Name	Description	Type
ETag	The MD5 algorithm check value of the file	String

List Parts

Feature description

This API (List Parts) is used to query the uploaded parts in a specific multipart upload process.

Method prototype

Call List Parts:

```
var params = {
Bucket : 'STRING_VALUE', /* Required */
Region : 'STRING_VALUE', /* Required */
Key : 'STRING_VALUE', /* Required */
UploadId : 'STRING_VALUE', /* Required */
EncodingType : 'STRING_VALUE', /* Not required */
MaxParts : 'STRING_VALUE', /* Not required */
PartNumberMarker : 'STRING_VALUE' /* Not required */
};

cos.multipartListPart(params, function(err, data) {
if(err) {
console.log(err);
} else {
console.log(data);
}
});
```

Operation parameter description

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	Yes
UploadId	Upload task No.	String	Yes
EncodingType	Indicates the encoding method of the returned value	String	No

Parameter Name	Description	Type	Required
MaxParts	Maximum number of entries returned at a time. Default is 1,000	String	No
PartNumberMarker	Entries are listed using UTF-8 binary order by default, starting from the marker	String	No

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
Bucket	The target Bucket for multipart upload	String
Encoding-type	The encoding method of the returned value	String
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String
UploadID	Indicates the ID of current multipart upload	String
Initiator	Indicates the information of the initiator of current upload. Child node includes UID	Object
UID	Developer's APPID	String
Owner	Indicates the information of the owner of these uploaded parts. Child node includes UID	Object
UID	Owner's QQ number	String
StorageClass	The storage class of uploaded parts. Enumerated values: Standard, Standard_IA	String
PartNumberMarker	Entries are listed using UTF-8 binary order by default, starting from the marker	String
NextPartNumberMarker	If the returned entry is truncated, NextMarker represents the starting point of the next entry	String
MaxParts	Maximum number of entries returned at a time	String

Parameter Name	Description	Type
IsTruncated	Indicates whether the returned entry is truncated. Value: 'true' or 'false'	String
Part	A collection of information about parts	Array
PartNumber	Part No.	String
LastModified	Indicates the time when the part was last modified	String
Etag	Indicates the SHA-1 algorithm check value of the part	String
Size	Indicates the part size (in bytes)	String

Abort Multipart Upload

Feature description

This API (Abort Multipart Upload) is used to abort a multipart upload operation and delete uploaded file chunks. When Abort Multipart Upload is called, a failure is returned for any request that is using Upload Parts.

Method prototype

Call Abort Multipart Upload:

```
var params = {
Bucket : 'STRING_VALUE', /* Required */
Region : 'STRING_VALUE', /* Required */
Key : 'STRING_VALUE', /* Required */
UploadId : 'STRING_VALUE' /* Required */
};

cos.multipartAbort(params, function(err, data) {
if(err) {
console.log(err);
} else {
console.log(data);
}
});
```

Operation parameter description

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes

Parameter Name	Description	Type	Required
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	Yes
UploadId	Upload task No.	String	Yes

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
MultipartAbortSuccess	Whether Multipart Abort is successful	Boolean

List Multipart Uploads

Feature description

This API (List Multiparts Uploads) is used to query multipart upload operations that are still in process. Up to 1000 such operations can be listed each time.

Method prototype

Call List Multipart Uploads:

```
var params = {
Bucket : 'STRING_VALUE', /* Required */
Region : 'STRING_VALUE', /* Required */
Delimiter : 'STRING_VALUE', /* Not required */
EncodingType : 'STRING_VALUE', /* Not required */
Prefix : 'STRING_VALUE', /* Not required */
MaxUploads : 'STRING_VALUE', /* Not required */
KeyMarker : 'STRING_VALUE', /* Not required */
UploadIdMarker : 'STRING_VALUE' /* Not required */
};

cos.multipartList(params, function(err, data) {
if(err) {
```

```
console.log(err);
} else {
console.log(data);
}
});
```

Operation parameter description

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Delimiter	Delimiter is a sign. If Prefix exists, the same paths between Prefix and delimiter are grouped as the same type and defined as Common Prefix, and then all Common Prefixes are listed. If Prefix does not exist, the listing process starts from the beginning of the path	String	No
EncodingType	Indicates the encoding method of the returned value	String	No
Prefix	Indicates the prefix match, which is used to specify the prefix address of the returned object key	String	No
MaxUploads	Maximum number of entries returned at a time. Default is 1,000	String	No
KeyMarker	Used together with upload-id-marker. • If upload-id-marker is not specified, entries whose ObjectNames are in front of key-marker (according to alphabetical order) will be listed. • If upload-id-marker is specified, entries whose ObjectNames are in front of key-marker (according to alphabetical order) will be listed, and entries whose ObjectNames are equal to key-marker and UploadIDs are in front of upload-id-marker (according to alphabetical order) will also be listed.	String	No
UploadIdMarker	Used together with key-marker. • If key-marker is not specified, upload-id-marker will be ignored. • If key-marker is specified, entries whose ObjectNames are in front of key-marker (according to alphabetical order) will be listed, and entries whose ObjectNames are equal to key-marker and UploadIDs are in front of upload-id-marker (according to alphabetical order) will also be listed.	String	No

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type	
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object	
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object	
Bucket	The target Bucket for multipart upload	String	
Encoding-type	The encoding method of the returned value	String	
KeyMarker	Entries will be listed starting from this key value	String	
UploadIdMarker	Entries will be listed starting from this UploadId value	String	
NextKeyMarker	If the returned entry is truncated, NextKeyMarker represents the starting point of the next entry	String	
NextUploadIdMarker	If the returned entry is truncated, UploadId represents the starting point of the next entry	String	
IsTruncated	Indicates whether the returned entry is truncated. Value: 'true' or 'false'	String	
delimiter	Delimiter is a sign. If Prefix exists, the same paths between Prefix and delimiter are grouped as the same type and defined as Common Prefix, and then all Common Prefixes are listed. If Prefix does not exist, the listing process starts from the beginning of the path.	String	
CommonPrefixs	The same paths between Prefix and delimiter are grouped as the same type and defined as Common Prefix	Array	
Prefix	Indicates the prefix match, which is used to specify the prefix address of the returned object key	String	No
Upload	A collection of information about Upload	Array	
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	
UploadID	Indicates the ID of current multipart upload	String	
StorageClass	Indicates the storage class of uploaded parts. Enumerated values: Standard, Standard_IA	String	
Initiator	Indicates the information of the initiator of current upload. Child node includes UID	Object	

Parameter Name	Description	Type
UID	Developer's APPID	String
Owner	Indicates the information of the owner of these uploaded parts. Child node includes UID	Object
UID	Owner's QQ number	String
Initiated	Indicates the starting time for multipart upload	String

Slice Upload File

Feature description

This API (Slice Upload File) is used to upload a file in parts.

Method prototype

Call Slice Upload File:

```
var params = {
Bucket: 'STRING_VALUE', /* Required */
Region: 'STRING_VALUE', /* Required */
Key: 'STRING_VALUE', /* Required */
FilePath: 'STRING_VALUE', /* Required */
SliceSize: 'STRING_VALUE', /* Not required */
AsyncLimit: 'NUMBER_VALUE', /* Not required */
onHashProgress: function (progressData) {
console.log(JSON.stringify(progressData));
},
onProgress: function (progressData) {
console.log(JSON.stringify(progressData));
},
};

cos.sliceUploadFile(params, function(err, data) {
if(err) {
console.log(err);
} else {
console.log(data);
}
});
```

Operation parameter description

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes

Parameter Name	Description	Type	Required
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	Yes
FilePath	Path to the local file	String	Yes
SliceSize	Indicates the size of part	String	No
AsyncLimit	Indicates the number of concurrent parts	String	No
onHashProgress	Progress callback function to calculate a file's sha1 value. Callback is an object which includes progress information	Function	No
onProgress	Progress callback function. Callback is an object which includes progress information	Function	No

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
Location	Domain name for public network access of the created Object	String
Bucket	The target Bucket for multipart upload	String
Key	Object name	String
ETag	The SHA-1 algorithm check value of the file	String

Progress callback parameters

Parameter Name	Description	Type
SliceSize	Indicates the size of part	String
PartNumber	No. of parts uploaded successfully	Number

Parameter Name	Description	Type
FileSize	Total size of the file	Number

Slice Copy File

Feature description

This API (Slice Copy File) is used to copy a file from the source path to the destination path. In the process of copying, file meta-attributes and ACLs can be modified. You can use this API to move or rename a file, modify file attributes and create a copy.

Method prototype

Call Slice Copy File:

```
cos.sliceCopyFile({
Bucket: 'STRING_VALUE', /* Required */
Region: 'STRING_VALUE', /* Required */
Key: 'STRING_VALUE', /* Required */
CopySource: 'STRING_VALUE', /* Required */
SliceSize: 'NUMBER_VALUE', /* Not required */
onProgress:function (progressData) { /* Not required */
console.log(JSON.stringify(progressData));
}
},function (err,data) {
console.log(err || data);
});
```

Operation parameter description

Parameter Name	Description	Type	Required
Bucket	Bucket name. The bucket entered must be in a format of {name}-{appid}	String	Yes
Region	The region where the Bucket resides. For enumeration values, please see Bucket Region	String	Yes
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String	Yes
CopySource	The path of source file URL. You can specify the history version with the versionid sub-resource	String	Yes
ChunkSize	Indicates the size of each part for multipart copy, which defaults to 1048576 (1 MB)	Number	No
SliceSize	Indicates the file size for multipart copy, which defaults to 5 GB	Number	No

Parameter Name	Description	Type	Required
onProgress	Progress callback function. Callback is an object which includes progress information	Function	No

Callback function description

```
function(err, data) { ... }
```

Callback parameter description

Parameter Name	Description	Type
err	Indicates the object returned when a request fails, including network error and business error. If the request is successful, it is left empty. Error Code Document	Object
data	Indicates the object returned when a request is successful. If the request fails, it is left empty.	Object
Location	Domain name for public network access of the created Object	String
Bucket	The target Bucket for multipart upload	String
Key	Indicates the object key (object name), which is the unique identifier of the object in the bucket. Object key description	String
ETag	The MD5 algorithm check value of the file	String

PHP SDK
Getting Started

Last updated：2018-07-20 18:10:21

Preparations for Development

Obtain SDK

1. GitHub

```
#Obtain the code from GitHub
https://github.com/tencentyun/cos-php-sdk-v5
```

You can use the source code by simply placing it under your project directory.

2. Composer

Under the project directory, create a new composer.json file, as shown below:

```
#Download SDK using Composer
{
"require": {
"qcloud/cos-sdk-v5": ">=1.0"
}
}
```

Then install the SDK with the following command:

```
composer install
```

Getting Started

Refer to the Demo. For more information, please see GitHub.

Configure file

```
# Enter the relative path of the file cos-autoloader.php.
require(__DIR__ . DIRECTORY_SEPARATOR . 'cos-autoloader.php');

$cosClient = new Qcloud\Cos\Client(array('region' => getenv('COS_REGION'),
'credentials'=> array(
```

```
'secretId' => getenv('COS_KEY'),
'secretKey' => getenv('COS_SECRET')));
```

Upload file

- Upload the file (limited to 5 GB) using the API putObject.
- Upload the file in multiple parts using the API Upload.

```
# Upload file
## putObject (API for upload. File size is limited to 5 GB)
### Upload strings in memory
try {
$result = $cosClient->putObject(array(
'Bucket' => $bucket,
'Key' => $key,
'Body' => 'Hello World!'));
print_r($result);
} catch (\Exception $e) {
echo "$e\n";
}

### Upload file stream
try {
$result = $cosClient->putObject(array(
'Bucket' => $bucket,
'Key' => $key,
'Body' => fopen($local_path, 'rb')));
print_r($result);
} catch (\Exception $e) {
echo "$e\n";
}

### Set header and meta
try {
$result = $cosClient->putObject(array(
'Bucket' => $bucket,
'Key' => $key,
'Body' => fopen($local_path, 'rb'),
'ACL' => 'string',
'CacheControl' => 'string',
'ContentDisposition' => 'string',
'ContentEncoding' => 'string',
'ContentLanguage' => 'string',
'ContentLength' => integer,
'ContentType' => 'string',
'Expires' => 'mixed type: string (date format)|int (unix timestamp)|\DateTime',
'GrantFullControl' => 'string',
'GrantRead' => 'string',
'GrantWrite' => 'string',
```

```
'Metadata' => array(
'string' => 'string',
),
'StorageClass' => 'string'));
print_r($result);
} catch (\Exception $e) {
echo "$e\n";
}

## Upload (Advanced API for upload. Multipart upload is used by default. File size is limited to 50 TB)
### Upload strings in memory
try {
$result = $cosClient->Upload(
$bucket = $bucket,
$key = $key,
$body = 'Hello World!');
print_r($result);
} catch (\Exception $e) {
echo "$e\n";
}

### Upload file stream
try {
$result = $cosClient->Upload(
$bucket = $bucket,
$key = $key,
$body = fopen($local_path, 'rb'));
print_r($result);
} catch (\Exception $e) {
echo "$e\n";
}

### Set header and meta
try {
$result = $cosClient->upload(
$bucket= $bucket,
$key = $key,
$body = fopen($local_path, 'rb'),
$options = array(
'ACL' => 'string',
'CacheControl' => 'string',
'ContentDisposition' => 'string',
'ContentEncoding' => 'string',
'ContentLanguage' => 'string',
'ContentLength' => integer,
'ContentType' => 'string',
'Expires' => 'mixed type: string (date format)|int (unix timestamp)|\DateTime',
'GrantFullControl' => 'string',
'GrantRead' => 'string',
'GrantWrite' => 'string',
'Metadata' => array(
```

```
'string' => 'string',
),
'StorageClass' => 'string'));
print_r($result);
} catch (\Exception $e) {
echo "$e\n";
}
```

Download file

- Download the file using the API getObject.
- Get the file download URL using the API getObjectUrl.

```
# Download File
## getObject (Download file)
### Download the file to memory
try {
$result = $cosClient->getObject(array(
'Bucket' => $bucket,
'Key' => $key));
echo($result['Body']);
} catch (\Exception $e) {
echo "$e\n";
}

### Download the file locally
try {
$result = $cosClient->getObject(array(
'Bucket' => $bucket,
'Key' => $key,
'SaveAs' => $local_path));
} catch (\Exception $e) {
echo "$e\n";
}

### Specify the range of file download
/*
* Range format: 'bytes=a-b'
*/
try {
$result = $cosClient->getObject(array(
'Bucket' => $bucket,
'Key' => $key,
'Range' => 'bytes=0-10',
'SaveAs' => $local_path));
} catch (\Exception $e) {
echo "$e\n";
}
```

```
### Set response header
try {
$result = $cosClient->getObject(array(
'Bucket' => $bucket,
'Key' => $key,
'ResponseCacheControl' => 'string',
'ResponseContentDisposition' => 'string',
'ResponseContentEncoding' => 'string',
'ResponseContentLanguage' => 'string',
'ResponseContentType' => 'string',
'ResponseExpires' => 'mixed type: string (date format)|int (unix timestamp)|\DateTime',
'SaveAs' => $local_path));
} catch (\Exception $e) {
echo "$e\n";
}

## getObjectUrl (Get the file URL)
try {
$url = "/{$key}";
$request = $cosClient->get($url);
$signedUrl = $cosClient->getObjectUrl($bucket, $key, '+10 minutes');
echo ($signedUrl);

} catch (\Exception $e) {
echo "$e\n";
}
```

API Documentation

Last updated：2018-09-26 11:12:39

Basic APIs

For more information on the definitions of SecretId, SecretKey, Bucket and other terms and how to obtain them, please see COS Glossary.

