Cloud Object Storage

SDK Documentation

Product Documentation

Contents

SDK Documentation

SDK Overview

Last updated：2024-06-25 10:53:13

Tencent Cloud COS provides a variety of APIs and SDKs for developers. The following table lists the SDKs supported by COS and the corresponding quick start documentation, based on which you can complete SDK download, installation, initialization, and more.

SDK	Quick Start Documentation
Android SDK	Getting Started with Android SDK
C SDK	Getting Started with C SDK
C++ SDK	Getting Started with C++ SDK
.NET SDK	Getting Started with .NET SDK
Go SDK	Getting Started with Go SDK
iOS SDK	Getting Started with iOS SDK
Java SDK	Getting Started with Java SDK
JavaScript SDK	Getting Started with JavaScript SDK
Node.js SDK	Getting Started with Node.js SDK
PHP SDK	Getting Started with PHP SDK
Python SDK	Getting Started with Python SDK
Mini Program SDK	Getting Started with Mini Program SDK

Preparations

Last updated：2024-06-25 10:53:13

The following terms may be involved when you use the SDK:

Term	Description
APPID	A unique user-level resource identifier for COS access. It can be obtained at Manage API Key.
SecretId	A developer-owned secret ID used for the project. It can be obtained at Manage API Key.
SecretKey	A developer-owned secret key used for the project. It can be obtained at Manage API Key.
Bucket	A container used for data storage. For more information, please see Bucket Overview.
BucketName-APPID	Bucket name, with the value of `APPID` as its suffix. Example: `examplebucket-1250000000`
Object	A file stored in COS. It is the basic entity that is stored.
ObjectKey	A unique identifier of an object stored in COS. For more information about objects and object keys, please see Object Overview.
Region	Region information. For more information about the enumerated values (such as ap-beijing, ap-hongkong, and eu-frankfurtplease), please see Regions and Access Endpoints.
ACL	Access Control List, a list of access control information for a specified bucket or object. For more information, please see ACL Overview.

Note:

If you encounter errors such as non-existent functions or methods when using the XML version of the SDK, please update the SDK to the latest version and try again.

Android SDK
Getting Started

Last updated：2024-06-25 10:53:13

Resources

Download the XML Android SDK source code here.
Download the demo here.
For SDK APIs and parameters, see SDK API Reference.
For the complete sample code, see Sample SDK Codes.
For the SDK changelog, see ChangeLog.
For SDK FAQs, see Android SDK FAQs.
Note:
If you encounter errors such as non-existent functions or methods when using the XML version of the SDK, please update the SDK to the latest version and try again.

Preparations

1. You need an Android app, which can be one of your existing projects or a new one.
2. Make sure that your Android app's target API level is 15 (Ice Cream Sandwich) or above.
3. You need a remote address where users can obtain your Tencent Cloud temporary key. For more information on temporary keys, see Direct Upload for Mobile Apps.

Step 1. Install the SDK

Method 1. Automatic integration (recommended)

Note:
As the Bintray repository is no longer available, COS's SDK has been migrated to Maven Central. The import path is different and thus you need to use the new import path during the update.

Using the Maven Central repository

Add the following code to the project-level `build.gradle` file (usually in the root directory):

```
repositories {
        google()
        // Add the following line
        mavenCentral()
    }
```

Standard SDK

Add dependencies to the app-level `build.gradle` file (usually under the App module).

```
dependencies {
    ...
    // Add the following line
    implementation 'com.qcloud.cos:cos-android:5.9.+'
}
```

Simplified SDK

Add dependencies to the app-level `build.gradle` file (usually under the App module).

```
dependencies {
    ...
    // Add the following line
    implementation 'com.qcloud.cos:cos-android-lite:5.9.+'
}
```

Disabling beacon reporting

We have introduced the Tencent Beacon into the SDK to track down and optimize the SDK quality for a better user experience.

Note:

DataInsight only monitors the COS request performance and doesn't report any business data.

To disable this feature, perform the following operations in the app-level `build.gradle` file (usually under the App module):

For v5.8.0 or later:

Change the dependency of `cos-android` to

```
dependencies {
    ...
    // Change to
    implementation 'com.qcloud.cos:cos-android-nobeacon:x.x.x'

    //For lite version, change to
    implementation 'com.qcloud.cos:cos-android-lite-nobeacon:x.x.x'
}
```

For v5.5.8–5.7.9:

Add the beacon removing statement

```
dependencies {
    ...
    implementation ('com.qcloud.cos:cos-android:x.x.x'){
        // Add the following line
        exclude group: 'com.tencent.qcloud', module: 'beacon-android-release'
    }
}
```

Method 2: Manually integrate

1. Download the SDK version

You can directly download the latest SDK version here or find all versions at SDK Releases.

After downloading and decompressing the file, you can see that it contains multiple `JAR` or `AAR` packages as described below. Please choose the ones you want to integrate.

Required libraries:

cos-android: COS protocol implementation

qcloud-foundation: foundation library

bolts-tasks: third-party task library

okhttp: third-party networking library

okio: third-party I/O library

Optional libraries:

quic: QUIC protocol, required if you transfer data over QUIC

beacon: mobile beacon analysis to improve the SDK

2. Integrate the SDK into your project

Put your libraries in the app-module `libs` folder and add dependencies to the app-level `build.gradle` file (usually under the App module):

```
dependencies {
    ...
    // Add the following line
    implementation fileTree(dir: 'libs', include: ['*.jar', '*.aar'])
}
```

Step 2. Configure Permissions

Network permissions

The SDK needs network permission to communicate with the COS server. Please add the following permission declarations to `AndroidManifest.xml` under the app module:

```
<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"/>
```

Storage permissions

If you need to read and write files from external storage, please add the following permission declarations to `AndroidManifest.xml` under the app module:

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

Note:

For Android 6.0 (API level 23) or above, you need to dynamically request storage permissions at runtime.

Step 3. Use the SDK

Note:

We recommend you use a temporary key as instructed in Generating and Using Temporary Keys to call the SDK for security purposes. When you apply for a temporary key, follow the Notes on Principle of Least Privilege to avoid leaking resources besides your buckets and objects.

If you must use a permanent key, we recommend you follow the Notes on Principle of Least Privilege to limit the scope of permission on the permanent key.

1. Obtain a temporary key

Implement a `BasicLifecycleCredentialProvider` subclass to request a temporary key and return the result.

```
public static class MySessionCredentialProvider
        extends BasicLifecycleCredentialProvider {

    @Override
    protected QCloudLifecycleCredentials fetchNewCredentials()
            throws QCloudClientException {

        // First, obtain the response containing the key from your temporary key se
        // Then, parse the response to obtain the temporary key
        String tmpSecretId = "SECRETID"; // SecretId of the temporary key
        String tmpSecretKey = "SECRETKEY"; // SecretKey of the temporary key
        String sessionToken = "SESSIONTOKEN"; // Token of the temporary key
        long expiredTime = 1556183496L;// End timestamp (in seconds) of the effecti

        // To avoid request expiration caused by the large discrepancy between the
        // Time returning to the server as the signature start time
        long startTime = 1556182000L; // Start timestamp (in seconds) of the effect

        // Finally, return the temporary key object
        return new SessionQCloudCredentials(tmpSecretId, tmpSecretKey,
                sessionToken, startTime, expiredTime);
    }
}
```

The following takes `MySessionCredentialProvider` as the class name example to initialize an instance to provide a key for the SDK.

```
QCloudCredentialProvider myCredentialProvider = new MySessionCredentialProvider();
```

Using a permanent key for local debugging

You can use your Tencent Cloud permanent key for local debugging during the development phase. **Since this method exposes the key to leakage risks, please be sure to replace it with a temporary key before launching your application.**

```
String secretId = "SECRETID"; // SecretId of the permanent key
String secretKey = "SECRETKEY"; // SecretKey of the permanent key

// keyDuration is the effective duration (in seconds) of the key in your request
QCloudCredentialProvider myCredentialProvider =
    new ShortTimeCredentialProvider(secretId, secretKey, 300);
```

Using the server-calculated signature to authorize the request

Implement a subclass of `QCloudSelfSigner` to get the server-side signature and add it to the request authorization.

```
QCloudSelfSigner myQCloudSelfSigner = new QCloudSelfSigner() {
    /**
     * Sign the request
     *
     * @param request The request to be signed
     * @throws QCloudClientException Client exception
     */
    @Override
    public void sign(QCloudHttpRequest request) throws QCloudClientException {
        // 1. Pass the request parameters to the server to calculate the signature
        String auth = "get auth from server";
        // 2. Add the signature to the request
        request.addHeader(HttpConstants.Header.AUTHORIZATION, auth);
    }
});
```

2. Initialize a COS instance

Use your `myCredentialProvider` instance that provides the key or the server-side signed and authorized instance `myQCloudSelfSigner` to initialize a `CosXmlService` instance.