Obtain Bucket list

Method prototype

```
public Guzzle\Service\Resource\Model listBucket(array $args = array())
```

Request example

```
//Obtain bucket list
$result = $cosClient->listBuckets();
```

Example of returned result

```
Array
(
[data:protected] => Array
(
[Owner] => Array
(
[ID] => qcs::cam::uin/3210232098:uin/3210232098
[DisplayName] => 3210232098
)

[Buckets] => Array
(
[0] => Array
(
[Name] => accesslog-10055004
[Location] => ap-shanghai
[CreationDate] => 2016-07-29T03:09:54Z
)

[1] => Array
(
[Name] => accesslogbj-10055004
[Location] => ap-beijing
```

```
[CreationDate] => 2017-08-02T04:00:24Z
)

)

[RequestId] => NWE3YzgxZmFfYWZhYzM1MGFfMzc3MF9iOGY5OQ==
)

)
```

Create Bucket

Method prototype

```
// Create a bucket
public Guzzle\Service\Resource\Model createBucket(array $args = array());
```

Request parameters

$args is an associative array containing the following fields:

Parameter Name	Description	Type	Required
Bucket	Bucket name	string	Yes
Acl	ACL permission control	string	No

Request example

```
//Create a bucket
//The bucket name entered must be in a format of {name}-{appid}
$result = $cosClient->createBucket(array('Bucket' => 'testbucket-125000000'));
```

Example of returned result

```
Array
(
[data:protected] => Array
(
[Location] =>
[RequestId] => NWE3YzgzMTZfMTdiMjk0MGFfNTQ1OF8xNjEyYmE=
)
)
```

Delete Bucket

Method prototype

```
// Delete a bucket
public Guzzle\Service\Resource\Model deleteBucket(array $args = array());
```

Request parameters

$args is an associative array containing the following fields:

Parameter Name	Description	Type	Required
Bucket	Bucket name	string	Yes

Request Example

```
//Delete a bucket
//The bucket name entered must be in a format of {name}-{appid}
$result = $cosClient->deleteBucket(array('Bucket' => 'testbucket-125000000'));
```

Response Example

```
Array
(
[data:protected] => Array
(
[RequestId] => NWE3YzgzMTZfMTdiMjk0MGFfNTQ2MF8xNjBjZTI=
)
)
```

Simple upload of file

Method prototype

```
public Guzzle\Service\Resource\Model putObject(array $args = array())
```

Request parameters

$args is an associative array containing the following fields:

Parameter Name	Description	Type	Required
Bucket	Bucket name, which is composed of numbers, lowercase letters and "-".	string	Yes
Body	The content of the uploaded file, which can be a file stream or a byte stream	file/string	Yes

Parameter Name	Description	Type	Required
Key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg, the object key is doc1/pic1.jpg.	string	Yes
ACL	Sets file ACL, such as 'private', 'public-read', and 'public-read-write'	string	No
GrantFullControl	Grants the specified account the permission to read and write files	string	No
GrantRead	Grants the specified account the permission to read files	string	No
GrantWrite	Grants the specified account the permission to write files	string	No
StorageClass	Sets file storage type: STANDARD and STANDARD_IA. Default: STANDARD	String	No
Expires	Sets Content-Expires	string	No
CacheControl	Cache policy. Sets Cache-Control	string	No
ContentType	Content type. Sets Content-Type	string	No
ContentDisposition	File name. Sets Content-Disposition	string	No
ContentEncoding	Encoding format. Sets Content-Encoding	string	No
ContentLanguage	Language type. Sets Content-Language	string	No
ContentLength	Sets transmission length	string	No
ContentMD5	Sets MD5 of the uploaded file for verification	string	No
Metadata	User-defined file meta information	array	No
ServerSideEncryption	Server-side encryption method	string	No

Request example

```
# Upload a file
## putObject (API for upload. File size is limited to 5 GB)
### Upload strings in memory
try {
$result = $cosClient->putObject(array(
'Bucket' => $bucket,
'Key' => $key,
'Body' => 'Hello World!'));
print_r($result);
} catch (\Exception $e) {
```

```
echo "$e\n";
}

### Upload file stream
try {
$result = $cosClient->putObject(array(
'Bucket' => $bucket,
'Key' => $key,
'Body' => fopen($local_path, 'rb')));
print_r($result);
} catch (\Exception $e) {
echo "$e\n";
}

### Set header and meta
try {
$result = $cosClient->putObject(array(
'Bucket' => $bucket,
'Key' => $key,
'Body' => fopen($local_path, 'rb'),
'ACL' => 'string',
'CacheControl' => 'string',
'ContentDisposition' => 'string',
'ContentEncoding' => 'string',
'ContentLanguage' => 'string',
'ContentLength' => integer,
'ContentType' => 'string',
'Expires' => 'mixed type: string (date format)|int (unix timestamp)|\DateTime',
'GrantFullControl' => 'string',
'GrantRead' => 'string',
'GrantWrite' => 'string',
'Metadata' => array(
'string' => 'string',
),
'StorageClass' => 'string'));
print_r($result);
} catch (\Exception $e) {
echo "$e\n";
}
```

Response Example

```
Array
(
[data:protected] => Array
(
[Expiration] =>
[ETag] => "ed076287532e86365e841e92bfc50d8c"
[ServerSideEncryption] => AES256
[VersionId] =>
```

```
[SSECustomerAlgorithm] =>
[SSECustomerKeyMD5] =>
[SSEKMSKeyId] =>
[RequestCharged] =>
[RequestId] => NWE3Yzg0M2NfOTcyMjViNjRfYTE1YV8xNTQzYTY=
[ObjectURL] => http://testbucket-1252448703.cos.cn-south.myqcloud.com/11//32//43
)
)
```

Multipart upload

This API is used to split a file (limited to 40 TB) into multiple parts for upload. Concurrent upload of multiple parts is allowed.

The steps of multipart upload are as follows:

1. Initialize multipart upload, and obtain uploadid. (createMultipartUpload)
2. Upload data in parts concurrently. (uploadPart)
3. Complete multipart upload. (completeMultipartUpload)

The steps of multipart upload also include obtaining uploaded parts (listParts) and terminating multipart upload (abortMultipartUpload).

Method prototype

```
// Initialize multipart upload
public Guzzle\Service\Resource\Model createMultipartUpload(array $args = array());

// Upload data in parts
public Guzzle\Service\Resource\Model uploadPart(array $args = array());

// Complete multipart upload
public Guzzle\Service\Resource\Model completeMultipartUpload(array $args = array());

// List uploaded parts
public Guzzle\Service\Resource\Model listParts(array $args = array());

// Abort multipart upload
public Guzzle\Service\Resource\Model abortMultipartUpload(array $args = array());
```

Upload files

Request example

```
## Upload (Advanced API for upload. Multipart upload is used by default. File size is limited to 50 TB)
### Upload strings in memory
try {
$result = $cosClient->Upload(
$bucket = $bucket,
```

```
$key = $key,
$body = 'Hello World!');
print_r($result);
} catch (\Exception $e) {
echo "$e\n";
}

### Upload file stream
try {
$result = $cosClient->Upload(
$bucket = $bucket,
$key = $key,
$body = fopen($local_path, 'rb'));
print_r($result);
} catch (\Exception $e) {
echo "$e\n";
}

### Set header and meta
try {
$result = $cosClient->upload(
$bucket= $bucket,
$key = $key,
$body = fopen($local_path, 'rb'),
$options = array(
'ACL' => 'string',
'CacheControl' => 'string',
'ContentDisposition' => 'string',
'ContentEncoding' => 'string',
'ContentLanguage' => 'string',
'ContentLength' => integer,
'ContentType' => 'string',
'Expires' => 'mixed type: string (date format)|int (unix timestamp)|\DateTime',
'GrantFullControl' => 'string',
'GrantRead' => 'string',
'GrantWrite' => 'string',
'Metadata' => array(
'string' => 'string',
),
'StorageClass' => 'string'));
print_r($result);
} catch (\Exception $e) {
echo "$e\n";
}
```

For a file smaller than 5 MB, use single upload.

Otherwise, use multipart upload.

Response Example

```
Array
(
[data:protected] => Array
(
[Location] => testbucket-1252448703.cos.cn-south.myqcloud.com/111.txt
[Bucket] => testbucket
[Key] => 111.txt
[ETag] => "715691804ee474f2eb94adb2c5c01155-1"
[Expiration] =>
[ServerSideEncryption] => AES256
[VersionId] =>
[SSEKMSKeyId] =>
[RequestCharged] =>
[RequestId] => NWE3Yzg0YTRfOTUyMjViNjRfNWYyZF8xNTI5ZDQ=
)
)
```

Download files

This API is used to download files locally or to memory.

Method prototype

```
// Download a file
public Guzzle\Service\Resource\Model getObject(array $args = array());
```

Request parameters

$args is an associative array containing the following fields:

Parameter Name	Description	Type	Required
Bucket	Bucket name	string	Yes
Key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg, the object key is doc1/pic1.jpg.	string	Yes
SaveAs	Local file path	string	No
VersionId	Version number of Object	string	No

Request Example

```
# Download a file
## getObject (Download file)
### Download the file to memory
try {
$result = $cosClient->getObject(array(
```

```
'Bucket' => $bucket,
'Key' => $key));
echo($result['Body']);
} catch (\Exception $e) {
echo "$e\n";
}

### Download the file locally
try {
$result = $cosClient->getObject(array(
'Bucket' => $bucket,
'Key' => $key,
'SaveAs' => $local_path));
} catch (\Exception $e) {
echo "$e\n";
}

### Specify the range of file download
/*
* Range format: 'bytes=a-b'
*/
try {
$result = $cosClient->getObject(array(
'Bucket' => $bucket,
'Key' => $key,
'Range' => 'bytes=0-10',
'SaveAs' => $local_path));
} catch (\Exception $e) {
echo "$e\n";
}

### Set response header
try {
$result = $cosClient->getObject(array(
'Bucket' => $bucket,
'Key' => $key,
'ResponseCacheControl' => 'string',
'ResponseContentDisposition' => 'string',
'ResponseContentEncoding' => 'string',
'ResponseContentLanguage' => 'string',
'ResponseContentType' => 'string',
'ResponseExpires' => 'mixed type: string (date format)|int (unix timestamp)|\DateTime',
'SaveAs' => $local_path));
} catch (\Exception $e) {
echo "$e\n";
}
```

Response Example

```
Array
(
[data:protected] => Array
(
[Body] =>
[DeleteMarker] =>
[AcceptRanges] => bytes
[Expiration] =>
[Restore] =>
[LastModified] => Fri, 09 Feb 2018 01:10:56 GMT
[ContentLength] => 5242880
[ETag] => "715691804ee474f2eb94adb2c5c01155-1"
[MissingMeta] =>
[VersionId] =>
[CacheControl] => private
[ContentDisposition] => attachment; filename*="UTF-8''111.txt"
[ContentEncoding] =>
[ContentLanguage] =>
[ContentRange] =>
[ContentType] => text/plain; charset=utf-8
[Expires] =>
[WebsiteRedirectLocation] =>
[ServerSideEncryption] => AES256
[SSECustomerAlgorithm] =>
[SSECustomerKeyMD5] =>
[SSEKMSKeyId] =>
[StorageClass] =>
[RequestCharged] =>
[ReplicationStatus] =>
[RequestId] => NWE3Yzg4ODlfMThiMjk0MGFfMmI3OV8xNWQxNDg=
)
)
```

Deleting Files

This API is used to delete objects on COS.

Method prototype

```
// Delete a file
public Guzzle\Service\Resource\Model deleteObject(array $args = array());
```

Request parameters

$args is an associate array containing the following fields:

Parameter Name	Description	Type	Required

Parameter Name	Description	Type	Required
Bucket	Bucket name	string	Yes
Key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg, the object key is doc1/pic1.jpg.	string	Yes
VersionId	Version number of Object	string	No

Request example

```
// Delete an object on COS
$result = $cosClient->deleteObject(array(
//The bucket name entered must be in a format of {name}-{appid}
'Bucket' => 'testbucket-125000000',
'Key' => 'hello.txt'));
```

Response Example

```
Array
(
[data:protected] => Array
(
[DeleteMarker] =>
[VersionId] =>
[RequestCharged] =>
[RequestId] => NWE3Yzg5MzJfY2FhMzNiMGFfNDVjOV8yY2QyMzg=
)
)
```

Obtaining object attributes

Obtain the attributes of an object on COS.

Method prototype

```
// Obtain file attributes
public Guzzle\Service\Resource\Model headObject(array $args = array());
```

Request parameters

$args is an associative array containing the following fields:

Parameter Name	Description	Type	Required
Bucket	Bucket name	string	Yes

Parameter Name	Description	Type	Required
Key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg, the object key is doc1/pic1.jpg.	string	Yes
VersionId	Version number of Object	string	No

Response Example

```
Array
(
[data:protected] => Array
(
[DeleteMarker] =>
[AcceptRanges] =>
[Expiration] =>
[Restore] =>
[LastModified] => Thu, 08 Feb 2018 17:34:53 GMT
[ContentLength] => 12
[ETag] => "ed076287532e86365e841e92bfc50d8c"
[MissingMeta] =>
[VersionId] =>
[CacheControl] =>
[ContentDisposition] =>
[ContentEncoding] =>
[ContentLanguage] =>
[ContentType] => application/octet-stream
[Expires] =>
[WebsiteRedirectLocation] =>
[ServerSideEncryption] => AES256
[SSECustomerAlgorithm] =>
[SSECustomerKeyMD5] =>
[SSEKMSKeyId] =>
[StorageClass] =>
[RequestCharged] =>
[ReplicationStatus] =>
[RequestId] => NWE3YzhhM2RfMWViZTk0MGFfNWMzMF8xNTFiZDg=
)
)
```

Request example

```
// Obtain the attributes of a file on COS
//The bucket name entered must be in a format of {name}-{appid}
$result $cosClient->headObject(array('Bucket' => 'testbucket-125000000', 'Key' => 'hello.txt'));
```

Check whether a Bucket exists

Check whether a bucket on COS exists.

Method prototype

```
// Obtain file attributes
public Guzzle\Service\Resource\Model headBucket(array $args = array());
```

Request parameters

$args is an associative array containing the following fields:

Parameter Name	Description	Type	Required
Bucket	Version number of Object	string	No

Request example

```
// Obtain the attributes of a file on COS
//The bucket name entered must be in a format of {name}-{appid}
$result $cosClient->headBucket(array('Bucket' => 'testbucket-125000000'));
```

Response Example

```
Array
(
[data:protected] => Array
(
[RequestId] => NWE3YzhhN2VfY2VhMzNiMGFfMmNmXzJjNzc3Zg==
)
)
```

Obtaining file list

Obtain the list of files on COS.

Method prototype

```
// Obtain a file list
public Guzzle\Service\Resource\Model listObjects(array $args = array());
```

Request parameters

$args is an associative array containing the following fields:

Parameter Name	Description	Type	Required
Bucket	Bucket name	string	Yes

Parameter Name	Description	Type	Required
Delimiter	Delimiter	string	No
Marker	Marker	string	No
MaxKeys	Maximum number of objects	int	No
Prefix	Prefix	string	No

Request example

```
// Obtain the bucket members
//The bucket name entered must be in a format of {name}-{appid}
$result = $cosClient->listObjects(array('Bucket' => 'testbucket-125000000'));
```

Example of returned result

```
Array
(
[data:protected] => Array
(
[Name] => testbucket-1252448703
[Prefix] =>
[Marker] =>
[MaxKeys] => 1000
[IsTruncated] =>
[Contents] => Array
(
[0] => Array
(
[Key] => 11/32/43
[LastModified] => 2018-02-08T17:09:16.000Z
[ETag] => "ed076287532e86365e841e92bfc50d8c"
[Size] => 12
[Owner] => Array
(
[ID] => 1252448703
[DisplayName] => 1252448703
)

[StorageClass] => STANDARD
)

[1] => Array
(
[Key] => 111
[LastModified] => 2018-02-08T17:41:11.000Z
[ETag] => "ed076287532e86365e841e92bfc50d8c"
```

```
[Size] => 12
[Owner] => Array
(
[ID] => 1252448703
[DisplayName] => 1252448703
)

[StorageClass] => STANDARD
)

[2] => Array
(
[Key] => 111.txt
[LastModified] => 2018-02-08T17:11:00.000Z
[ETag] => "715691804ee474f2eb94adb2c5c01155-1"
[Size] => 5242880
[Owner] => Array
(
[ID] => 1252448703
[DisplayName] => 1252448703
)

[StorageClass] => STANDARD
)

)

[RequestId] => NWE3YzhiYjdfMWJiMjk0MGFfMzA4M18xNjdiNDM=
)
)
```

putBucketACL

Method prototype

```
// Obtain a file list
public Guzzle\Service\Resource\Model putBucketACL(array $args = array());
```

Request parameters

$args is an associative array containing the following fields:

Parameter Name	Description	Type	Required
Bucket	Bucket name	string	Yes
ACL	ACL permission control	string	No

Parameter Name	Description	Type	Required
GrantRead	Grants read permission to the authorized user. Format: id=" ", id=" ". For authorization to a subaccount, id="qcs::cam::uin/:uin/"; for authorization to the root account, id="qcs::cam::uin/:uin/". For example: 'id="qcs::cam::uin/123:uin/123", id="qcs::cam::uin/123:uin/456"'	string	No
GrantWrite	Grants write permission to the authorized user. Format: id=" ", id=" ". For authorization to a subaccount, id="qcs::cam::uin/:uin/"; for authorization to the root account, id="qcs::cam::uin/:uin/". For example: 'id="qcs::cam::uin/123:uin/123", id="qcs::cam::uin/123:uin/456"'	string	No
GrantFullControl	Grants read and write permissions to the authorized user. Format: id=" ", id=" ". For authorization to a subaccount, id="qcs::cam::uin/:uin/"; for authorization to the root account, id="qcs::cam::uin/:uin/". For example: 'id="qcs::cam::uin/123:uin/123", id="qcs::cam::uin/123:uin/456"'	string	No

Request example

```
#putBucketACL
try {
$result = $cosClient->PutBucketAcl(array(
//The bucket name entered must be in a format of {name}-{appid}
'Bucket' => 'testbucket-125000000',
'Grants' => array(
array(
'Grantee' => array(
'DisplayName' => 'qcs::cam::uin/327874225:uin/327874225',
'ID' => 'qcs::cam::uin/327874225:uin/327874225',
'Type' => 'CanonicalUser',
),
'Permission' => 'FULL_CONTROL',
),
// ... repeated
),
'Owner' => array(
'DisplayName' => 'qcs::cam::uin/3210232098:uin/3210232098',
'ID' => 'qcs::cam::uin/3210232098:uin/3210232098',
),));
print_r($result);
} catch (\Exception $e) {
echo "$e\n";
}
```

Response Example

```
Array
(
[data:protected] => Array
(
[RequestId] => NWE3YzhiZTZfZDRiMjk0MGFfODMwXzJjODllYw==
)
)
```

getBucketACL

Method prototype

```
// Obtain a file list
public Guzzle\Service\Resource\Model getBucketACL(array $args = array());
```

Request parameters

Field Name	Type	Default	Required	Description
Bucket	string	None	Yes	Bucket name

Request example

```
#getBucketACL
try {
$result = $cosClient->GetBucketAcl(array(
//The bucket name entered must be in a format of {name}-{appid}
'Bucket' => 'testbucket-125000000',));
print_r($result);
} catch (\Exception $e) {
echo "$e\n";
}
```

Response Example

```
Array
(
[data:protected] => Array
(
[Owner] => Array
(
[ID] => qcs::cam::uin/3210232098:uin/3210232098
[DisplayName] => qcs::cam::uin/3210232098:uin/3210232098
)

[Grants] => Array
(
```

```
[0] => Array
(
[Grantee] => Array
(
[ID] => qcs::cam::uin/327874225:uin/327874225
[DisplayName] => qcs::cam::uin/327874225:uin/327874225
)

[Permission] => FULL_CONTROL
)

)

[RequestId] => NWE3YzhjMTRfYzdhMzNiMGFfYjdiOF8yYzZmMzU=
)
)
```

putObjectACL

Method prototype

```
// Obtain a file list
public Guzzle\Service\Resource\Model putObjectACL(array $args = array());
```

Request parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name	string	Yes
Key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg, the object key is doc1/pic1.jpg.	string	Yes
ACL	ACL permission control	string	No
GrantRead	Grants read permission to the authorized user. Format: id=" ", id=" ". For authorization to a subaccount, id="qcs::cam::uin/:uin/"; for authorization to the root account, id="qcs::cam::uin/:uin/". For example: 'id="qcs::cam::uin/123:uin/123", id="qcs::cam::uin/123:uin/456"'	string	No
GrantWrite	Grants write permission to the authorized user. Format: id=" ", id=" ". For authorization to a subaccount, id="qcs::cam::uin/:uin/"; for authorization to the root account, id="qcs::cam::uin/:uin/". For example: 'id="qcs::cam::uin/123:uin/123", id="qcs::cam::uin/123:uin/456"'	string	No

Parameter Name	Description	Type	Required
GrantFullControl	Grants read and write permissions to the authorized user. Format: id="", id="". For authorization to a subaccount, id="qcs::cam::uin/:uin/"; for authorization to the root account, id="qcs::cam::uin/:uin/". For example: 'id="qcs::cam::uin/123:uin/123", id="qcs::cam::uin/123:uin/456"'	string	No

Request example

```
#putObjectACL
try {
$result = $cosClient->PutObjectAcl(array(
'Bucket' => 'testbucket-1252448703',
'Key' => '111.txt',
'Grants' => array(
array(
'Grantee' => array(
'DisplayName' => 'qcs::cam::uin/327874225:uin/327874225',
'ID' => 'qcs::cam::uin/327874225:uin/327874225',
'Type' => 'CanonicalUser',
),
'Permission' => 'FULL_CONTROL',
),
// ... repeated
),
'Owner' => array(
'DisplayName' => 'qcs::cam::uin/3210232098:uin/3210232098',
'ID' => 'qcs::cam::uin/3210232098:uin/3210232098',
),));
print_r($result);
} catch (\Exception $e) {
echo "$e\n";
}
```

Response Example

```
Array
(
[data:protected] => Array
(
[RequestCharged] =>
[RequestId] => NWE3YzhjZDdfY2JhMzNiMGFfNjVhOV8yZDJhNjY=
)
)
```

getObjectACL

Method prototype

```
// Obtain a file list
public Guzzle\Service\Resource\Model getObjectACL(array $args = array());
```

Request parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name	string	Yes
Key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg, the object key is doc1/pic1.jpg.	string	Yes

Request example

```
#getObjectACL
try {
$result = $cosClient->getObjectAcl(array(
//The bucket name entered must be in a format of {name}-{appid}
'Bucket' => 'testbucket-125000000',
'Key' => '11'));
print_r($result);
} catch (\Exception $e) {
echo "$e\n";
}
```

Example of returned result

```
Array
(
[data:protected] => Array
(
[Owner] => Array
(
[ID] => qcs::cam::uin/3210232098:uin/3210232098
[DisplayName] => qcs::cam::uin/3210232098:uin/3210232098
)

[Grants] => Array
(
[0] => Array
(
[Grantee] => Array
(
[ID] => qcs::cam::uin/327874225:uin/327874225
[DisplayName] => qcs::cam::uin/327874225:uin/327874225
```

```
)

[Permission] => FULL_CONTROL
)

)

[RequestCharged] =>
[RequestId] => NWE3YzhjZDdfY2JhMzNiMGFfNjU5OF8yYzlkMmE=
)
)
```

putBucketCors

Method prototype

```
// Obtain a file list
public Guzzle\Service\Resource\Model putBucketCors(array $args = array());
```

Request parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name	string	Yes
CORSRules	CORS rules	array	Yes
AllowedMethods	Allowed HTTP operations. Enumerated values: GET, PUT, HEAD, POST, DELETE.	array	No
AllowedOrigins	Allowed access sources. The wildcard "*" is supported. Format: protocol://domain_name[:port], for example, http://www.qq.com	array	No
AllowedHeaders	When an OPTIONS request is sent, notifies the server about which custom HTTP request headers are allowed for subsequent requests. Wildcard "*" is supported.	array	No
ExposeHeaders	Sets the custom header information that can be received by the browser from the server end.	array	No
MaxAgeSeconds	Sets the validity period of the results obtained by OPTIONS	string	No
ID	Sets rule ID	string	No

Request example

```
###putBucketCors
try {
$result = $cosClient->putBucketCors(array(
//The bucket name entered must be in a format of {name}-{appid}
```

```
'Bucket' => 'testbucket-125000000',
// CORSRules is required
'CORSRules' => array(
array(
'AllowedHeaders' => array('*',),
// AllowedMethods is required
'AllowedMethods' => array('Put', ),
// AllowedOrigins is required
'AllowedOrigins' => array('*', ),
'ExposeHeaders' => array('*', ),
'MaxAgeSeconds' => 1,
),
// ... repeated
),
));
print_r($result);
} catch (\Exception $e) {
echo "$e\n";
}
```

Example of returned result

```
Array
(
[data:protected] => Array
(
[RequestCharged] =>
[RequestId] => NWE3YzhjZDdfY2JhMzNiMGFfNjVhOV8yZDJhNjY=
)
)
```

getBucketCors

Method prototype

```
// Obtain a file list
public Guzzle\Service\Resource\Model getBucketCors(array $args = array());
```

Request parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name	string	Yes

Request example

```
#getBucketCors
try {
```

```
$result = $cosClient->getBucketCors(array(
//The bucket name entered must be in a format of {name}-{appid}
'Bucket' => 'testbucket-125000000',
));
print_r($result);
} catch (\Exception $e) {
echo "$e\n";
}
```

Response Example

```
Array
(
[data:protected] => Array
(
[CORSRules] => Array
(
[0] => Array
(
[ID] => 1234
[AllowedHeaders] => Array
(
[0] => *
)

[AllowedMethods] => Array
(
[0] => PUT
)

[AllowedOrigins] => Array
(
[0] => http://www.qq.com
)

)

)

[RequestId] => NWE3YzhkMmRfMTdiMjk0MGFfNTQzZl8xNWUwMGU=
)
)
```

deleteBucketCors

Method prototype

```
// Obtain a file list
public Guzzle\Service\Resource\Model deleteBucketCors(array $args = array());
```

Request parameters

- **params** (Object): Parameter list
  - Bucket - (String): Bucket name

Request example

```
#deleteBucketCors
try {
$result = $cosClient->deleteBucketCors(array(
//The bucket name entered must be in a format of {name}-{appid}
'Bucket' => 'testbucket-125000000',
));
print_r($result);
} catch (\Exception $e) {
echo "$e\n";
}
```

Example of returned result

```
Array
(
[data:protected] => Array
(
[RequestCharged] =>
[RequestId] => NWE3YzhjZDdfY2JhMzNiMGFfNjVhOV8yZDJhNjY=
)
)
```

Copy objects

Method prototype

```
// Obtain a file list
public Guzzle\Service\Resource\Model copyObject(array $args = array());
```

Request parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name, which is composed of numbers, lowercase letters and "-".	string	Yes
CopySource	Copies the source object	string	Yes