`CosXmlService` provides all APIs for accessing COS. We recommend you use it as an **application singleton**.

```
// Bucket region abbreviation. For example, "ap-guangzhou" is the abbreviation of t
String region = "COS_REGION";

// Create a `CosXmlServiceConfig` object and modify the default configuration param
CosXmlServiceConfig serviceConfig = new CosXmlServiceConfig.Builder()
        .setRegion(region)
        .isHttps(true) // Use the HTTPS request. HTTP is used by default
        .builder();

// Initialize the COS service to obtain the instance
CosXmlService cosXmlService = new CosXmlService(context,
    serviceConfig, myCredentialProvider);

// Initialize the COS service via the server-side signature authorization to get th
CosXmlService cosXmlService = new CosXmlService(context,
    serviceConfig, myQCloudSelfSigner);
```

Note:

For more information on the abbreviations of the COS bucket regions, see Regions and Access Endpoints.

Step 4. Access COS

Uploading an object

The SDK supports uploading local files, binary data, URIs, and input streams. The following uses uploading a local file as an example:

```
// Initialize TransferConfig. The default configuration is used here. To customize
TransferConfig transferConfig = new TransferConfig.Builder().build();
```

```
// Initialize TransferManager.
TransferManager transferManager = new TransferManager(cosXmlService,
        transferConfig);

// Bucket name in the format of `BucketName-APPID` (`APPID` is required), which can
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; // Location identifier of the object in the bucke
String srcPath = new File(context.getCacheDir(), "exampleobject")
        .toString(); // Absolute path of the local file
// If there is an uploadId for the initialized multipart upload, assign the value o
String uploadId = null;

// Upload the file
COSXMLUploadTask cosxmlUploadTask = transferManager.upload(bucket, cosPath,
        srcPath, uploadId);

// Set the callback for initializing multipart upload (supported starting from v5.9
cosxmlUploadTask.setInitMultipleUploadListener(new InitMultipleUploadListener() {
    @Override
    public void onSuccess(InitiateMultipartUpload initiateMultipartUpload) {
        //The uploadId required for the subsequent checkpoint restarts
        String uploadId = initiateMultipartUpload.uploadId;
    }
});
// Set the upload progress callback
cosxmlUploadTask.setCosXmlProgressListener(new CosXmlProgressListener() {
    @Override
    public void onProgress(long complete, long target) {
        // todo Do something to update progress...
    }
});
// Set the response callback
cosxmlUploadTask.setCosXmlResultListener(new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        COSXMLUploadTask.COSXMLUploadTaskResult uploadResult =
                (COSXMLUploadTask.COSXMLUploadTaskResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest request,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else{
            serviceException.printStackTrace();
        }
    }
});
// Set the job status callback to view the job progress
cosxmlUploadTask.setTransferStateListener(new TransferStateListener() {
    @Override
    public void onStateChanged(TransferState state) {
        // todo notify transfer state
    }
});
```

Note:

For the complete sample, please visit GitHub.

After the upload, you can generate a download URL for the uploaded file with the same key. For detailed directions, see Generating Pre-signed Links. Please note that for private-read files, the download URL is only valid for a limited period of time.

Downloading an object

```
// The advanced download API supports checkpoint restart. Therefore, a HEAD request
// If you are using a temporary key or accessing with a sub-account, ensure that yo

// Initialize TransferConfig. The default configuration is used here. To customize
TransferConfig transferConfig = new TransferConfig.Builder().build();
// Initialize TransferManager
TransferManager transferManager = new TransferManager(cosXmlService,
        transferConfig);

// Bucket name in the format of `BucketName-APPID` (`APPID` is required), which can
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; // Location identifier of the object in the bucke
// Path of the local directory
String savePathDir = context.getExternalCacheDir().toString();
// File name saved locally. If not specified (null), it will be the same as the COS
String savedFileName = "exampleobject";

Context applicationContext = context.getApplicationContext(); // application
// context
COSXMLDownloadTask cosxmlDownloadTask =
        transferManager.download(applicationContext,
                bucket, cosPath, savePathDir, savedFileName);

// Set the download progress callback
cosxmlDownloadTask.setCosXmlProgressListener(new CosXmlProgressListener() {
    @Override
    public void onProgress(long complete, long target) {
        // todo Do something to update progress...
    }
});
// Set the response callback
cosxmlDownloadTask.setCosXmlResultListener(new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        COSXMLDownloadTask.COSXMLDownloadTaskResult downloadTaskResult =
                (COSXMLDownloadTask.COSXMLDownloadTaskResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest request,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else{
            serviceException.printStackTrace();
        }
    }
});
// Set the job status callback to view the job progress
cosxmlDownloadTask.setTransferStateListener(new TransferStateListener() {
    @Override
    public void onStateChanged(TransferState state) {
        // todo notify transfer state
    }
});
```

Note:

For the complete sample, please visit GitHub.

The advanced download API supports checkpoint restart. Therefore, a HEAD request will be sent before the download to obtain the file information. If you are using a temporary key or accessing with a sub-account, ensure that your permission list includes `HeadObject` .

Android SDK FAQs

Last updated：2024-06-25 10:53:13

What do I do if the client network is normal, while the access to COS over HTTP is very slow, or the error message "Connection reset" is reported?

In some regions, carriers may hijack COS endpoints. Therefore, you are advised to access COS over HTTPS.

What do I do if `ETag` is not included in the `Complete Multipart Upload` request and an error message "400 Bad Request" is reported?

The possible cause is that the `ETag` header is filtered out by the network. After the parts are uploaded, the SDK fails to parse the `ETag` parameter and reports the error in the Complete Multipart Upload operation.

What do I do if `QCloudResultListener` or other callback functions did not work?

If you view logs to determine whether a callback function works, the possible cause is that the log level is set too high, or the desired log is filtered out by other filtering rules. You can adjust the filtering rule, or set breakpoints in the callback function to determine whether the callback function works properly.

What do I do if `NoClassDefFoundError` is reported when I call an API?

The SDK depends on the bolts and OkHttp common classes. If the methods in these classes cannot be found, you might have imported these two dependencies to your project, and the version number is too low. You are advised to use a version consistent with the SDK version or a later one.

What do I do if the SDK failed to obtain permissions to the phone?

To upload files to or download files from an external storage device, you must have network permission and read/write permissions on the device. Other permissions, such as location permission and device information permission, are not mandatory. If you have strict requirements on permissions, you can skip importing the MtaUtils package or upgrade the SDK to 5.5.8 or above.

What do I do if the `java.security.cert.CertPathValidatorException: Trust anchor for certification path not found` error is reported when HTTPS is used?

If you access COS via a proxy, check whether the proxy supports HTTPS. If not, please contact us.

What do I do if the upload progress reaches 100%, while the `onFailed` method is called?

The 100% upload progress indicates only the SDK packet sending progress. The upload is successful only when the `onSuccess` method is called. If an exception occurs when the `Complete Multipart Upload` request is sent, the `onFailed` method will be called. You can check the exception details and solution based on the `onFailed` callback information.

What do I do if an error, such as `400 Bad Request` and `409 Conflict` , occurs in a multipart upload?

Use the advanced API `TransferManager` provided by the SDK for upload/download if possible. Encapsulating the multipart upload API may easily cause an error.

What do I do if a permission error is reported when I use `TransferManager` for upload/download?

The Head operation will be performed when `TransferManager` downloads an object. Therefore, the `HeadObject` and `GetObject` permissions should be granted for the download. For an upload operation, permissions on all simple upload and multipart upload APIs should be granted.

What do I do if an error, such as `lock timeout` , `no credential for sign` , or expired signature, is reported?