Parameter Name	Description	Type	Required
Key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg, the object key is doc1/pic1.jpg.	string	Yes
ACL	Sets file ACL, such as 'private', 'public-read', and 'public-read-write'	string	No
GrantFullControl	Grants the specified account the permission to read and write files	string	No
GrantRead	Grants the specified account the permission to read files	string	No
GrantWrite	Grants the specified account the permission to write files	string	No
StorageClass	Sets file storage type: STANDARD and STANDARD_IA. Default: STANDARD	String	No
Expires	Sets Content-Expires	string	No
CacheControl	Cache policy. Sets Cache-Control	string	No
ContentType	Content type. Sets Content-Type	string	No
ContentDisposition	File name. Sets Content-Disposition	string	No
ContentEncoding	Encoding format. Sets Content-Encoding	string	No
ContentLanguage	Language type. Sets Content-Language	string	No
ContentLength	Sets transmission length	string	No
ContentMD5	Sets MD5 of the uploaded file for verification	string	No
Metadata	User-defined file meta information	array	No
ServerSideEncryption	Server-side encryption method	string	No

Request example

```
#copyobject
# API for simple copy
try {
$result = $cosClient->copyObject(array(
//The bucket name entered must be in a format of {name}-{appid}
'Bucket' => 'testbucket-125000000',
// CopySource is required
'CopySource' => 'lewzylu03-1252448703.cos.ap-guangzhou.myqcloud.com/tox.ini',
// Key is required
'Key' => 'string',
```

```
));
print_r($result);
} catch (\Exception $e) {
echo "$e\n";
}

#copy
#For files larger than 5 GB, multipart copy is used by default.
try {
$result = $cosClient->Copy($bucket = 'lewzylu01-1252448703',
$key = 'string',
$copysource = 'lewzylu02-1252448703.cos.ap-guangzhou.myqcloud.com/test1G',
$options = array('VersionId'=>'MTg0NDY3NDI1NTk0MzUwNDQ1OTg'));
print_r($result);
} catch (\Exception $e) {
echo "$e\n";
}
```

putBucketLifecycle

Method prototype

```
// Obtain a file list
public Guzzle\Service\Resource\Model putBucketLifecycle(array $args = array());
```

Request parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name, which is composed of numbers, lowercase letters and "-".	string	Yes
Rules	Sets the appropriate rules, including ID, Filter, Status, Expiration, Transition, NoncurrentVersionExpiration, NoncurrentVersionTransition, AbortIncompleteMultipartUpload	array	Yes
ID	Sets rule ID	string	No
Filter	Describes a collection of Objects that are subject to the rules.	array	Yes
Status	Sets whether Rule is enabled. Available values: Enabled or Disabled	string	Yes
Expiration	Sets the expiration rule for Object. You can specify the number of days (Days) or a date (Date).	array	No
Transition	Sets the rule for changing the storage type of Object. You can specify the number of days (Days) or a date (Date). Available value for StorageClass: Standard_IA.	array	No

Request example

```
#putBucketLifecycle
try {
$result = $cosClient->putBucketLifecycle(array(
//The bucket name entered must be in a format of {name}-{appid}
'Bucket' => 'testbucket-125000000',
// Rules is required
'Rules' => array(
array(
'Expiration' => array(
'Days' => 1,
),
'ID' => 'id1',
'Filter' => array(
'Prefix' => 'documents/'
),
// Status is required
'Status' => 'Enabled',
'Transitions' => array(
array(
'Days' => 200,
'StorageClass' => 'Standard_IA')),
// ... repeated
),
)));
print_r($result);
} catch (\Exception $e) {
echo "$e\n";
}
```

Example of returned result

```
Array
(
[data:protected] => Array
(
[RequestCharged] =>
[RequestId] => NWE3YzhjZDdfY2JhMzNiMGFfNjVhOV8yZDJhNjY=
)
)
```

getBucketLifecycle

Method prototype

```
// Obtain a file list
public Guzzle\Service\Resource\Model getBucketLifecycle(array $args = array());
```

Request parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name	string	Yes

Request example

```
#getBucketLifecycle
try {
$result = $cosClient->getBucketLifecycle(array(
//The bucket name entered must be in a format of {name}-{appid}
'Bucket' =>'testbucket-125000000',
));
print_r($result);
} catch (\Exception $e) {
echo "$e\n";
}
```

Example of returned result

```
Array
(
[data:protected] => Array
(
[Rules] => Array
(
[0] => Array
(
[ID] => id1
[Filter] => Array
(
[Prefix] => documents/
)

[Status] => Enabled
[Transition] => Array
(
[Days] => 200
[StorageClass] => Standard_IA
)

[Expiration] => Array
(
[Days] => 1000
)

)

)
```

```
[RequestId] => NWE3YzhlZjNfY2FhMzNiMGFfNDVkNF8yZDIxODE=
)
)
```

deleteBucketLifecycle

Method prototype

```
// Obtain a file list
public Guzzle\Service\Resource\Model deleteBucketLifecycle(array $args = array());
```

Request parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name	string	Yes

Request example

```
#deleteBucketLifecycle
try {
$result = $cosClient->deleteBucketLifecycle(array(
//The bucket name entered must be in a format of {name}-{appid}
'Bucket' =>'testbucket-125000000',
));
print_r($result);
} catch (\Exception $e) {
echo "$e\n";
}
```

Example of returned result

```
Array
(
[data:protected] => Array
(
[RequestCharged] =>
[RequestId] => NWE3YzhjZDdfY2JhMzNiMGFfNjVhOV8yZDJhNjY=
)
)
```

Obtain object download URL

Obtain a signed download URL of an object

Request example

```
//Obtain object download URL
//The bucket name entered must be in a format of {name}-{appid}
$bucket = 'testbucket-125000000';
$key = 'hello.txt';
$region = 'cn-south';
$url = "/{$key}";
$request = $cosClient->get($url);
$signedUrl = $cosClient->getObjectUrl($bucket, $key, '+10 minutes');
```

Use a temporary key

```
$cosClient = new Qcloud\Cos\Client(
array(
'region' => 'cn-south',
'timeout' => ,
'credentials'=> array(
'appId' => '',
'secretId' => '',
'secretKey' => '',
'token' => '')));
```

Restore an archived file

Method prototype

```
// Restore an archived file
public Guzzle\Service\Resource\Model deleteObject(array $args = array());
```

Request parameters

$args is an associative array containing the following fields:

Parameter Name	Description	Type	Required
Bucket	Bucket name	string	Yes
Key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg, the object key is doc1/pic1.jpg.	string	Yes
Days	Storage duration	integer	Yes
Tier	Restoration type	string	No

Request example

```
try {
$result = $cosClient->restoreObject(array(
// Bucket is required
'Bucket' => 'lewzylu02',
// Objects is required
'Key' => '11',
'Days' => 7,
'CASJobParameters' => array(
'Tier' =>'Bulk'
)
));
print_r($result);
} catch (\Exception $e) {
echo "$e\n";
}
```

Enable multiple versions

Method prototype

```
// Enable multiple versions
public Guzzle\Service\Resource\Model putBucketVersioning(array $args = array());
```

Request parameters

$args is an associate array containing the following fields:

Parameter Name	Description	Type	Required
Bucket	Bucket name	string	Yes
Status	Multi-version status	string	Yes

Request example

```
#putBucketVersioning
try {
$result = $cosClient->putBucketVersioning(
array('Bucket' => 'lewzylu02',
'Status' => 'Enabled')
);
print_r($result);
} catch (\Exception $e) {
echo "$e\n";
}
```

Example of returned result

```
Array
(
[data:protected] => Array
(
[RequestCharged] =>
[RequestId] => NWE3YzhjZDdfY2JhMzNiMGFfNjVhOV8yZDJhNjY=
)
)
```

Obtain bucket version

Method prototype

```
// Obtain bucket version
public Guzzle\Service\Resource\Model getBucketVersioning(array $args = array());
```

Request parameters

$args is an associative array containing the following fields:

Parameter Name	Description	Type	Required
Bucket	Bucket name	string	Yes

Request example

```
try {
$result = $cosClient->getBucketVersioning(
array('Bucket' => 'lewzylu02',)
);
print_r($result);
} catch (\Exception $e) {
echo "$e\n";
}
```

Example of returned result

```
Array
(
[data:protected] => Array
(
[Status] => Enabled
[RequestId] => NWE3YzhmZTVfNjIyNWI2NF80YzQ3XzJkNjU4NQ==
)
)
```

Print the file lists of different versions

Method prototype

```
// Print the file lists of different versions
public Guzzle\Service\Resource\Model listObjectVersions(array $args = array());
```

Request parameters

$args is an associative array containing the following fields:

Parameter Name	Description	Type	Required
Bucket	Bucket name	string	Yes
Delimiter	Delimiter	string	No
Marker	Marker	string	No
MaxKeys	Maximum number of objects	int	No
Prefix	Prefix	string	No

Request example

```
try {
$result = $cosClient->listObjectVersions(
array('Bucket' => 'lewzylu02',
'Prefix'=>'test1G')
);
print_r($result);
} catch (\Exception $e) {
echo "$e\n";
}
```

Example of returned result

```
Array
(
[data:protected] => Array
(
[Name] => lewzylu02-1252448703
[Prefix] => test1G
[KeyMarker] =>
[VersionIdMarker] =>
[MaxKeys] => 1000
[IsTruncated] =>
[Versions] => Array
(
[0] => Array
(
[Key] => test1G
```

```
[VersionId] => MTg0NDY3NDI1NTg1ODc4Nzk3NjI
[IsLatest] => 1
[LastModified] => 2018-01-05T03:07:51.000Z
[ETag] => "202cb962ac59075b964b07152d234b70"
[Size] => 3
[StorageClass] => STANDARD
[Owner] => Array
(
[UID] => 1252448703
)

)

[1] => Array
(
[Key] => test1G
[VersionId] => MTg0NDY3NDI1NTk0MzI3NDU3NTk
[IsLatest] =>
[LastModified] => 2017-12-26T08:26:50.000Z
[ETag] => "13ddf6552868644926ba606cd287106b-1"
[Size] => 5242880
[StorageClass] => STANDARD
[Owner] => Array
(
[UID] => 1252448703
)

)

[2] => Array
(
[Key] => test1G
[VersionId] => MTg0NDY3NDI1NTk0MzI3ODAzODc
[IsLatest] =>
[LastModified] => 2017-12-26T08:26:16.000Z
[ETag] => "3c86b7371340b2174b875fa7bcc0bd9a-1"
[Size] => 5242880
[StorageClass] => STANDARD
[Owner] => Array
(
[UID] => 1252448703
)

)
)

[RequestId] => NWE3YzkwMGFfMTliYjk0MGFfMWUwOWRfMmJlZWIx
)
)
```

Python SDK
Getting Started

Last updated：2018-08-14 17:09:11

Preparations for Development

Related resources

Download XML Python SDK resources of COS service from: XML Python SDK .
Download Demo from: XML Python Demo.

Environment dependencies

The XML Python SDK for COS service supports Python 2.6, Python 2.7 and Python 3.x.

> For more information on the definitions of SecretId, SecretKey, Bucket and other terms and how to obtain them, please see COS Glossary.

Install SDK

There are three ways to install the SDK: installation using pip, manual installation, and offline installation.

Installation using pip (recommended)

```
pip install -U cos-python-sdk-v5
```

Manual installation

Download the source code from XML Python SDK and install it manually via setup:

```
python setup.py install
```

Offline installation

```
# Run the following command on a server with public network
mkdir cos-python-sdk-packages
pip download cos-python-sdk-v5 -d cos-python-sdk-packages
tar -czvf cos-python-sdk-packages.tar.gz cos-python-sdk-packages

# Copy the installer package to a server without public network and run the following command
tar -xzvf cos-python-sdk-packages.tar.gz
pip install cos-python-sdk-v5 --no-index -f cos-python-sdk-packages
```

Getting Started

```
# appid has been removed from the configuration. Please include the appid in the parameter Bucket. Bucket is in the format of bucketname-appid.
# 1. Set user configuration, including secretId, secretKey and Region
# -*- coding=utf-8
from qcloud_cos import CosConfig
from qcloud_cos import CosS3Client
import sys
import logging

logging.basicConfig(level=logging.INFO, stream=sys.stdout)

secret_id = 'xxxxxxxx' # Replaced with user's secretId
secret_key = 'xxxxxxx' # Replaced with user's secretKey
region = 'ap-beijing-1' # Replaced with user's Region
token = None # Token is required to use a temporary key. It is optional. Default is empty.
config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=token, Scheme=scheme)
# 2. Obtain client object
client = CosS3Client(config)
# Refer to the description below or the Demo. For more information, please see https://github.com/tencentyun/cos-python-sdk-v5/blob/master/qcloud_cos/demo.py.
```

Uploading File

Simple upload of file stream

```
file_name = 'test.txt'
with open('test.txt', 'rb') as fp:
response = client.put_object(
Bucket='test04-123456789',
Body=fp,
Key=file_name,
StorageClass='STANDARD',
ContentType='text/html; charset=utf-8'
)
print(response['ETag'])
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name, in the format of bucketname-appid	String	Yes

Parameter Name	Description	Type	Required
Body	The content of the uploaded file, which can be a file stream or a byte stream	file/bytes	Yes
Key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name `bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg` , the object key is doc1/pic1.jpg.	String	Yes
StorageClass	Storage type of file: STANDARD and STANDARD_IA. Default: STANDARD	String	No
ContentType	Content Type. Set Content-Type	String	No

For more optional parameters, please see Python SDK Documentation.

Simple upload of byte stream

```
response = client.put_object(
Bucket='test04-123456789',
Body=b'abcdefg',
Key=file_name,
)
print(response['ETag'])
```

Simple upload from local path

```
response = client.put_object_from_local_file(
Bucket='test04-123456789',
LocalFilePath='local.txt',
Key=file_name,
)
print(response['ETag'])
```

Simple upload by setting HTTP header

```
response = client.put_object(
Bucket='test04-123456789',
Body=b'test',
Key=file_name,
ContentType='text/html; charset=utf-8'
)
print(response['ETag'])
```

Simple upload by setting custom header

```
response = client.put_object(
Bucket='test04-123456789',
Body=b'test',
Key=file_name,
Metadata={
'x-cos-meta-key1': 'value1',
'x-cos-meta-key2': 'value2'
}
)
print(response['ETag'])
```

Advanced API for upload (recommended)

Simple upload or multipart upload is selected automatically based on the file size. Multipart upload supports resuming download from breakpoint.

```
response = client.upload_file(
Bucket='test04-123456789',
LocalFilePath='local.txt',
Key=file_name,
PartSize=10,
MAXThread=10
)
print(response['ETag'])
```

Downloading File

Download the file locally

```
response = client.get_object(
Bucket='test04-123456789',
Key=file_name,
)
response['Body'].get_stream_to_file('output.txt')
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name, in the format of bucketname-appid	String	Yes
Key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg, the object key is doc1/pic1.jpg.	string	Yes

For more optional parameters, please see Python SDK Documentation.

Get file stream

```
response = client.get_object(
Bucket='test04-123456789',
Key=file_name,
)
fp = response['Body'].get_raw_stream()
print(fp.read(2))
```

Set Response HTTP header

```
response = client.get_object(
Bucket='test04-123456789',
Key=file_name,
ResponseContentType='text/html; charset=utf-8'
)
print response['Content-Type']
fp = response['Body'].get_raw_stream()
print(fp.read(2))
```

Specify the range of file download

```
response = client.get_object(
Bucket='test04-123456789',
Key=file_name,
Range='bytes=0-10'
)
fp = response['Body'].get_raw_stream()
print(fp.read())
```

Exception Types

Exceptions include CosClientError (SDK client error) and CosServiceError (COS server error).

CosClientError

CosClientError generally refers to a client error caused by the reasons such as timeout. When capturing such an error, you can choose to retry or perform other operations.

CosServiceError

CosServiceError provides the message returned by the server.

```
#except CosServiceError as e
e.get_origin_msg() # Get original error message in XML format
```

```
e.get_digest_msg() # Get the processed error message in dict format
e.get_status_code() # Get http error code (e.g. 4XX, 5XX)
e.get_error_code() # Get COS-defined error code
e.get_error_msg() # Get a detailed description of the COS error code
e.get_trace_id() # Get the trace_id of the request
e.get_request_id() # Get the request_id of the request
e.get_resource_location() # Get the URL address
```

API Documentation

Last updated：2018-09-26 11:16:52

For a successful operation of COS XML API Python SDK, dict or None is returned. For a failed operation, an exception (CosClientError and CosServiceError) is thrown. An error message is provided for the corresponding exception type. For more information, please see Exception Types at the end of this document.

> For more information on the definitions of SecretId, SecretKey, Bucket and other terms and how to obtain them, please see COS Glossary.

Bucket APIs

Create Bucket

Feature description

This API is used to create a Bucket under the specified account. An error is returned if a bucket exists.

Method prototype

```
create_bucket(Bucket, **kwargs)
```

Request example

```
response = client.create_bucket(
Bucket='test01-123456789',
ACL='private'|'public-read'|'public-read-write',
GrantFullControl='string',
GrantRead='string',
GrantWrite='string'
)
```

Parameters

Parameter Name	Description	Type	Required
Bucket	The name of the bucket to be created, in the format of bucketname-appid	String	Yes
ACL	Sets the bucket ACL, such as 'private', 'public-read', and 'public-read-write'	String	No

Parameter Name	Description	Type	Required
GrantFullControl	Grants the specified account the permission to read and write buckets in the format of `id=" ",id=" "` . For authorization to a sub-account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{SubUin}"` . For authorization to a root account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{OwnerUin}"` . For example, `'id="qcs::cam::uin/123:uin/456",id="qcs::cam::uin/123:uin/123"'`	String	No
GrantRead	Grants the specified account the permission to read buckets in the format of `id=" ",id=" "` . For authorization to a sub-account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{SubUin}"` . For authorization to a root account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{OwnerUin}"` . For example, `'id="qcs::cam::uin/123:uin/456",id="qcs::cam::uin/123:uin/123"'`	String	No
GrantWrite	Grants the specified account the permission to write buckets in the format of `id=" ",id=" "` . For authorization to a sub-account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{SubUin}"` . For authorization to a root account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{OwnerUin}"` . For example, `'id="qcs::cam::uin/123:uin/456",id="qcs::cam::uin/123:uin/123"'`	String	No

Returned result

The returned value for this method is None.

Delete Bucket

Feature description

This API is used to delete an existing Bucket under the specified account. The Bucket must be empty before it can be deleted.

Method prototype

```
delete_bucket(Bucket)
```

Request example

```
response = client.delete_bucket(
Bucket='test01-123456789'
)
```

Parameters

Parameter Name	Description	Type	Required

Parameter Name	Description	Type	Required
Bucket	Name of the Bucket to be deleted, in the format of bucketname-appid	String	Yes

Returned result

The returned value for this method is None.

Check whether a Bucket exists

Feature description

This API is used to check whether a bucket exists or whether you have the access to it.

Method prototype

```
head_bucket(Bucket)
```

Request example

```
response = client.head_bucket(
Bucket='test01-123456789'
)
```

Parameters

Parameter Name	Description	Type	Required
Bucket	The name of the Bucket to be queried, in the format of bucketname-appid	String	Yes

Returned result

The returned value for this method is None.

Obtain the Bucket's region information

Feature description

This API is used to query the information on the region in which a bucket resides.

Method prototype

```
get_bucket_location(Bucket)
```

Request example

```
response = client.get_bucket_location(
Bucket='test01-123456789'
)
```

Parameters

Parameter Name	Description	Type	Required
Bucket	The name of the Bucket to be queried, in the format of bucketname-appid	String	Yes

Returned result

The Bucket's region information. Type is dict.

```
{
'LocationConstraint': 'ap-beijing-1'|'ap-beijing'|'ap-shanghai'|'ap-guangzhou'|'ap-chengdu'|'ap-chongqing'|'ap-singapore'|'ap-hongkong'|'na-toronto'|'eu-frankfurt'|'ap-mumbai'|'ap-seoul'|'na-siliconvalley'|'na-ashburn'
}
```

Parameter Name	Description	Type
LocationConstraint	The information on the region in which the Bucket resides	String

List all files under the Bucket

Feature description

This API is used to get all Objects under the specified Bucket.

Method prototype

```
list_objects(Bucket, Delimiter="", Marker="", MaxKeys=1000, Prefix="", EncodingType="", **kwargs)
```

Request example

```
response = client.list_objects(
Bucket='test01-123456789',
Delimiter='string',
Marker='string',
MaxKeys=100,
Prefix='string',
EncodingType='url'
)
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name, in the format of bucketname-appid	String	Yes
Delimiter	Sets a delimiter, for example, as "/" to simulate a folder. It is left empty by default.	String	No
Marker	Marks the starting point of the list of returned objects. Entries are listed using UTF-8 binary order by default.	String	No
MaxKeys	The maximum number of returned objects. Default is 1,000.	Int	No
Prefix	Filters the keys of objects by matching the objects prefixed with this parameter. It is left empty by default.	String	No
EncodingType	Indicates the encoding method of the returned value. The value is not encoded by default. Available value: url	String	No

Returned result

The meta information of objects. Type is dict:

```
{
'MaxKeys': '1000',
'Prefix': 'string',
'Delimiter': 'string',
'Marker': 'string',
'NextMarker': 'string',
'Name': 'test04-1252448703',
'IsTruncated': 'false'|'true',
'EncodingType': 'url',
'Contents':[
{
'ETag': '"a5b2e1cfb08d10f6523f7e6fbf3643d5"',
'StorageClass': 'STANDARD',
'Key': 'zh.cn.txt'
'Owner': {
'DisplayName': '1252448703',
'ID': '1252448703'
},
'LastModified': '2017-08-08T09:43:35.000Z',
'Size': '23'
},
],
'CommonPrefixes':[
{
'Prefix': 'string'
},
],
}
```

Parameter Name	Description	Type
MaxKeys	The maximum number of returned objects. Default is 1,000.	String
Prefix	Filters the keys of objects by matching the objects prefixed with this parameter. It is left empty by default.	String
Delimiter	Sets a delimiter, for example, as "/" to simulate a folder. It is left empty by default.	String
Marker	Marks the starting point of the list of returned objects. Entries are listed using UTF-8 binary order by default.	String
NextMarker	Marks the starting point of the next list of returned if IsTruncated is true.	String
Name	Bucket name, in the format of bucketname-appid	String
IsTruncated	Indicates whether the returned objects are truncated	String
EncodingType	Indicates the encoding method of the returned value. The value is not encoded by default. Available value: url	String
Contents	The list containing the meta information of all objects, including 'ETag', 'StorageClass', 'Key', 'Owner', 'LastModified', 'Size', etc.	List
CommonPrefixes	All keys starting with Prefix and ending with Delimiter are grouped into the same type	List

List all multipart uploads under the Bucket

Feature description

This API is used to get all multipart uploads in progress under the specified Bucket.

Method prototype

```
list_multipart_uploads(Bucket, Prefix="", Delimiter="", KeyMarker="", UploadIdMarker="", MaxUploads=1000, EncodingType="", **kwargs)
```

Request example

```
response = client.list_multipart_uploads(
Bucket='test01-123456789',
Prefix='string',
Delimiter='string',
KeyMarker='string',
UploadIdMarker='string'
MaxUploads=100,
EncodingType='url'
)
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name, in the format of bucketname-appid	String	Yes
Prefix	Filters the keys of multipart uploads by matching the multipart uploads prefixed with this parameter. It is left empty by default.	String	No
Delimiter	Sets a delimiter. It is left empty by default.	String	No
KeyMarker	Marks the starting point of a multipart upload task. It is used with UploadIdMarker.	String	No
UploadIdMarker	Marks the starting point of a multipart upload task. It is used with KeyMarker. If KeyMarker is not specified, UploadIdMarker will be ignored.	String	No
MaxUploads	The maximum number of returned multipart uploads. Default is 1,000.	Int	No
EncodingType	Indicates the encoding method of the returned value. The value is not encoded by default. Available value: url	String	No

Returned result

The information of a multipart upload task. Type is dict:

```
{
'Bucket': 'test04-1252448703',
'Prefix': 'string',
'Delimiter': 'string',
'KeyMarker': 'string',
'UploadIdMarker': 'string',
'NextKeyMarker': 'string',
'NextUploadIdMarker': 'string',
'MaxUploads': '1000',
'IsTruncated': 'true'|'false',,
'EncodingType': 'url',
'Upload':[
{
'UploadId': 'string',
'Key': 'string',
'Initiated': 'string',
'StorageClass': 'STANDARD',
'Owner': {
'DisplayName': 'string',
'ID': 'string'
},
'Initiator': {
'ID': 'string',
'DisplayName': 'string'
```

```
}
},
],
'CommonPrefixes':[
{
'Prefix': 'string'
},
],
}
```

Parameter Name	Description	Type
Bucket	Bucket name, in the format of bucketname-appid	String
Prefix	Filters the keys of multipart uploads by matching the multipart uploads prefixed with this parameter. It is left empty by default.	String
Delimiter	Sets a delimiter. It is left empty by default.	String
KeyMarker	Marks the starting point of a multipart upload task. It is used with UploadIdMarker.	String
UploadIdMarker	Marks the starting point of uploadid for a multipart upload task. It is used with KeyMarker. If KeyMarker is not specified, UploadIdMarker will be ignored.	String
NextKeyMarker	Marks the starting point of the next list of keys of multipart uploads	String
NextUploadIdMarker	Marks the starting point of the next list of uploadid of multipart uploads	String
MaxUploads	The maximum number of returned multipart uploads. Default is 1,000.	Int
IsTruncated	Indicates whether the returned multipart uploads are truncated	String
EncodingType	Indicates the encoding method of the returned value. The value is not encoded by default. Available value: url	String
Upload	The list containing information of all multipart uploads, including 'UploadId', 'StorageClass', 'Key', 'Owner', 'Initiator', 'Initiated', etc.	List
CommonPrefixes	All keys starting with Prefix and ending with Delimiter are grouped into the same type	List

Set Bucket ACL information

Feature description

This API is used to set the Bucket ACL information by passing header through ACL, GrantFullControl, GrantRead, GrantWrite or by passing body through AccessControlPolicy. You can only use one of these two methods, otherwise a conflict is returned.

Method prototype

```
put_bucket_acl(Bucket, AccessControlPolicy={}, **kwargs)
```

Request example

```
response = client.put_bucket_acl(
Bucket='test01-123456789',
ACL='private'|'public-read'|'public-read-write',
GrantFullControl='string',
GrantRead='string',
GrantWrite='string',
AccessControlPolicy={
'Grant': [
{
'Grantee': {
'DisplayName': 'string',
'Type': 'CanonicalUser'|'Group',
'ID': 'string',
'URI': 'string'
},
'Permission': 'FULL_CONTROL'|'WRITE'|'READ'
},
],
'Owner': {
'DisplayName': 'string',
'ID': 'string'
}
}
)
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name, in the format of bucketname-appid	String	Yes
ACL	Sets the bucket ACL, such as 'private', 'public-read', and 'public-read-write'	String	No
GrantFullControl	Grants the specified account the permission to read and write buckets in the format of `id=" ",id=" "` . For authorization to a sub-account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{SubUin}"` . For authorization to a root account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{OwnerUin}"` . For example, `'id="qcs::cam::uin/123:uin/456",id="qcs::cam::uin/123:uin/123"'`	String	No

Parameter Name	Description	Type	Required
GrantRead	Grants the specified account the permission to read buckets in the format of `id=" ",id=" "` . For authorization to a sub-account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{SubUin}"` . For authorization to a root account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{OwnerUin}"` . For example, `'id="qcs::cam::uin/123:uin/456",id="qcs::cam::uin/123:uin/123"'`	String	No
GrantWrite	Grants the specified account the permission to write buckets in the format of `id=" ",id=" "` . For authorization to a sub-account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{SubUin}"` . For authorization to a root account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{OwnerUin}"` . For example, `'id="qcs::cam::uin/123:uin/456",id="qcs::cam::uin/123:uin/123"'`	String	No
AccessControlPolicy	Grants the specified account the access to buckets. For more information on the format, please see the response for "get bucket acl".	Dict	No

Returned result

The returned value for this method is None.

Get Bucket ACL information

Feature description

This API is used to set the ACL information of the specified Bucket.

Method prototype

```
get_bucket_acl(Bucket, **kwargs)
```

Request example

```
response = client.get_bucket_acl(
Bucket='test01-123456789',
)
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name, in the format of bucketname-appid	String	Yes

Returned result

Bucket ACL information. Type is dict.

```
{
'Owner': {
'DisplayName': 'string',
'ID': 'string'
},
'Grant': [
{
'Grantee': {
'DisplayName': 'string',
'Type': 'CanonicalUser'|'Group',
'ID': 'string',
'URI': 'string'
},
'Permission': 'FULL_CONTROL'|'WRITE'|'READ'
},
]
}
```

Parameter Name	Description	Type
Owner	Information of the Bucket owner, including DisplayName and ID	Dict
Grant	Information of a user granted the Bucket permissions, including Grantee and Permission	List
Grantee	Information of grantee, including DisplayName, Type, ID and URI	Dict
DisplayName	Name of grantee	String
Type	Type of grantee: CanonicalUser and Group	String
ID	ID of grantee when Type is CanonicalUser	String
URI	URI of grantee when Type is Group	String
Permission	Bucket permissions of grantee. Available values: FULL_CONTROL (read and write permissions), WRITE (write permission), and READ (read permission)	String

Set Bucket cross-origin configuration

Feature description

This API is used to set the cross-origin resource configuration for the specified Bucket.