If you have implemented the `BasicLifecycleCredentialProvider#fetchNewCredentials()` method, please check whether the key is updated in time, or whether it is still valid. For a temporary key, the token should be carried.

What do I do if the `java.lang.RuntimeException: Can't create handler inside thread that has not called Looper.prepare()` error is reported?

If the error is reported when the `TransferManager#upload()` method is called in the master thread, this is a false positive reported by the MTA and can be ignored. You can also upgrade your SDK to 5.5.8 or above to solve this issue.

What do I do if an application error is reported when I directly operate the UI in a callback?

The SDK callback thread is not necessarily the master thread. Please do not operate the UI directly.

What do I do if `calculate md5 error` is reported during upload?

The possible cause is either that you have modified the file during the upload, changing the MD5 checksum, or the network is poor, causing a packet receive error on the server.

What do I do if `ServerError` is returned for a request?

If you access COS using a proxy, the possible cause is that the proxy returned the incorrect packet, causing the SDK to fail the parsing. You can capture the packet received by the client to verify the packet.

What do I do if the 403 permission error is reported when I call an API?

In general, a permission error is irrelevant to SDK. You can check your permissions or contact us.

Does Android SDK support checkpoint restart?

Advanced APIs of the Android SDK of COS support checkpoint restart. To implement checkpoint restart, refer to the descriptions of the advanced APIs in Uploading and Copying Objects.

Quick Experience

Last updated：2024-06-25 10:53:13

Background

Applications are the building blocks of the mobile Internet. As they often require massive data upload and download, data security and reliability are extremely crucial. Now, developers can let Tencent Cloud COS（Cloud Object Storage） handle data storage for them, allowing them to solely focus on the business logic of their applications with a lighter workload and higher development efficiency. This document describes how to quickly build a COS-based application transfer service to upload and download your application data through Tencent Cloud COS. All you need to do is deploy your business components and generate and manage temporary keys on your server.
COS provides a demo for XML, which you can run as instructed below.

Relevant Resources

Download the demo from the repository on GitHub.

Preparations

Android OS: 4.4 or above.
Your APPID, SecretId, and SecretKey from the API Key Management page on the Tencent Cloud console.

Setting up a User's Application Client

Configuring the client

1. Download the project files from GitHub, and open them in your IDE.
2. Configure your COS_APP_ID, SecretId、SecretKey in the environment variables.
3. Run your project to try out the demo.

Note:

Do not expose your plaintext SecretId or SecretKey to any unsecure environments.
The `ShortTimeCredentialProvider` authorization method in the demo is simply used for demonstration purposes and should not be used in your production environment. Instead, we recommend authorizing via temporary keys.

After the environment variables have been altered, you may need to restart Android Studio for the updated configuration to take effect.

Running the Demo

Querying a bucket list

Open the sample app, and you will see all the buckets that you have created.

Creating a bucket

Click **New Bucket** in the upper right-hand corner, enter the bucket name on the configuration page, and select the region in which the bucket resides.

Querying an object list

Click on a bucket, and you will see all the files and folders that it contains.

Uploading a file

On the file list page, click **Upload** in the upper right corner, and select the file to upload.

Downloading a file

On the file list page, click **Download** below a file to directly download it.

Bucket Operations

Last updated：2024-06-25 10:53:13

Overview

This document provides an overview of APIs and SDK code samples related to basic bucket operations.

API	Operation	Description
GET Service (List Buckets)	Querying a bucket list	Queries the list of all buckets under a specified account
PUT Bucket	Creating a bucket	Creates a bucket under a specified account
HEAD Bucket	Checking a bucket and its permissions	Checks whether a bucket exists and whether you have permission to access it
DELETE Bucket	Deleting a bucket	Deletes an empty bucket from a specified account

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

Querying a Bucket List

Description

This API (GET Service (List Buckets)) is used to query the list of all buckets under a specified account.

Sample code

```
GetServiceRequest getServiceRequest = new GetServiceRequest();
cosXmlService.getServiceAsync(getServiceRequest, new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        GetServiceResult getServiceResult = (GetServiceResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Creating a Bucket

Description

This API (PUT Bucket) is used to create a bucket.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
PutBucketRequest putBucketRequest = new PutBucketRequest(bucket);
cosXmlService.putBucketAsync(putBucketRequest, new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        PutBucketResult putBucketResult = (PutBucketResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Checking a Bucket and Its Permissions

Description

This API is used to verify whether a bucket exists and whether you have permission to access it.

If the bucket exists and you have permission to read it, HTTP status code 200 will be returned.

If you do not have permission to read the bucket, HTTP status code 403 will be returned.

If the bucket does not exist, HTTP status code 404 will be returned.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
HeadBucketRequest headBucketRequest = new HeadBucketRequest(bucket);
cosXmlService.headBucketAsync(headBucketRequest, new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        HeadBucketResult headBucketResult = (HeadBucketResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Deleting a Bucket

Description

This API is used to delete a specified bucket.

Note:

Before deleting a bucket, please make sure that all the data and incomplete multipart uploads in the bucket have been deleted; otherwise, the bucket cannot be deleted.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
DeleteBucketRequest deleteBucketRequest = new DeleteBucketRequest(bucket);
cosXmlService.deleteBucketAsync(deleteBucketRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        DeleteBucketResult deleteBucketResult = (DeleteBucketResult) result;
    }

    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Object Operations

Uploading an Object

Last updated：2024-06-25 10:53:13

Overview

This document provides an overview of APIs and SDK sample codes related to uploading and replicating objects.

Simple operations

API	Operation	Description
PUT Object	Uploading an object	Uploads an object to a bucket.

Multipart operations

API	Operation	Description
List Multipart Uploads	Querying multipart uploads	Queries in-progress multipart uploads.
Initiate Multipart Upload	Initializing a multipart upload operation	Initializes a multipart upload operation.
Upload Part	Uploading parts	Uploads an object in multiple parts.
List Parts	Querying uploaded parts	Queries the uploaded parts of a multipart upload.
Complete Multipart Upload	Completing a multipart upload	Completes the multipart upload of a file.
Abort Multipart Upload	Aborting a multipart upload	Aborts a multipart upload and deletes the uploaded parts.

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

Advanced APIs (Recommended)

Uploading an object

The advanced APIs encapsulate the simple upload and multipart upload APIs and can intelligently select the upload method based on file size. They also support checkpoint restart for resuming interrupted operations.

Sample 1. Uploading a local file

```
// Initialize TransferConfig. The default configuration is used here. To customize
TransferConfig transferConfig = new TransferConfig.Builder().build();
// Initialize TransferManager
TransferManager transferManager = new TransferManager(cosXmlService,
        transferConfig);

// Bucket name in the format of `BucketName-APPID` (`APPID` is required), which can
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; // Location identifier of the object in the bucke
String srcPath = new File(context.getCacheDir(), "exampleobject")
        .toString(); // Absolute path of the local file
// If there is an uploadId for the initialized multipart upload, assign the value o
String uploadId = null;

// Upload the file
COSXMLUploadTask cosxmlUploadTask = transferManager.upload(bucket, cosPath,
        srcPath, uploadId);

// Set the callback for initializing multipart upload (supported starting from v5.9
cosxmlUploadTask.setInitMultipleUploadListener(new InitMultipleUploadListener() {
    @Override
    public void onSuccess(InitiateMultipartUpload initiateMultipartUpload) {
        // The `uploadId` used for next upload
        String uploadId = initiateMultipartUpload.uploadId;
    }
});
// Set the upload progress callback
cosxmlUploadTask.setCosXmlProgressListener(new CosXmlProgressListener() {
    @Override
    public void onProgress(long complete, long target) {
        // todo Do something to update progress...
    }
});
// Set the response callback
cosxmlUploadTask.setCosXmlResultListener(new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        COSXMLUploadTask.COSXMLUploadTaskResult uploadResult =
                (COSXMLUploadTask.COSXMLUploadTaskResult) result;
    }
    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest request,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
// Set the job status callback to view the job progress
cosxmlUploadTask.setTransferStateListener(new TransferStateListener() {
    @Override
    public void onStateChanged(TransferState state) {
        // todo notify transfer state
    }
});
```

Note:

For the complete sample, please visit GitHub.

You can generate a download URL for the uploaded file using the same key. For detailed directions, see Generating Pre-Signed URLs. Please note that for private-read files, the download URL is only valid for a limited period of time.

Sample 2. Uploading binary data

```
TransferConfig transferConfig = new TransferConfig.Builder().build();
TransferManager transferManager = new TransferManager(cosXmlService,
        transferConfig);

// Bucket name in the format of `BucketName-APPID` (`APPID` is required), which can
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; // Location identifier of the object in the bucke

// Upload a byte array
byte[] bytes = "this is a test".getBytes(Charset.forName("UTF-8"));
COSXMLUploadTask cosxmlUploadTask = transferManager.upload(bucket, cosPath,
        bytes);

// Set the response callback
cosxmlUploadTask.setCosXmlResultListener(new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        COSXMLUploadTask.COSXMLUploadTaskResult uploadResult =
                  (COSXMLUploadTask.COSXMLUploadTaskResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest request,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For the complete sample, please visit GitHub.