Method prototype

```
put_bucket_cors(Bucket, CORSConfiguration={}, **kwargs)
```

Request example

```
response = client.put_bucket_cors(
Bucket='test01-123456789',
CORSConfiguration={
'CORSRule': [
{
'ID': 'string',
'MaxAgeSeconds': 100,
'AllowedOrigin': [
'string',
],
'AllowedMethod': [
'string',
],
'AllowedHeader': [
'string',
],
'ExposeHeader': [
'string',
]
}
]
},
)
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name, in the format of bucketname-appid	String	Yes
CORSRule	Sets the appropriate cross-origin rules, including ID, MaxAgeSeconds, AllowedOrigin, AllowedMethod, AllowedHeader, and ExposeHeader	List	Yes
ID	Sets rule ID	String	No
MaxAgeSeconds	Sets the validity period of the results obtained by OPTIONS	Int	No
AllowedOrigin	Sets allowed access sources, e.g. `"http://cloud.tencent.com"` . The wildcard "*" is supported.	Dict	Yes
AllowedMethod	Sets allowed methods, including GET, PUT, HEAD, POST, and DELETE	Dict	Yes
AllowedHeader	Sets the custom HTTP request headers that are allowed to be used by requests. The wildcard "*" is supported.	Dict	No
ExposeHeader	Sets the custom header information that can be received by the browser from the server end.	Dict	No

Returned result

The returned value for this method is None.

Get Bucket cross-origin configuration

Feature description

This API is used to get the cross-origin configuration of the specified Bucket.

Method prototype

```
get_bucket_cors(Bucket, **kwargs)
```

Request example

```
response = client.get_bucket_cors(
Bucket='test01-123456789',
)
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name, in the format of bucketname-appid	String	Yes

Returned result

Bucket cross-origin configuration. Type is dict.

```
{
'CORSRule': [
{
'ID': 'string',
'MaxAgeSeconds': 100,
'AllowedOrigin': [
'string',
],
'AllowedMethod': [
'string',
],
'AllowedHeader': [
'string',
],
'ExposeHeader': [
'string',
],
}
]
}
```

Parameter Name	Description	Type
CORSRule	Cross-origin rules, including ID, MaxAgeSeconds, AllowedOrigin, AllowedMethod, AllowedHeader, and ExposeHeader	List
ID	Rule ID	String
MaxAgeSeconds	The validity period of the results obtained by OPTIONS request	String
AllowedOrigin	Allowed access sources, e.g. `"http://cloud.tencent.com"` . The wildcard "*" is supported.	Dict
AllowedMethod	Allowed methods, including GET, PUT, HEAD, POST, and DELETE	Dict
AllowedHeader	The custom HTTP request headers that are allowed to be used by requests. The wildcard "*" is supported.	Dict
ExposeHeader	Sets the custom header information that can be received by the browser from the server end.	Dict

Delete Bucket cross-origin configuration

Feature description

This API is used to delete the cross-origin configuration of the specified Bucket.

Method prototype

```
delete_bucket_cors(Bucket, **kwargs)
```

Request example

```
response = client.delete_bucket_cors(
Bucket='test01-123456789',
)
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name, in the format of bucketname-appid	String	Yes

Returned result

The returned value for this method is None.

Set Bucket lifecycle configuration

Feature description

This API is used to set the lifecycle configuration of the specified Bucket.

Method prototype

```
put_bucket_lifecycle(Bucket, LifecycleConfiguration={}, **kwargs)
```

Request example

```
from qcloud_cos import get_date
response = client.put_bucket_lifecycle(
Bucket='test01-123456789',
LifecycleConfiguration={
'Rule': [
{
'ID': 'string',
'Filter': {
'Prefix': 'string',
'Tag': [
{
'Key': 'string',
'Value': 'string'
}
]
},
'Status': 'Enabled'|'Disabled',
'Expiration': {
'Days': 100,
'Date': get_date(2018, 4, 20)
},
'Transition': [
{
'Days': 100,
'Date': get_date(2018, 4, 20),
'StorageClass': 'Standard_IA'|'Archive'
},
],
'NoncurrentVersionExpiration': {
'NoncurrentDays': 100
},
'NoncurrentVersionTransition': [
{
'NoncurrentDays': 100,
'StorageClass': 'Standard_IA'
},
],
'AbortIncompleteMultipartUpload': {
'DaysAfterInitiation': 100
}
}
]
```

```
}
)
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name, in the format of bucketname-appid	String	Yes
Rule	Sets the appropriate rules, including ID, Filter, Status, Expiration, Transition, NoncurrentVersionExpiration, NoncurrentVersionTransition, and AbortIncompleteMultipartUpload	List	Yes
ID	Sets rule ID	String	No
Filter	Describes a collection of Objects that are subject to the rules. To set rules for all objects in the bucket, leave Prefix empty.	Dict	Yes
Status	Sets whether Rule is enabled. Available values: Enabled or Disabled	Dict	Yes
Expiration	Sets the expiration rule for Object. You can specify the number of days (Days) or the specified date (Date). The format of Date must be GMT ISO 8601. You can specify the date using get_date method.	Dict	No
Transition	Sets the rule for changing the storage type of Object. You can specify the number of days (Days) or the specified date (Date). The format of Date must be GMT ISO 8601. You can specify the date using get_date method. Available values for StorageClass: Standard_IA and Archive. Multiple rules can be set at a time.	List	No
NoncurrentVersionExpiration	Sets the expiration rule for noncurrent Object versions. You can specify the number of days (NoncurrentDays).	Dict	No
NoncurrentVersionTransition	Sets the rule for changing the storage type of noncurrent Object versions. You can specify the number of days (NoncurrentDays). Available value for StorageClass: Standard_IA. Multiple rules can be set at a time.	List	No
AbortIncompleteMultipartUpload	Indicates the number of days within which the multipart upload must be completed after the upload starts.	Dict	No

Returned result

The returned value for this method is None.

Get Bucket lifecycle configuration

Feature description

This API is used to get the lifecycle configuration of the specified Bucket.

Method prototype

```
get_bucket_lifecycle(Bucket, **kwargs)
```

Request example

```
response = client.get_bucket_lifecycle(
Bucket='test01-123456789',
)
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name, in the format of bucketname-appid	String	Yes

Returned result

Bucket lifecycle configuration. Type is dict.

```
{
'Rule': [
{
'ID': 'string',
'Filter': {
'Prefix': 'string',
'Tag': [
{
'Key': 'string',
'Value': 'string'
}
]
},
'Status': 'string',
'Expiration': {
'Days': 100,
'Date': 'string'
},
'Transition': [
{
'Days': 100,
'Date': 'string',
```

```
'StorageClass': 'STANDARD_IA'|'Archive'
},
],
'NoncurrentVersionExpiration': {
'NoncurrentDays': 100
},
'NoncurrentVersionTransition': [
{
'NoncurrentDays': 100,
'StorageClass': 'STANDARD_IA'
},
],
'AbortIncompleteMultipartUpload': {
'DaysAfterInitiation': 100
}
}
]
}
```

Parameter Name	Description	Type
Rule	Rules, including ID, Filter, Status, Expiration, Transition, NoncurrentVersionExpiration, NoncurrentVersionTransition, and AbortIncompleteMultipartUpload	List
ID	Rule ID	String
Filter	Describes a collection of Objects that are subject to the rules.	Dict
Status	Indicates whether the rule is enabled. Available values: Enabled and Disabled	Dict
Expiration	Expiration rule for Object. You can specify the number of days (Days) or the specified date (Date).	Dict
Transition	Rule for changing the storage type of Object. You can specify the number of days (Days) or the specified date (Date). Available values for StorageClass: Standard_IA and Archive.	List
NoncurrentVersionExpiration	Expiration rule for noncurrent Object versions. You can specify the number of days (NoncurrentDays).	Dict
NoncurrentVersionTransition	Rule for changing the storage type of noncurrent Object versions. You can specify the number of days (NoncurrentDays). Available value for StorageClass: Standard_IA.	List
AbortIncompleteMultipartUpload	Number of days within which the multipart upload must be completed after the upload starts.	Dict

Delete Bucket lifecycle configuration

Feature description

This API is used to delete the lifecycle configuration of the specified Bucket.

Method prototype

```
delete_bucket_lifecycle(Bucket, **kwargs)
```

Request example

```
response = client.delete_bucket_lifecycle(
Bucket='test01-123456789',
)
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name, in the format of bucketname-appid	String	Yes

Returned result

The returned value for this method is None.

Object APIs

Simple Upload of File

Feature description

This API is used to upload a local file or an input stream to the specified Bucket. It is recommended to upload small files not larger than 20 MB. The file size for a single upload is limited to 5 GB. Use multipart upload to upload larger files.

Method prototype

```
put_object(Bucket, Body, Key, **kwargs)
```

Request example

```
response = client.put_object(
Bucket='test01-123456789',
Body=b'abc'|file,
Key='test.txt',
)
```

Request example for all parameters

```
response = client.put_object(
Bucket='test01-123456789',
Body=b'abc'|file,
Key='test.txt',
ACL='private'|'public-read'|'public-read-write', # Use this parameter with caution. Otherwise, a limit of 1,000 ACL r
ules may be reached.
GrantFullControl='string',
GrantRead='string',
GrantWrite='string',
StorageClass='STANDARD'|'STANDARD_IA',
Expires='string',
CacheControl='string',
ContentType='string',
ContentDisposition='string',
ContentEncoding='string',
ContentLanguage='string',
ContentLength='123',
ContentMD5='string',
Metadata={
'x-cos-meta-key1': 'value1',
'x-cos-meta-key2': 'value2'
}
)
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name, in the format of bucketname-appid	String	Yes
Body	The content of the uploaded file, which can be a file stream or a byte stream	file/bytes	Yes
Key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name `bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg`, the object key is doc1/pic1.jpg.	String	Yes
ACL	Sets file ACL, such as 'private', 'public-read', and 'public-read-write'	String	No
GrantFullControl	Grants the specified account the permission to read and write files in the format of `id=" ",id=" "`. For authorization to a sub-account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{SubUin}"`. For authorization to a root account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{OwnerUin}"`. For example, `'id="qcs::cam::uin/123:uin/456",id="qcs::cam::uin/123:uin/123"'`	String	No

Parameter Name	Description	Type	Required
GrantRead	Grants the specified account the permission to read files in the format of `id="",id=""`. For authorization to a sub-account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{SubUin}"`. For authorization to a root account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{OwnerUin}"`. For example, `'id="qcs::cam::uin/123:uin/456",id="qcs::cam::uin/123:uin/123"'`	String	No
GrantWrite	Grants the specified account the permission to write files in the format of `id="",id=""`. For authorization to a sub-account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{SubUin}"`. For authorization to a root account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{OwnerUin}"`. For example, `'id="qcs::cam::uin/123:uin/456",id="qcs::cam::uin/123:uin/123"'`	String	No
StorageClass	Sets file storage type: STANDARD and STANDARD_IA. Default: STANDARD	String	No
Expires	Sets Content-Expires	String	No
CacheControl	Cache policy. Sets Cache-Control	String	No
ContentType	Content type. Sets Content-Type	String	No
ContentDisposition	File name. Sets Content-Disposition	String	No
ContentEncoding	Encoding format. Sets Content-Encoding	String	No
ContentLanguage	Language type. Sets Content-Language	String	No
ContentLength	Sets transmission length	String	No
ContentMD5	Sets MD5 of the uploaded file for verification	String	No
Metadata	User-defined file meta information. It must start with x-cos-meta. Otherwise, it will be ignored.	Dict	No

Returned result

Attributes of the uploaded file. Type is dict:

```
{
'ETag': 'string',
'x-cos-expiration': 'string'
}
```

Parameter Name	Description	Type
ETag	MD5 of the uploaded file	String
x-cos-expiration	After the lifecycle is set, the file expiration rule is returned	String

Download files

Feature description

This API is used to download the files of the specified Bucket locally.

Method prototype

```
get_object(Bucket, Key, **kwargs)
```

Request example

```
response = client.get_object(
Bucket='test01-123456789',
Key='test.txt',
Range='string',
IfMatch='string',
IfModifiedSince='string',
IfNoneMatch='string',
IfUnmodifiedSince='string',
ResponseCacheControl='string',
ResponseContentDisposition='string',
ResponseContentEncoding='string',
ResponseContentLanguage='string',
ResponseContentType='string',
ResponseExpires='string',
VersionId='string'
)
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name, in the format of bucketname-appid	String	Yes
Key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name `bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg` , the object key is doc1/pic1.jpg.	String	Yes
Range	Sets the range of the downloaded file, in the format of bytes=first-last.	String	No
IfMatch	The file is returned if ETag is identical to the specified value	String	No
IfModifiedSince	The file is returned after it has been modified since the specified time	String	No

Parameter Name	Description	Type	Required
IfNoneMatch	The file is returned if ETag is different from the specified value	String	No
IfUnmodifiedSince	The file is returned if it has been modified at or before the specified time	String	No
ResponseCacheControl	Sets response header Cache-Control	String	No
ResponseContentDisposition	Sets Content-Disposition in the response header	String	No
ResponseContentEncoding	Sets Content-Encoding in the response header	String	No
ResponseContentLanguage	Sets Content-Language in the response header	String	No
ResponseContentType	Sets Content-Type in the response header	String	No
ResponseExpires	Sets Content-Expires in the response header	String	No
VersionId	Specifies the version of the downloaded file	String	No

Returned result

Body and meta information of the download file. Type is dict:

```
{
'Body': StreamBody(),
'Accept-Ranges': 'bytes',
'Content-Type': 'application/octet-stream',
'Content-Length': '16807',
'Content-Disposition': 'attachment; filename="filename.jpg"',
'Content-Range': 'bytes 0-16086/16087',
'ETag': '"9a4802d5c99dafe1c04da0a8e7e166bf"',
'Last-Modified': 'Wed, 28 Oct 2014 20:30:00 GMT',
'x-cos-request-id': 'NTg3NzQ3ZmVfYmRjMzVfMzE5N182NzczMQ=='
}
```

Parameter Name	Description	Type
Body	The content of the downloaded file. You can get a file stream by means of get_raw_stream, and download the file content to the specified local file using `get_stream_to_file` .	StreamBody
File meta information	Meta information of the downloaded file, including Etag and x-cos-request-id. The meta information of the configured file is also returned.	String

Get pre-signed download URL

Feature description

This API is used to get a pre-signed download URL to directly download a file.

Method prototype

```
get_presigned_download_url(Bucket, Key, Expired=300)
```

Request example

```
response = client.get_presigned_download_url(
Bucket='test01-123456789',
Key='test.txt'
)
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name, in the format of bucketname-appid	String	Yes
Key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg , the object key is doc1/pic1.jpg.	String	Yes
Expired	Signature expiration time (in seconds)	Int	No

Returned result

The returned value for this method is pre-signed URL.

Delete a file

Feature description

This API is used to delete a file in the specified Bucket.

Method prototype

```
delete_object(Bucket, Key, **kwargs)
```

Request example

```
response = client.delete_object(
Bucket='test01-123456789',
Key='test.txt'
)
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name, in the format of bucketname-appid	String	Yes
Key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg , the object key is doc1/pic1.jpg.	String	Yes

Returned result

The returned value for this method is None.

Batch deletion of files

Feature description

This API is used to delete the files in the specified Bucket in batches.

Method prototype

```
delete_objects(Bucket, Delete={}, **kwargs)
```

Request example

```
response = client.delete_objects(
Bucket='test01-123456789',
Delete={
'Object': [
{
'Key': 'string',
},
],
'Quiet': 'true'|'false'
}
)
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name, in the format of bucketname-appid	String	Yes
Delete	Indicates the method by which the result is returned for the deletion and the target Object	Dict	Yes
Object	Provides the information of each target Object to be deleted	List	Yes

Parameter Name	Description	Type	Required
Key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg , the object key is doc1/pic1.jpg.	String	No
Quiet	Indicates the method by which the result is returned for the deletion. Available values: 'true' and 'false'. Default is 'false'. If it is set to 'true', only error message for failed deletion is returned. If it is set to 'false', messages indicating successful and failed deletion are returned.	String	No

Returned result

Result of batch deletion of files. Type is dict:

```
{
'Deleted': [
{
'Key': 'string',
},
],
'Error': [
{
'Key': 'string',
'Code': 'string',
'Message': 'string'
},
]
}
```

Parameter Name	Description	Type
Deleted	The information of the Object that has been deleted	List
Key	The path of the Object that has been deleted	String
Error	The information of the Object that failed to be deleted	List
Key	The path of the Object that failed to be deleted	String
Code	The error code for the Object that failed to be deleted	String
Message	The error message for the Object that failed to be deleted	String

Obtain file attributes

Feature description

This API is used to obtain the meta information of the specified file.

Method prototype

```
head_object(Bucket, Key, **kwargs)
```

Request example

```
response = client.head_object(
Bucket='test01-123456789',
Key='test.txt',
IfModifiedSince='string'
)
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name, in the format of bucketname-appid	String	Yes
Key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name `bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg` , the object key is doc1/pic1.jpg.	String	Yes
IfModifiedSince	The file is returned after it has been modified since the specified time	String	No

Returned result

The meta information of the file obtained. Type is dict:

```
{
'Content-Type': 'application/octet-stream',
'Content-Length': '16807',
'ETag': '"9a4802d5c99dafe1c04da0a8e7e166bf"',
'Last-Modified': 'Wed, 28 Oct 2014 20:30:00 GMT',
'x-cos-request-id': 'NTg3NzQ3ZmVfYmRjMzVfMzE5N182NzczMQ=='
}
```

Parameter Name	Description	Type
File meta information	Meta information of the file obtained, including ETag and x-cos-request-id. The meta information of the configured file is also included.	String

Create multipart upload

Feature description

This API is used to create a new multipart upload task. UploadId is returned.

Method prototype

```
create_multipart_upload(Bucket, Key, **kwargs):
```

Request example

```
response = client.create_multipart_upload(
Bucket='test01-123456789',
Key='multipart.txt',
StorageClass='STANDARD'|'STANDARD_IA',
Expires='string'
CacheControl='string',
ContentType='string',
ContentDisposition='string',
ContentEncoding='string',
ContentLanguage='string',
Metadata={
'x-cos-meta-key1': 'value1',
'x-cos-meta-key2': 'value2'
},
ACL='private'|'public-read'|'public-read-write',
GrantFullControl='string',
GrantRead='string',
GrantWrite='string'
)
# Obtain UploadId for use by subsequent APIs
uploadid = response['UploadId']
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name, in the format of bucketname-appid	String	Yes
Key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name `bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg` , the object key is doc1/pic1.jpg.	String	Yes
StorageClass	Sets file storage type: STANDARD and STANDARD_IA. Default: STANDARD	String	No
Expires	Sets Content-Expires	String	No
CacheControl	Cache policy. Sets Cache-Control	String	No
ContentType	Content type. Sets Content-Type	String	No
ContentDisposition	File name. Sets Content-Disposition	String	No

Parameter Name	Description	Type	Required
ContentEncoding	Encoding format. Sets Content-Encoding	String	No
ContentLanguage	Language type. Sets Content-Language	String	No
Metadata	User-defined file meta information	Dict	No
ACL	Sets file ACL, such as 'private', 'public-read', and 'public-read-write'	String	No
GrantFullControl	Grants the specified account the permission to read and write files in the format of `id=" ",id=" "` . For authorization to a sub-account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{SubUin}"` . For authorization to a root account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{OwnerUin}"` . For example, `'id="qcs::cam::uin/123:uin/456",id="qcs::cam::uin/123:uin/123"'`	String	No
GrantRead	Grants the specified account the permission to read files in the format of `id=" ",id=" "` . For authorization to a sub-account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{SubUin}"` . For authorization to a root account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{OwnerUin}"` . For example, `'id="qcs::cam::uin/123:uin/456",id="qcs::cam::uin/123:uin/123"'`	String	No
GrantWrite	Grants the specified account the permission to write files in the format of `id=" ",id=" "` . For authorization to a sub-account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{SubUin}"` . For authorization to a root account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{OwnerUin}"` . For example, `'id="qcs::cam::uin/123:uin/456",id="qcs::cam::uin/123:uin/123"'`	String	No

Returned result

The initialization information of the multipart upload task obtained. Type is dict:

```
{
'UploadId': '150219101333cecfd6718d0caea1e2738401f93aa531a4be7a2afee0f8828416f3278e5570',
'Bucket': 'test01-123456789',
'Key': 'multipartfile.txt'
}
```

Parameter Name	Description	Type
UploadId	Indicates the ID of multipart upload	String
Bucket	Bucket name, in the format of bucket-appid	String
Key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name `bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg` , the object key is doc1/pic1.jpg.	String

Abort multipart upload

Feature description

This API is used to abort a multipart upload task, and all uploaded parts are deleted.

Method prototype

```
abort_multipart_upload(Bucket, Key, UploadId, **kwargs)
```

Request example

```
response = client.abort_multipart_upload(
Bucket='test01-123456789',
Key='multipart.txt',
UploadId=uploadid
)
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name, in the format of bucketname-appid	String	Yes
Key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name `bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg` , the object key is doc1/pic1.jpg.	String	Yes
UploadId	Indicates the ID of multipart upload	String	Yes

Returned result

The returned value for this method is None.

Upload a part

Feature description

This API is used to upload a part to the specified UploadId. The size of a part is limited to 5 GB.

Method prototype

```
upload_part(Bucket, Key, Body, PartNumber, UploadId, **kwargs)
```

Request example

```
# Note: The maximum number of parts to be uploaded is 10,000.
response = client.upload_part(
Bucket='test01-123456789',
```

```
Key='multipart.txt',
Body=b'abc'|file,
PartNumber=1,
UploadId='string',
ContentLength=123,
ContentMD5='string'
)
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name, in the format of bucketname-appid	String	Yes
Key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name `bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg` , the object key is doc1/pic1.jpg.	String	Yes
Body	The content of the uploaded part, which can be a local file stream or an input stream	file/bytes	Yes
PartNumber	Indicates the number of the uploaded part	Int	Yes
UploadId	Indicates the ID of multipart upload	String	Yes
ContentLength	Sets transmission length	Int	No
ContentMD5	Sets MD5 of the uploaded file for verification	String	No

Returned result

Attributes of the uploaded part. Type is dict:

```
{
'ETag': 'string'
}
```

Parameter Name	Description	Type
ETag	MD5 of the uploaded part.	String

List uploaded parts

Feature description

This API is used to list the information of the uploaded parts in the specified UploadId.

Method prototype

```
list_parts(Bucket, Key, UploadId, MaxParts=1000, PartNumberMarker=0, EncodingType='', **kwargs)
```

Request example

```
response = client.list_parts(
Bucket='test01-123456789',
Key='multipart.txt',
UploadId=uploadid,
MaxParts=1000,
PartNumberMarker=100,
EncodingType='url'
)
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name, in the format of bucketname-appid	String	Yes
Key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name `bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg` , the object key is doc1/pic1.jpg.	String	Yes
UploadId	Indicates the ID of multipart upload	String	Yes
MaxParts	The maximum number of returned parts. Default is 1,000.	Int	No
PartNumberMarker	Indicates that the parts are listed from the one following PartNumberMarker. Default is 0, which means the parts are listed from the first one.	Int	No
EncodingType	Indicates the encoding method of the returned value. The value is not encoded by default. Available value: url	String	No

Returned result

Information of all uploaded parts. Type is dict:

```
{
'Bucket': 'test01-123456789',
'Key': 'multipartfile.txt',
'UploadId': '1502192444bdb382add546a35b2eeab81e06ed84086ca0bb75ea45ca7fa073fa9cf74ec4f2',
'EncodingType': None,
'MaxParts': '1000',
'IsTruncated': 'true',
'PartNumberMarker': '0',
'NextPartNumberMarker': '1000',
'StorageClass': 'Standard',
```

```
'Part': [
{
'LastModified': '2017-08-08T11:40:48.000Z',
'PartNumber': '1',
'ETag': '"8b8378787c0925f42ccb829f6cc2fb97"',
'Size': '10485760'
},
],
'Initiator': {
'DisplayName': '3333333333',
'ID': 'qcs::cam::uin/3333333333:uin/3333333333'
},
'Owner': {
'DisplayName': '124564654654',
'ID': '124564654654'
}
}
```

Parameter Name	Description	Type
Bucket	Bucket name, in the format of bucketname-appid	String
Key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg , the object key is doc1/pic1.jpg.	String
UploadId	Indicates the ID of multipart upload	String
EncodingType	Indicates the encoding method of the returned value. The value is not encoded by default. Available value: url	String
MaxParts	The maximum number of returned parts. Default is 1,000.	String
IsTruncated	Indicates whether the returned parts are truncated	String
PartNumberMarker	Indicates that the parts are listed from the one following PartNumberMarker. Default is 0, which means the parts are listed from the first one.	String
NextPartNumberMarker	Marks the starting point of the next list of parts	String
StorageClass	File storage type: STANDARD and STANDARD_IA. Default: STANDARD	String
Part	Information of the uploaded part, including ETag, PartNumber, Size, and LastModified	String
Initiator	Creator of the multipart upload, including DisplayName and ID	Dict
Owner	Information of the file owner, including DisplayName and ID	Dict

Complete multipart upload

Feature description

This API is used to construct all parts in the specified UploadId into a complete file. The size of the resulting file must be larger than 1 MB, otherwise an error is returned.

Method prototype

```
complete_multipart_upload(Bucket, Key, UploadId, MultipartUpload={}, **kwargs)
```

Request example

```
response = client.complete_multipart_upload(
Bucket='test01-123456789',
Key='multipart.txt',
UploadId=uploadid,
MultipartUpload={
'Part': [
{
'ETag': 'string',
'PartNumber': 1
},
{
'ETag': 'string',
'PartNumber': 2
},
]
},
)
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name, in the format of bucketname-appid	String	Yes
Key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name `bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg` , the object key is doc1/pic1.jpg.	String	Yes
UploadId	Indicates the ID of multipart upload	String	Yes
MultipartUpload	ETag and PartNumber information for all parts.	Dict	Yes

Returned result

Information about the constructed file. Type is dict:

```
{
'ETag': '"3f866d0050f044750423e0a4104fa8cf-2"',
'Bucket': 'test01-123456789',
'Location': 'test01-123456789.cn-north.myqcloud.com/multipartfile.txt',
'Key': 'multipartfile.txt'
}
```

Parameter Name	Description	Type
ETag	The unique tag of the resulting object. It is not the MD5 check value for the object content, but is only used to check the uniqueness of the object. To verify the file content, you can check the ETag of each part during the process of upload.	String
Bucket	Bucket name, in the format of bucketname-appid	String
Location	URL	String
Key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name `bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg` , the object key is doc1/pic1.jpg.	String

Set Object ACL information

Feature description

This API is used to set the file ACL information by passing header through ACL, GrantFullControl, GrantRead, and GrantWrite or by passing body through AccessControlPolicy. You can only choose one method, otherwise a conflict error is returned.

Method prototype

```
put_object_acl(Bucket, Key, AccessControlPolicy={}, **kwargs)
```

Request example

```
response = client.put_object_acl(
Bucket='test01-123456789',
Key='test.txt',
ACL='private'|'public-read'|'public-read-write',
GrantFullControl='string',
GrantRead='string',
GrantWrite='string',
AccessControlPolicy={
'Grant': [
{
'Grantee': {
'DisplayName': 'string',
'Type': 'CanonicalUser'|'Group',
```

```
'ID': 'string',
'URI': 'string'
},
'Permission': 'FULL_CONTROL'|'WRITE'|'READ'
},
],
'Owner': {
'DisplayName': 'string',
'ID': 'string'
}
}
)
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name, in the format of bucketname-appid	String	Yes
Key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name `bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg` , the object key is doc1/pic1.jpg.	String	Yes
ACL	Sets file ACL, such as 'private', 'public-read', and 'public-read-write'	String	No
GrantFullControl	Grants the specified account the permission to read and write files in the format of `id=" ",id=" "` . For authorization to a sub-account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{SubUin}"` . For authorization to a root account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{OwnerUin}"` . For example, `'id="qcs::cam::uin/123:uin/456",id="qcs::cam::uin/123:uin/123"'`	String	No
GrantRead	Grants the specified account the permission to read files in the format of `id=" ",id=" "` . For authorization to a sub-account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{SubUin}"` . For authorization to a root account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{OwnerUin}"` . For example, `'id="qcs::cam::uin/123:uin/456",id="qcs::cam::uin/123:uin/123"'`	String	No
GrantWrite	Grants the specified account the permission to write files in the format of `id=" ",id=" "` . For authorization to a sub-account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{SubUin}"` . For authorization to a root account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{OwnerUin}"` . For example, `'id="qcs::cam::uin/123:uin/456",id="qcs::cam::uin/123:uin/123"'`	String	No
AccessControlPolicy	Grants the specified account the access to files. For more information on the format, please see the response for "get object acl".	Dict	No

Returned result

The returned value for this method is None.