You can generate a download URL for the uploaded file using the same key. For detailed directions, see Generating Pre-Signed URLs. Please note that for private-read files, the download URL is only valid for a limited period of time.

Sample 3. Stream upload

```
TransferConfig transferConfig = new TransferConfig.Builder().build();
TransferManager transferManager = new TransferManager(cosXmlService,
        transferConfig);

// Bucket name in the format of `BucketName-APPID` (`APPID` is required), which can
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; // Location identifier of the object in the bucke

// Stream upload
InputStream inputStream =
        new ByteArrayInputStream("this is a test".getBytes(Charset.forName(
                "UTF-8")));
COSXMLUploadTask cosxmlUploadTask = transferManager.upload(bucket, cosPath,
        inputStream);

// Set the response callback
cosxmlUploadTask.setCosXmlResultListener(new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        COSXMLUploadTask.COSXMLUploadTaskResult uploadResult =
                (COSXMLUploadTask.COSXMLUploadTaskResult) result;
    }
    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest request,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For the complete sample, please visit GitHub.

You can generate a download URL for the uploaded file using the same key. For detailed directions, see Generating Pre-Signed URLs. Please note that for private-read files, the download URL is only valid for a limited period of time.

Sample 4. Uploading using a URI

```
TransferConfig transferConfig = new TransferConfig.Builder().build();
TransferManager transferManager = new TransferManager(cosXmlService,
        transferConfig);

// Bucket name in the format of `BucketName-APPID` (`APPID` is required), which can
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; // Location identifier of the object in the bucke

// URI of the file
Uri uri = Uri.parse("exampleObject");
// If there is an `uploadId` for an initialized multipart upload, assign the value
String uploadId = null;
COSXMLUploadTask cosxmlUploadTask = transferManager.upload(bucket, cosPath,
        uri, uploadId);

// Set the response callback
cosxmlUploadTask.setCosXmlResultListener(new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        COSXMLUploadTask.COSXMLUploadTaskResult uploadResult =
                (COSXMLUploadTask.COSXMLUploadTaskResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest request,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For the complete sample, please visit GitHub.

You can generate a download URL for the uploaded file using the same key. For detailed directions, see Generating Pre-Signed URLs. Please note that for private-read files, the download URL is only valid for a limited period of time.

Sample 5. Setting the threshold for smart multipart upload

```
// By default, TransferManager automatically executes multipart upload for files wh
TransferConfig transferConfig = new TransferConfig.Builder()
        .setDivisionForUpload(2 * 1024 * 1024) // Set multipart upload for files wh
        .build();

TransferManager transferManager = new TransferManager(cosXmlService,
        transferConfig);
```

Note:

For the complete sample, please visit GitHub.

Sample 6. Suspending, resuming, and canceling an upload

To suspend an upload, use the code below:

```
// If the final Complete Multipart Upload request has been initiated, the suspensio
boolean pauseSuccess = cosxmlUploadTask.pauseSafely();
```

To resume a suspended download, use the code below:

```
// A suspended upload can be resumed.
if (pauseSuccess) {
    cosxmlUploadTask.resume();
}
```

To cancel an upload, use the code below:

```
cosxmlUploadTask.cancel();
```

Note:

For the complete sample, please visit GitHub.

Sample 7. Uploading multiple objects

```
// Absolute paths to the local files
File[] files = new File(context.getCacheDir(), "exampleDirectory").listFiles();

// Initiate a batch upload
for (File file : files) {
    // If there is an uploadId for the initialized multipart upload, assign the val
    String uploadId = null;

    // Upload the file
    COSXMLUploadTask cosxmlUploadTask = transferManager.upload(bucket, cosPath,
            file.getAbsolutePath(), uploadId);

    // Set the response callback
    cosxmlUploadTask.setCosXmlResultListener(new CosXmlResultListener() {
        @Override
        public void onSuccess(CosXmlRequest request, CosXmlResult result) {
            COSXMLUploadTask.COSXMLUploadTaskResult uploadResult =
                    (COSXMLUploadTask.COSXMLUploadTaskResult) result;
        }

        // If you use the Kotlin language to call this, please note that the except
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest request,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
            if (clientException != null) {
                clientException.printStackTrace();
            } else {
                serviceException.printStackTrace();
            }
        }
    });
}
```

Note:

For the complete sample, please visit GitHub.

Sample 8. Creating a directory

```
// Bucket name in the format of `BucketName-APPID` (`APPID` is required), which can
String bucket = "examplebucket-1250000000";
// The location identifier of a directory in a bucket (i.e., the object key), which
String cosPath = "exampleobject/";
PutObjectRequest putObjectRequest = new PutObjectRequest(bucket, cosPath, new byte[
cosXmlService.putObjectAsync(putObjectRequest, new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        PutObjectResult putObjectResult =
                (PutObjectResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest request,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For the complete sample, please visit GitHub.

You can generate a download URL for the uploaded file using the same key. For detailed directions, see Generating Pre-Signed URLs. Please note that for private-read files, the download URL is only valid for a limited period of time.

Sample 9. Setting a low-priority task

```
// Bucket name in the format of `BucketName-APPID` (`APPID` is required), which can
String bucket = "examplebucket-1250000000";
// The location identifier of a directory in a bucket (i.e., the object key), which
String cosPath = "exampleobject/";
String srcPath = new File(context.getCacheDir(), "exampleobject")
        .toString(); // Absolute path of the local file
PutObjectRequest putObjectRequest= new PutObjectRequest(bucket, cosPath, srcPath);
putObjectRequest.setPriorityLow(); // Set the priority of the task to low.
final COSXMLUploadTask uploadTask = transferManager.upload(putObjectRequest, null);
uploadTask.setCosXmlResultListener(new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {}
    @Override
    public void onFail(CosXmlRequest request, CosXmlClientException clientException
});
```

Note:

For the complete sample, please visit GitHub.

Simple Operations

Uploading an object using simple upload

Feature description

This API (PUT Object) is used to upload an object smaller than 5 GB to a specified bucket. To call this API, you need to have permission to write to the bucket. If the object size is larger than 5 GB, please use Multipart Upload or Advanced APIs for the upload.

Note:

The key (filename) cannot end with `/` ; otherwise, it will be identified as a folder.

Each root account (`APPID`) can have up to 1,000 bucket ACLs and an unlimited number of object ACLs. Do not configure ACLs for an object during upload if you don’t need to control access to it. The object will inherit the permissions of its bucket by default.

Sample code

```
// Bucket name in the format of `BucketName-APPID` (`APPID` is required), which can
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; // The location identifier of the object in the b
String srcPath = new File(context.getCacheDir(), "exampleobject")
        .toString(); // The absolute path of the local file
PutObjectRequest putObjectRequest = new PutObjectRequest(bucket, cosPath,
        srcPath);

putObjectRequest.setProgressListener(new CosXmlProgressListener() {
    @Override
    public void onProgress(long progress, long max) {
        // todo Do something to update progress...
    }
});
cosXmlService.putObjectAsync(putObjectRequest, new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult result) {
        PutObjectResult putObjectResult = (PutObjectResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

You can generate a download URL for the uploaded file using the same key. For detailed directions, see Generating Pre-Signed URLs. Please note that for private-read files, the download URL is only valid for a limited period of time.

Uploading an object using an HTML form

Feature description

This API is used to upload an object using an HTML form.

Sample code

```
String bucket = "examplebucket-1250000000"; // Bucket, formatted as BucketName-APPI
String cosPath = "exampleobject"; // The location identifier of the object in the b
String srcPath = new File(context.getCacheDir(), "exampleobject")
        .toString(); // The absolute path of the local file

PostObjectRequest postObjectRequest = new PostObjectRequest(bucket, cosPath,
        srcPath);

postObjectRequest.setProgressListener(new CosXmlProgressListener() {
    @Override
    public void onProgress(long progress, long max) {
        // todo Do something to update progress...
    }
});
cosXmlService.postObjectAsync(postObjectRequest, new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult result) {
        PutObjectResult putObjectResult = (PutObjectResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Multipart Operations

The multipart upload process is outlined below.

Performing a multipart upload

1. Initialize the multipart upload with `Initiate Multipart Upload` and get the `UploadId` .
2. Use the `UploadId` to upload parts with `Upload Part` or copy parts with `Upload Part Copy`
3. Complete the multipart upload with `Complete Multipart Upload` .

Resuming a multipart upload

1. If you did not record the `UploadId` of the multipart upload, you can query the multipart upload job with `List Multipart Uploads` to get the `UploadId` of the corresponding file.
2. Use the `UploadId` to list the uploaded parts with `List Parts` .
3. Use the `UploadId` to upload the remaining parts with `Upload Part` or copy the remaining parts with `Upload Part Copy` .
4. Complete the multipart upload with `Complete Multipart Upload` .

Aborting a multipart upload

1. If you did not record the `UploadId` of the multipart upload, you can query the multipart upload job with `List Multipart Uploads` to get the `UploadId` of the corresponding file.

2. Abort the multipart upload and delete the uploaded parts with `Abort Multipart Upload` .

Querying multipart uploads

Feature description

This API (`List Multipart Uploads`) is used to query in-progress multipart uploads in a specified bucket.

Sample code

```
// Bucket name in the format of `BucketName-APPID` (`APPID` is required), which can
String bucket = "examplebucket-1250000000";
ListMultiUploadsRequest listMultiUploadsRequest =
        new ListMultiUploadsRequest(bucket);
cosXmlService.listMultiUploadsAsync(listMultiUploadsRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult result) {
        ListMultiUploadsResult listMultiUploadsResult =
                (ListMultiUploadsResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Initializing a multipart upload

Feature description

This API is used to initialize a multipart upload operation and get its `uploadId` .

Sample code

```
// Bucket name in the format of `BucketName-APPID` (`APPID` is required), which can
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; // The location identifier of the object in the b

InitMultipartUploadRequest initMultipartUploadRequest =
        new InitMultipartUploadRequest(bucket, cosPath);
cosXmlService.initMultipartUploadAsync(initMultipartUploadRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult result) {
        // UploadId of the multipart upload
        uploadId = ((InitMultipartUploadResult) result)
                .initMultipartUpload.uploadId;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Uploading parts

This API (`Upload Part`) is used to upload an object in parts.

Sample code

```
// Bucket name in the format of `BucketName-APPID` (`APPID` is required), which can
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; // The location identifier of the object in the b
UploadPartRequest uploadPartRequest = new UploadPartRequest(bucket, cosPath,
        partNumber, srcFile.getPath(), offset, PART_SIZE, uploadId);

uploadPartRequest.setProgressListener(new CosXmlProgressListener() {
    @Override
    public void onProgress(long progress, long max) {
        // todo Do something to update progress...
    }
});

cosXmlService.uploadPartAsync(uploadPartRequest, new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult result) {
        String eTag = ((UploadPartResult) result).eTag;
        eTags.put(partNumber, eTag);
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Querying uploaded parts

Feature description

This API (`List Parts`) is used to query the uploaded parts of a multipart upload.

Sample code

```
// Bucket name in the format of `BucketName-APPID` (`APPID` is required), which can
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; // The location identifier of the object in the b

ListPartsRequest listPartsRequest = new ListPartsRequest(bucket, cosPath,
        uploadId);
cosXmlService.listPartsAsync(listPartsRequest, new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult result) {
        ListParts listParts = ((ListPartsResult) result).listParts;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Completing a multipart upload

Feature description

This API (`Complete Multipart Upload`) is used to complete the multipart upload of a file.

Sample code

```
// Bucket name in the format of `BucketName-APPID` (`APPID` is required), which can
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; // The location identifier of the object in the b

CompleteMultiUploadRequest completeMultiUploadRequest =
        new CompleteMultiUploadRequest(bucket,
        cosPath, uploadId, eTags);
cosXmlService.completeMultiUploadAsync(completeMultiUploadRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult result) {
        CompleteMultiUploadResult completeMultiUploadResult =
                (CompleteMultiUploadResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                         @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Aborting a multipart upload

Feature description

This API (`Abort Multipart Upload`) is used to abort a multipart upload and delete the uploaded parts.

Sample code

```
// Bucket name in the format of `BucketName-APPID` (`APPID` is required), which can
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; // The location identifier of the object in the b

AbortMultiUploadRequest abortMultiUploadRequest =
        new AbortMultiUploadRequest(bucket,
        cosPath, uploadId);
cosXmlService.abortMultiUploadAsync(abortMultiUploadRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult result) {
        AbortMultiUploadResult abortMultiUploadResult =
                (AbortMultiUploadResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                         @Nullable CosXmlClientException clientException,
                         @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more complete samples, visit GitHub.

Downloading Objects

Last updated：2024-06-25 10:53:13

Overview

This document provides an overview of APIs and SDK code samples for downloading an object.

API	Operation	Description
GET Object	Downloading an object	Downloads an object to the local file system

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

Advanced APIs (Recommended)

Downloading an object

The advanced version of the GET Object API uses more encapsulated logic to allow you to suspend, resume (via checkpoint restart), or cancel download requests.

Sample 1. Downloading an object

```
// The advanced download API supports checkpoint restart. Therefore, a HEAD request
// If you are using a temporary key or accessing with a sub-account, ensure that yo

// Initialize TransferConfig. The default configuration is used here. To customize
TransferConfig transferConfig = new TransferConfig.Builder().build();
// Initialize TransferManager
TransferManager transferManager = new TransferManager(cosXmlService,
        transferConfig);

// Bucket name in the format of `BucketName-APPID` (`APPID` is required), which can
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; // Location identifier of the object in the bucke
// Path of the local directory
String savePathDir = context.getExternalCacheDir().toString();
// File name saved locally. If not specified (null), it will be the same as the COS
String savedFileName = "exampleobject";
```

```
Context applicationContext = context.getApplicationContext(); // application
// context
COSXMLDownloadTask cosxmlDownloadTask =
        transferManager.download(applicationContext,
                bucket, cosPath, savePathDir, savedFileName);

// Set the download progress callback
cosxmlDownloadTask.setCosXmlProgressListener(new CosXmlProgressListener() {
    @Override
    public void onProgress(long complete, long target) {
        // todo Do something to update progress...
    }
});
// Set the response callback
cosxmlDownloadTask.setCosXmlResultListener(new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        COSXMLDownloadTask.COSXMLDownloadTaskResult downloadTaskResult =
                (COSXMLDownloadTask.COSXMLDownloadTaskResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest request,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
// Set the job status callback to view the job progress
cosxmlDownloadTask.setTransferStateListener(new TransferStateListener() {
    @Override
    public void onStateChanged(TransferState state) {
        // todo notify transfer state
    }
});
```

Note:

For more samples, please visit GitHub.

Sample 2. Suspending, resuming, or cancelling a download

To suspend a download, use the code below:

```
cosxmlDownloadTask.pause();
```

To resume a suspended download, use the code below:

```
cosxmlDownloadTask.resume();
```

To cancel a download, use the code below:

```
cosxmlDownloadTask.cancel();
```

Note:

For more samples, please visit GitHub.

Sample 3. Setting checkpoint restart for download

```
// Initialize TransferConfig. The default configuration is used here. To customize
// TransferManager supports checkpoint restart for download. You only need to ensur
// Then the SDK will resume the download from where interrupted.
TransferConfig transferConfig = new TransferConfig.Builder().build();
// Initialize TransferManager
TransferManager transferManager = new TransferManager(cosXmlService,
        transferConfig);
// Bucket name in the format of `BucketName-APPID` (`APPID` is required), which can
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; // Location identifier of the object in the bucke
// Path of the local directory
String savePathDir = context.getExternalCacheDir().toString();
// File name saved locally. If not specified (null), it will be the same as the COS
String savedFileName = "exampleobject";