Get Object ACL information

Feature description

This API is used to get the ACL information of the specified file.

Method prototype

```
get_object_acl(Bucket, Key, **kwargs)
```

Request example

```
response = client.get_object_acl(
Bucket='test01-123456789',
Key='test.txt'
)
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name, in the format of bucketname-appid	String	Yes
Key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name `bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg` , the object key is doc1/pic1.jpg.	String	Yes

Returned result

Bucket ACL information. Type is Dict.

```
{
'Owner': {
'DisplayName': 'string',
'ID': 'string'
},
'Grant': [
{
'Grantee': {
'DisplayName': 'string',
'Type': 'CanonicalUser'|'Group',
'ID': 'string',
'URI': 'string'
},
'Permission': 'FULL_CONTROL'|'WRITE'|'READ'
},
```

```
]
}
```

Parameter Name	Description	Type
Owner	Information of the file owner, including DisplayName and ID	Dict
Grant	Information of a user granted the file permissions, including Grantee and Permission	List
Grantee	Information of grantee, including DisplayName, Type, ID and URI	Dict
DisplayName	Name of grantee	String
Type	Type of grantee: CanonicalUser and Group	String
ID	ID of grantee when Type is CanonicalUser	String
URI	URI of grantee when Type is Group	String
Permission	File permissions of grantee. Available values: FULL_CONTROL (read and write permissions), WRITE (write permission), and READ (read permission)	String

Copy a file

Feature description

This API is used to copy a file from the source path to the destination path, during which the file meta attributes and ACL can be modified. To copy an object, if it is larger than 5 GB and the source and destination objects are in different regions, you must use the API create_multipart_upload() to create a multipart upload, then use upload_part_copy() to copy parts, and use complete_multipart_upload() to complete the multipart upload. If the object to be copied is smaller than or equal to 5 GB, or the source and destination objects are in the same region, you can just call copy_object().

Method prototype

```
copy_object(Bucket, Key, CopySource, CopyStatus='Copy', **kwargs)
```

Request example

```
response = client.copy_object(
Bucket='test01-123456789',
Key='test.txt',
CopySource={
'Appid': '1252408340',
'Bucket': 'test02',
'Key': 'test.txt',
'Region': 'ap-guangzhou'
},
CopyStatus='Copy'|'Replaced',
```

```
ACL='private'|'public-read'|'public-read-write',
GrantFullControl='string',
GrantRead='string',
GrantWrite='string',
StorageClass='STANDARD'|'STANDARD_IA',
Expires='string'
CacheControl='string',
ContentType='string',
ContentDisposition='string',
ContentEncoding='string',
ContentLanguage='string',
Metadata={
'x-cos-meta-key1': 'value1',
'x-cos-meta-key2': 'value2'
}
)
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name, in the format of bucketname-appid	String	Yes
Key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name `bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg` , the object key is doc1/pic1.jpg.	String	Yes
CopySource	Indicates the path of the copied source file, including Appid, Bucket, Key, and Region	Dict	Yes
CopyStatus	Available values: 'Copy' and 'Replaced'. When it is set to 'Copy', ignore the configured user metadata information and copy the file directly. When it is set to 'Replaced', modify the metadata according to the configured meta information. If the destination path is identical to the source path, it must be set to 'Replaced'.	String	Yes
ACL	Sets file ACL, such as 'private', 'public-read', and 'public-read-write'	String	No
GrantFullControl	Grants the specified account the permission to read and write files in the format of `id=" ",id=" "` . <**eci**> For authorization to a sub-account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{SubUin}"` . For authorization to a root account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{OwnerUin}"` . For example, `'id="qcs::cam::uin/123:uin/456",id="qcs::cam::uin/123:uin/123"'`	String	No

Parameter Name	Description	Type	Required
GrantRead	Grants the specified account the permission to read files in the format of `id="",id=""`. For authorization to a sub-account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{SubUin}"`. For authorization to a root account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{OwnerUin}"`. For example, `'id="qcs::cam::uin/123:uin/456",id="qcs::cam::uin/123:uin/123"'`	String	No
GrantWrite	Grants the specified account the permission to write files in the format of `id="",id=""`. For authorization to a sub-account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{SubUin}"`. For authorization to a root account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{OwnerUin}"`. For example, `'id="qcs::cam::uin/123:uin/456",id="qcs::cam::uin/123:uin/123"'`	String	No
StorageClass	Sets file storage type: STANDARD and STANDARD_IA. Default: STANDARD	String	No
Expires	Sets Content-Expires	String	No
CacheControl	Cache policy. Sets Cache-Control	String	No
ContentType	Content type. Sets Content-Type	String	No
ContentDisposition	File name. Sets Content-Disposition	String	No
ContentEncoding	Encoding format. Sets Content-Encoding	String	No
ContentLanguage	Language type. Sets Content-Language	String	No
Metadata	User-defined file meta information	Dict	No

Returned result

Attributes of the uploaded file. Type is dict:

```
{
'ETag': 'string',
'LastModified': 'string',
}
```

Parameter Name	Description	Type
ETag	MD5 of the copied file	String
LastModified	The time when the copied file was last modified	String

Copy parts

Feature description

This API is used to copy the parts of a file from source path to the destination path.

Method prototype

```
upload_part_copy(Bucket, Key, PartNumber, UploadId, CopySource, CopySourceRange='', **kwargs)
```

Request example

```
response = client.upload_part_copy(
Bucket='test01-123456789',
Key='test.txt',
PartNumber=100,
UploadId='string',
CopySource={
'Appid': '1252408340',
'Bucket': 'test02',
'Key': 'test.txt',
'Region': 'ap-guangzhou'
},
CopySourceRange='string',
CopySourceIfMatch='string',
CopySourceIfModifiedSince='string',
CopySourceIfNoneMatch='string',
CopySourceIfUnmodifiedSince='string'
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name, in the format of bucketname-appid	String	Yes
Key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name `bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg` , the object key is doc1/pic1.jpg.	String	Yes
PartNumber	Indicates the number of the uploaded part	Int	Yes
UploadId	Indicates the ID of multipart upload	String	Yes
CopySource	Indicates the path of the copied source file, including Appid, Bucket, Key, and Region	Dict	Yes
CopySourceRange	Indicates the range of the copied file, in the format of bytes=first-last. If it is not specified, the entire source file is copied by default.	String	No
CopySourceIfMatch	The file is copied when the ETag of the source file is identical to the specified value	String	No

Parameter Name	Description	Type	Required
CopySourceIfModifiedSince	The source file is copied after it has been modified since the specified time	String	No
CopySourceIfNoneMatch	The file is copied when the ETag of the source file is different from the specified value	String	No
CopySourceIfUnmodifiedSince	The source file is copied after it is not modified since the specified time	String	No

Returned result

Attributes of the copied part. Type is dict:

```
{
'ETag': 'string',
'LastModified': 'string',
}
```

Parameter Name	Description	Type
ETag	MD5 of the copied part	String
LastModified	The time when the copied part was last modified	String

Restore an archive file

Feature description

This API is used to restore an object that has been archived as "archive" via COS.

Method prototype

```
restore_object(Bucket, Key, RestoreRequest={}, **kwargs)
```

Request example

```
response = client.restore_object(
Bucket='test01-123456789',
Key='test.txt',
RestoreRequest={
'Days': 100,
'CASJobParameters': {
'Tier': 'Expedited'|'Standard'|'Bulk'
}

}
)
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name, in the format of bucketname-appid	String	Yes
Key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name `bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg` , the object key is doc1/pic1.jpg.	String	Yes
RestoreRequest	Describes the rule for restoring temporary files	Dict	Yes
Days	Describes the validity period of a temporary file	Int	Yes
CASJobParameters	Describes the configuration information of restore type	Dict	No
Tier	Describes the mode of restoring temporary files. Available values: 'Expedited' (fast), 'Standard' (moderate), and 'Bulk' (slow).	String	No

Returned result

The returned value for this method is None.

High-level API

Upload files (resuming upload from breakpoint)

Feature description

This API for file upload selects easy upload or multipart upload according to the file length. Easy upload is used for files smaller than or equal to 20 MB. Multipart upload is used for files larger than 20 MB. If the upload of a part is not completed, you can resume the upload from the breakpoint.

Method prototype

```
upload_file(Bucket, Key, LocalFilePath, PartSize=1, MAXThread=5, **kwargs)
```

Request example

```
response = client.upload_file(
Bucket='test01-123456789',
Key='test.txt',
LocalFilePath='local.txt'
)
```

Request example for all parameters

```
response = client.upload_file(
Bucket='test01-123456789',
Key='test.txt',
LocalFilePath='local.txt',
PartSize=1,
MAXThread=5,
ACL='private'|'public-read'|'public-read-write', # Please use this parameter with caution. Otherwise, a limit of 1,000 ACL rules may be reached.
GrantFullControl='string',
GrantRead='string',
GrantWrite='string',
StorageClass='STANDARD'|'STANDARD_IA',
Expires='string',
CacheControl='string',
ContentType='string',
ContentDisposition='string',
ContentEncoding='string',
ContentLanguage='string',
ContentLength='123',
ContentMD5='string',
Metadata={
'x-cos-meta-key1': 'value1',
'x-cos-meta-key2': 'value2'
}
)
```

Parameters

Parameter Name	Description	Type	Required
Bucket	Bucket name, in the format of bucketname-appid	String	Yes
Key	Object key is the unique identifier of the object in the bucket. For example, in the object's access domain name `bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg`, the object key is doc1/pic1.jpg.	String	Yes
LocalFilePath	Name of the path to the local file	String	Yes
PartSize	Part size in multipart upload. Default is 1 MB.	Int	No
MAXThread	The maximum number of parts to be uploaded at a time. Default is 5. Parts are uploaded through threads	Int	No
ACL	Sets file ACL, such as 'private', 'public-read', and 'public-read-write'	String	No

Parameter Name	Description	Type	Required
GrantFullControl	Grants the specified account the permission to read and write files in the format of `id="",id=""`. For authorization to a sub-account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{SubUin}"`. For authorization to a root account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{OwnerUin}"`. For example, `'id="qcs::cam::uin/123:uin/456",id="qcs::cam::uin/123:uin/123"'`	String	No
GrantRead	Grants the specified account the permission to read files in the format of `id="",id=""`. For authorization to a sub-account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{SubUin}"`. For authorization to a root account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{OwnerUin}"`. For example, `'id="qcs::cam::uin/123:uin/456",id="qcs::cam::uin/123:uin/123"'`	String	No
GrantWrite	Grants the specified account the permission to write files in the format of `id="",id=""`. For authorization to a sub-account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{SubUin}"`. For authorization to a root account, the format is `id="qcs::cam::uin/{OwnerUin}:uin/{OwnerUin}"`. For example, `'id="qcs::cam::uin/123:uin/456",id="qcs::cam::uin/123:uin/123"'`	String	No
StorageClass	Sets file storage type: STANDARD and STANDARD_IA. Default: STANDARD	String	No
Expires	Sets Content-Expires	String	No
CacheControl	Cache policy. Sets Cache-Control	String	No
ContentType	Content type. Sets Content-Type	String	No
ContentDisposition	File name. Sets Content-Disposition	String	No
ContentEncoding	Encoding format. Sets Content-Encoding	String	No
ContentLanguage	Language type. Sets Content-Language	String	No
ContentLength	Sets transmission length	String	No
ContentMD5	Sets MD5 of the uploaded file for verification	String	No
Metadata	User-defined file meta information	Dict	No

Returned result

Attributes of the uploaded file. Type is dict:

```
{
'ETag': 'string',
'x-cos-expiration': 'string'
}
```

Parameter Name	Description	Type
ETag	MD5 of the uploaded file	String
x-cos-expiration	After the lifecycle is set, the file expiration rule is returned	String

API for Obtaining Signature

Obtain a signature

Feature description

This API is used to obtain the signature of the specified operation, which is commonly used for signature distribution.

Method prototype

```
get_auth(Method, Bucket, Key, Expired=300, Headers={}, Params={})
```

Request example

```
response = client.get_auth(
Method='PUT'|'POST'|'GET'|'DELETE'|'HEAD',
Bucket='test01-123456789',
Key='test.txt',
Expired=300,
Headers={
'Content-Length': 'string',
'Content-MD5': 'string'
},
Params={
'param1': 'string',
'param2': 'string'
}
)
```

Parameters

Parameter Name	Description	Type	Required
Method	Indicates the method of the operation. Available values: 'PUT', 'POST', 'GET', 'DELETE', and 'HEAD'.	String	Yes
Bucket	Bucket name, in the format of bucketname-appid	String	Yes
Key	For bucket operations, enter a root path "/". For object operations, enter a file path.	String	Yes

Parameter Name	Description	Type	Required
Expired	Signature expiration time (in seconds)	Int	No
Headers	Indicates the request header required in the signature	Dict	No
Params	Indicates the request parameters required in the signature	Dict	No

Returned result

The returned value for this method is the signature value of the corresponding operation.

Exception Types

Exceptions include CosClientError (SDK client error) and CosServiceError (COS server error).

CosClientError

CosClientError generally refers to a client error caused by the reasons such as timeout. When capturing such an error, you can choose to retry or perform other operations.

CosServiceError

CosServiceError provides the message returned by the server. For more information on error codes, please see COS Error Codes.

```
#except CosServiceError as e
e.get_origin_msg() # Get original error message in XML format
e.get_digest_msg() # Get the processed error message in dict format
e.get_status_code()# Get http error code (e.g. 4XX, 5XX)
e.get_error_code() # Get COS-defined error code
e.get_error_msg() # Get a detailed description of the COS error code
e.get_trace_id() # Get the trace_id of the request
e.get_request_id() # Get the request_id of the request
e.get_resource_location()# Get the URL
```