GetObjectRequest getObjectRequest = new GetObjectRequest(bucket, cosPath, savePathD

Context applicationContext = context.getApplicationContext(); // application
// context
COSXMLDownloadTask cosxmlDownloadTask =
        transferManager.download(applicationContext, getObjectRequest);
```

Note:

For more samples, please visit GitHub.

Sample 4. Batch download

```
// The location of the object in the bucket, i.e., the object key
String[] cosPaths = new String[] {
        "exampleobject1",
        "exampleobject2",
        "exampleobject3",
};

for (String cosPath : cosPaths) {

    COSXMLDownloadTask cosxmlDownloadTask =
            transferManager.download(applicationContext,
                    bucket, cosPath, savePathDir, savedFileName);
    // Set the response callback
    cosxmlDownloadTask.setCosXmlResultListener(new CosXmlResultListener() {
        @Override
        public void onSuccess(CosXmlRequest request, CosXmlResult result) {
            COSXMLDownloadTask.COSXMLDownloadTaskResult downloadResult =
                    (COSXMLDownloadTask.COSXMLDownloadTaskResult) result;
        }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest request,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
            if (clientException != null) {
                clientException.printStackTrace();
            } else {
                serviceException.printStackTrace();
            }
        }
    });
}
```

Note:

For more samples, please visit GitHub.

Sample 5. Creating a directory

```
boolean isTruncated = true;
String marker = null;
try {
    while (isTruncated) {
        GetBucketRequest getBucketRequest = new GetBucketRequest(region, bucket, di
        // Configure pagination
        getBucketRequest.setMarker(marker);
        // Configure not to query subdirectories
        getBucketRequest.setDelimiter("/");
        GetBucketResult getBucketResult = cosXmlService.getBucket(getBucketRequest)
        // Batch download
        for (ListBucket.Contents content : getBucketResult.listBucket.contentsList)
            GetObjectRequest getObjectRequest = new GetObjectRequest(bucket, conten
            transferManager.download(context,getObjectRequest);
        }
        isTruncated = getBucketResult.listBucket.isTruncated;
        marker = getBucketResult.listBucket.nextMarker;
    }
} catch (CosXmlServiceException serviceException) {
    serviceException.printStackTrace();
} catch (CosXmlClientException clientException) {
    clientException.printStackTrace();
}
```

Note:

For more samples, please visit GitHub.

Sample 6. Downloading files anonymously (downloading public files without passing in the key)

```
// Initialize TransferConfig. The default configuration is used here. To customize
TransferConfig transferConfig = new TransferConfig.Builder().build();
// Initialize TransferManager
CosXmlServiceConfig cosXmlServiceConfig = new CosXmlServiceConfig.Builder()
        .setRegion("ap-guangzhou")
        .builder();
// The `CosXmlService` generated by anonymous download does not require passing in
CosXmlService cosXmlService = new CosXmlService(context, cosXmlServiceConfig);
TransferManager transferManager = new TransferManager(cosXmlService, transferConfig
// Bucket name in the format of `BucketName-APPID` (`APPID` is required), which can
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; // Location identifier of the object in the bucke
// Path of the local directory
String savePathDir = context.getExternalCacheDir().toString();
// The filename saved locally. If not specified (null), it will be the same as the
String savedFileName = "exampleobject";

GetObjectRequest getObjectRequest = new GetObjectRequest(bucket, cosPath, savePathD

Context applicationContext = context.getApplicationContext(); // application
// context
COSXMLDownloadTask cosxmlDownloadTask =
        transferManager.download(applicationContext, getObjectRequest);
```

Note:

For more samples, please visit GitHub.

Simple Operations

Downloading an object

Feature description

This API is used to download an object to the local file system.

Sample code

```
String bucket = "examplebucket-1250000000"; // Bucket, formatted as BucketName-APPI
String cosPath = "exampleobject"; // The location identifier of the object in the b
String savePath = context.getExternalCacheDir().toString(); // Local path

GetObjectRequest getObjectRequest = new GetObjectRequest(bucket, cosPath,
        savePath);
getObjectRequest.setProgressListener(new CosXmlProgressListener() {
    @Override
    public void onProgress(long progress, long max) {
        // todo Do something to update progress...
    }
});

cosXmlService.getObjectAsync(getObjectRequest, new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest cosXmlRequest,
                          CosXmlResult cosXmlResult) {
        GetObjectResult getObjectResult = (GetObjectResult) cosXmlResult;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Copying and Moving Objects

Last updated：2024-06-25 10:53:13

Overview

This document provides an overview of APIs and SDK code samples related to object copy and movement.

Simple operations

API	Operation	Description
PUT Object - Copy	Copying an object (modifying object attributes)	Copies a file to a destination path.
DELETE Object	Deleting an object	Deletes a specified object from a bucket.

Multipart operations

API	Operation	Description
List Multipart Uploads	Querying multipart uploads/copy	Queries in-progress multipart uploads/copy.
Initiate Multipart Upload	Initializing a multipart upload/copy operation	Initializes a multipart upload/copy operation.
Upload Part - Copy	Copying a part	Copies an object as a part.
List Parts	Querying uploaded/copied parts	Queries the uploaded/copied parts of a multipart operation.
Complete Multipart Upload	Completing a multipart upload/copy	Completes the multipart upload/copy of a file.
Abort Multipart Upload	Aborting a multipart upload/copy	Aborts a multipart operation and deletes the uploaded/copied parts.

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

Copying objects

The advanced APIs encapsulate async requests for the simple copy and multipart copy APIs and support pausing, resuming, and canceling copy requests.

Sample code

```
// Initialize TransferConfig. The default configuration is used here. To customize
TransferConfig transferConfig = new TransferConfig.Builder().build();
// Initialize TransferManager
TransferManager transferManager = new TransferManager(cosXmlService,
        transferConfig);

String sourceAppid = "1250000000"; // Account APPID
String sourceBucket = "sourcebucket-1250000000"; // Bucket of the source object
String sourceRegion = "COS_REGION"; // Region where the bucket of the source object
String sourceCosPath = "sourceObject"; // Key of the source object
// Construct source object attributes
CopyObjectRequest.CopySourceStruct copySourceStruct =
        new CopyObjectRequest.CopySourceStruct(
                sourceAppid, sourceBucket, sourceRegion, sourceCosPath);
// Destination bucket
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
// Destination object
String cosPath = "exampleobject"; // The location identifier of the object in the b

// Copy the object
COSXMLCopyTask cosxmlCopyTask = transferManager.copy(bucket, cosPath,
        copySourceStruct);

// Set the response callback
cosxmlCopyTask.setCosXmlResultListener(new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        COSXMLCopyTask.COSXMLCopyTaskResult copyResult =
                (COSXMLCopyTask.COSXMLCopyTaskResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest request,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
                serviceException.printStackTrace();
            }
        }
});
// Set the job status callback to view the job progress
cosxmlCopyTask.setTransferStateListener(new TransferStateListener() {
    @Override
    public void onStateChanged(TransferState state) {
        // todo notify transfer state
    }
});
```

Moving an object

Object movement involves copying the source object to the target location and deleting the source object.
Since COS uses the bucket name (`Bucket`) and object key (`ObjectKey`) to identify objects, moving an object will change the object identifier. Currently, COS's Android SDK does not provide a standalone API to change object identifiers. However, you can still move the object with a combination of basic operations (**object copy** and **object delete**).
For example, if you want to move the `picture.jpg` object to the "doc" directory that is in the same bucket (`mybucket-1250000000`), you can copy the `picture.jpg` to the "doc" directory (making the object key `doc/picture.jpg`) and then delete the source object.
Likewise, if you need to move `picture.jpg` in the `mybucket-1250000000` bucket to another bucket `myanothorbucket-1250000000` , you can copy the object to the `myanothorbucket-1250000000` bucket first and then delete the source object.

Sample code

```
final String sourceAppid = "1250000000"; // Account appid
final String sourceBucket = "sourcebucket-1250000000"; // Bucket of the source obje
final String sourceRegion = "COS_REGION"; // Region where the bucket of the source
final String sourceKey = "sourceObject"; // Key of the source object
// Construct source object attributes
CopyObjectRequest.CopySourceStruct copySource = new CopyObjectRequest.CopySourceStr
        sourceRegion, sourceKey);

String bucket = "examplebucket-1250000000"; // Destination bucket in the format of
String key = "exampleobject"; // Object key of the destination bucket

// copy(String bucket, String cosPath, CopyObjectRequest.CopySourceStruct copySourc
COSXMLCopyTask copyTask = transferManager.copy(bucket, key, copySource);
copyTask.setCosXmlResultListener(new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        try {
            // Delete the file after it is successfully copied.
            DeleteObjectRequest deleteObjectRequest = new DeleteObjectRequest(sourc
            DeleteObjectResult deleteResult = cosXmlService.deleteObject(deleteObje
        } catch (CosXmlClientException e) {
            e.printStackTrace();
        } catch (CosXmlServiceException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void onFail(CosXmlRequest request, CosXmlClientException exception, CosX

    }
});
```

Note:

For more samples, please visit GitHub.

Copying an object (modifying object attributes)

This API (PUT Object-Copy) is used to copy an object to a destination path.

Sample 1. Copying an object with its attributes preserved

```
String sourceAppid = "1250000000"; // Account APPID
String sourceBucket = "sourcebucket-1250000000"; // Bucket of the source object
String sourceRegion = "COS_REGION"; // Region where the bucket of the source object
String sourceCosPath = "sourceObject"; // Key of the source object
// Construct the source object attributes
CopyObjectRequest.CopySourceStruct copySourceStruct =
        new CopyObjectRequest.CopySourceStruct(
        sourceAppid, sourceBucket, sourceRegion, sourceCosPath);

// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; // The location identifier of the object in the b
CopyObjectRequest copyObjectRequest = new CopyObjectRequest(bucket, cosPath,
        copySourceStruct);

cosXmlService.copyObjectAsync(copyObjectRequest, new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        CopyObjectResult copyObjectResult = (CopyObjectResult) result;
    }
    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Sample 2. Copying an object while replacing its attributes

```
String sourceAppid = "1250000000"; // Account APPID
String sourceBucket = "sourcebucket-1250000000"; // Bucket of the source object
String sourceRegion = "COS_REGION"; // Region where the bucket of the source object
String sourceCosPath = "sourceObject"; // Key of the source object
// Construct the source object attributes
CopyObjectRequest.CopySourceStruct copySourceStruct =
        new CopyObjectRequest.CopySourceStruct(
        sourceAppid, sourceBucket, sourceRegion, sourceCosPath);

// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; // The location identifier of the object in the b
CopyObjectRequest copyObjectRequest = new CopyObjectRequest(bucket, cosPath,
        copySourceStruct);
copyObjectRequest.setCopyMetaDataDirective(MetaDataDirective.REPLACED);
copyObjectRequest.setXCOSMeta("x-cos-metadata-oldKey", "newValue");

cosXmlService.copyObjectAsync(copyObjectRequest, new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        CopyObjectResult copyObjectResult = (CopyObjectResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Sample 3. Modifying object metadata

```
String appId = "1250000000"; // Account APPID
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
String region = "COS_REGION"; // Region where the bucket of the source object resid
String cosPath = "exampleobject"; // The location identifier of the object in the b
// Construct the source object attributes
CopyObjectRequest.CopySourceStruct copySourceStruct =
        new CopyObjectRequest.CopySourceStruct(
        appId, bucket, region, cosPath);

CopyObjectRequest copyObjectRequest = new CopyObjectRequest(bucket, cosPath,
        copySourceStruct);
copyObjectRequest.setCopyMetaDataDirective(MetaDataDirective.REPLACED);
// Replace metadata
copyObjectRequest.setXCOSMeta("x-cos-metadata-oldKey", "newValue");

cosXmlService.copyObjectAsync(copyObjectRequest, new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        CopyObjectResult copyObjectResult = (CopyObjectResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                        @Nullable CosXmlClientException clientException,
                        @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Sample 4. Modifying the storage class of an object

```
String appId = "1250000000"; // Account APPID
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
String region = "COS_REGION"; // Region where the bucket of the source object resid
String cosPath = "exampleobject"; // The location identifier of the object in the b
// Construct the source object attributes
CopyObjectRequest.CopySourceStruct copySourceStruct =
        new CopyObjectRequest.CopySourceStruct(
        appId, bucket, region, cosPath);

CopyObjectRequest copyObjectRequest = new CopyObjectRequest(bucket, cosPath,
        copySourceStruct);
// Set the storage class to STANDARD_IA
copyObjectRequest.setCosStorageClass(COSStorageClass.STANDARD_IA);

cosXmlService.copyObjectAsync(copyObjectRequest, new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        CopyObjectResult copyObjectResult = (CopyObjectResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Listing Objects

Last updated：2024-06-25 10:53:13

Overview

This document provides an overview of APIs and SDK code samples related to listing objects.

API	Operation	Description
GET Bucket (List Objects)	Querying an object list	Queries some or all objects in a bucket
GET Bucket Object Versions	Querying objects and their version history	Queries some or all the objects in a bucket and their version history.

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

Querying an Object List

Description

This API is used to query some or all the objects in a bucket.

Sample 1. Getting the first page of data

```
String bucketName = "examplebucket-1250000000";  // Format: BucketName-APPID;
final GetBucketRequest getBucketRequest = new GetBucketRequest(bucketName);

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

// The maximum number of entries returned at a time; the default value is 1,000
getBucketRequest.setMaxKeys(100);

cosXmlService.getBucketAsync(getBucketRequest, new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        GetBucketResult getBucketResult = (GetBucketResult) result;
        if (getBucketResult.listBucket.isTruncated) {
            // The data is truncated, and the next page of data needs to be pulled
            prevPageResult = getBucketResult;
        }
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Sample 2. Requesting the next page of data

```
String bucketName = "examplebucket-1250000000";  // Format: BucketName-APPID;


GetBucketRequest getBucketRequest = new GetBucketRequest(bucketName);

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

// `prevPageResult` is the result returned on the previous page, where `nextMarker`
String nextMarker = prevPageResult.listBucket.nextMarker;
getBucketRequest.setMarker(nextMarker);

// The maximum number of entries returned at a time; the default value is 1,000
getBucketRequest.setMaxKeys(100);

cosXmlService.getBucketAsync(getBucketRequest, new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        GetBucketResult getBucketResult = (GetBucketResult) result;
        if (getBucketResult.listBucket.isTruncated) {
            // The data is truncated, and the next page of data needs to be pulled
        }
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Sample 3. Getting an object list and subdirectories

```
String bucketName = "examplebucket-1250000000";  // Format: BucketName-APPID;
GetBucketRequest getBucketRequest = new GetBucketRequest(bucketName);

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

// The maximum number of entries returned at a time; the default value is 1,000
getBucketRequest.setMaxKeys(100);

// The delimiter is a symbol. If the prefix exists,
// identical paths between the prefix and delimiter will be grouped as together and
// and then all common prefixes are listed. If there is no prefix, the listing star
getBucketRequest.setDelimiter("/");

cosXmlService.getBucketAsync(getBucketRequest, new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        GetBucketResult getBucketResult = (GetBucketResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
```

```
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Querying an Object Version List

Description

This API is used to query some or all objects in a versioning-enabled bucket.

Sample 1. Getting the object version list’s first page of data

```
String bucketName = "examplebucket-1250000000";  // Format: BucketName-APPID;
final GetBucketObjectVersionsRequest getBucketRequest =
        new GetBucketObjectVersionsRequest(bucketName);

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

// The maximum number of entries returned at a time; the default value is 1,000
getBucketRequest.setMaxKeys(100);

cosXmlService.getBucketObjectVersionsAsync(getBucketRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        GetBucketObjectVersionsResult getBucketResult =
                (GetBucketObjectVersionsResult) result;
        if (getBucketResult.listVersionResult.isTruncated) {
            // The data is truncated, and the next page of data needs to be pulled
            prevPageResult = getBucketResult;
        }
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                         @Nullable CosXmlClientException clientException,
                         @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Sample 2. Getting the object version list's next page of data

```
String bucketName = "examplebucket-1250000000";  // Format: BucketName-APPID;
final GetBucketObjectVersionsRequest getBucketRequest =
        new GetBucketObjectVersionsRequest(bucketName);

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

// The maximum number of entries returned at a time; the default value is 1,000
getBucketRequest.setMaxKeys(100);

// `prevPageResult` is the result returned on the previous page, where `nextMarker`
// indicate the starting point of the next page
getBucketRequest.setKeyMarker(prevPageResult.listVersionResult
        .nextKeyMarker);
getBucketRequest.setVersionIdMarker(prevPageResult.listVersionResult
        .nextVersionIdMarker);

cosXmlService.getBucketObjectVersionsAsync(getBucketRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        GetBucketObjectVersionsResult getBucketResult =
                (GetBucketObjectVersionsResult) result;
        if (getBucketResult.listVersionResult.isTruncated) {
            // The data is truncated, and the next page of data needs to be pulled
            prevPageResult = getBucketResult;
        }
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Deleting Objects

Last updated：2024-06-25 10:53:13

Overview

This document provides an overview of APIs and SDK code samples related to object deletion.

API	Operation	Description
DELETE Object	Deleting an object	Deletes an object from a bucket.
DELETE Multiple Objects	Deleting multiple objects	Deletes multiple objects from a bucket in a single request

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

Deleting a Single Object

Description

This API (`DELETE Object`) is used to delete a specified object.

Sample code

```
String bucket = "examplebucket-1250000000"; // Bucket, formatted as BucketName-APPI
String cosPath = "exampleobject"; // The location identifier of the object in the b

DeleteObjectRequest deleteObjectRequest = new DeleteObjectRequest(bucket,
        cosPath);
cosXmlService.deleteObjectAsync(deleteObjectRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult result) {
        DeleteObjectResult deleteObjectResult = (DeleteObjectResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Deleting Multiple Objects

Description

This API is used to delete multiple objects in a single request.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
List<String> objectList = new ArrayList<String>();
objectList.add("exampleobject1"); // The location identifier of the object in the b
objectList.add("exampleobject2"); // The location identifier of the object in the b

DeleteMultiObjectRequest deleteMultiObjectRequest =
        new DeleteMultiObjectRequest(bucket, objectList);
// In quiet mode, only information on objects that failed to be deleted will be ret
deleteMultiObjectRequest.setQuiet(true);
cosXmlService.deleteMultiObjectAsync(deleteMultiObjectRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult result) {
        DeleteMultiObjectResult deleteMultiObjectResult =
                (DeleteMultiObjectResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Deleting a Directory

Description

COS uses slashes (/) as the delimiter to show directories in order to achieve the effect of a file system. Therefore, if you want to delete a directory in COS, you need to delete objects that are prefixed with a specified value. For example, the directory `prefix/` is actually all objects prefixed with `prefix/` . Therefore, you can delete all objects prefixed with `prefix/` to delete the `prefix/` directory.

Currently, COS's Android SDK did not provide an API to perform this operation. However, you can still do it using a combination of basic operations.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
String prefix = "folder1/"; // Specify a prefix.

GetBucketRequest getBucketRequest = new GetBucketRequest(bucket);
getBucketRequest.setPrefix(prefix);

// "prefix" indicates the directory to delete.
getBucketRequest.setPrefix(prefix);
// Set the maximum number of traversed objects (up to 1,000 per listobject request)
getBucketRequest.setMaxKeys(1000);
GetBucketResult getBucketResult = null;

do {
    try{
        getBucketResult = cosXmlService.getBucket(getBucketRequest);
        List<ListBucket.Contents> contents = getBucketResult.listBucket.contentsLis
        DeleteMultiObjectRequest deleteMultiObjectRequest = new DeleteMultiObjectRe
```

```
            for (ListBucket.Contents content : contents) {
                deleteMultiObjectRequest.setObjectList(content.key);
            }
            cosXmlService.deleteMultiObject(deleteMultiObjectRequest);
            getBucketRequest.setMarker(getBucketResult.listBucket.nextMarker);
        } catch (CosXmlClientException e) {
            e.printStackTrace();
            return;
        } catch (CosXmlServiceException e) {
            e.printStackTrace();
            return;
        }
    } while (getBucketResult.listBucket.isTruncated);
```

Note:

For more samples, please visit GitHub.

Restoring Archived Objects

Last updated：2024-06-25 10:53:13

Overview

This document provides an overview of APIs and SDK code samples related to restoring an archived object.

API	Operation	Description
POST Object restore	Restoring an archived object	Restores an archived object for access.

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

Restoring an Archived Object

Description

This API (`POST Object restore`) is used to restore an archived object for access.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; // The location identifier of the object in the b
RestoreRequest restoreRequest = new RestoreRequest(bucket, cosPath);
restoreRequest.setExpireDays(5); // Retain for 5 days
restoreRequest.setTier(RestoreConfigure.Tier.Standard); // Standard restoration mod

cosXmlService.restoreObjectAsync(restoreRequest, new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        RestoreResult restoreResult = (RestoreResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                         @Nullable CosXmlClientException clientException,
                         @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Querying Object Metadata

Last updated：2024-06-25 10:53:13

Overview

This document provides an overview of APIs and SDK code samples related to querying object metadata.

API	Operation	Description
HEAD Object	Querying object metadata	Queries the metadata of an object.

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

Querying Object Metadata

Description

This API is used to query the metadata of an object.

Sample code

```
String bucket = "examplebucket-1250000000"; // Bucket, formatted as BucketName-APPI
String cosPath = "exampleobject"; // The location identifier of the object in the b
HeadObjectRequest headObjectRequest = new HeadObjectRequest(bucket, cosPath);
cosXmlService.headObjectAsync(headObjectRequest, new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        HeadObjectResult headObjectResult = (HeadObjectResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
```

```
          } else {
              serviceException.printStackTrace();
          }
      }
  });
```

Note:

For more samples, please visit GitHub.

Generating Pre-Signed URLs

Last updated：2024-06-25 10:53:13

Overview

This document provides an overview of SDK code samples related to generating pre-signed object URLs.
For details about how to use a pre-signed URL for uploads, see Upload via Pre-Signed URL. For details about how to use a pre-signed URL for downloads, see Download via Pre-Signed URL.

Note:

You are advised to use a temporary key to generate pre-signed URLs for the security of your requests such as uploads and downloads. When you apply for a temporary key, follow the Principle of Least Privilege to avoid leaking resources besides your buckets and objects.
If you need to use a permanent key to generate a pre-signed URL, you are advised to limit the permission of the permanent key to uploads and downloads only to avoid risks.

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

Generating a Pre-Signed Object URL

Sample code 1. Generating a pre-signed upload URL

```
try {
     // Bucket name
    String bucket = "examplebucket-1250000000";
    // Object key, the unique location ID of an object in a bucket. For more inform
    // Note: The key does not need to be encoded.
    String cosPath = "exampleobject";
    // HTTP request method
    String method = "PUT";
    PresignedUrlRequest presignedUrlRequest = new PresignedUrlRequest(bucket
            , cosPath) {
        @Override
        public RequestBodySerializer getRequestBody()
                throws CosXmlClientException {
            // Used to calculate a pre-signed URL for requests like `PUT` that requ
            return RequestBodySerializer.string("text/plain",
                    "this is test");
        }
    };
    presignedUrlRequest.setRequestMethod(method);
    // Set the signature validity period to be 60s. Note that here is the signature
    presignedUrlRequest.setSignKeyTime(60);
    // Set not to sign `Host`
    presignedUrlRequest.addNoSignHeader("Host");
    String urlWithSign = cosXmlService.getPresignedURL(presignedUrlRequest);
} catch (CosXmlClientException e) {
    e.printStackTrace();
}
```

Note:

For the complete sample, go to GitHub.

Sample code 2. Generating a pre-signed download URL

```
try {
    // Bucket name
    String bucket = "examplebucket-1250000000";
    // Object key, the unique location ID of an object in a bucket. For more inform
    // Note: The key does not need to be encoded.
    String cosPath = "exampleobject";
    // HTTP request method
    String method = "GET";
    PresignedUrlRequest presignedUrlRequest = new PresignedUrlRequest(bucket
            , cosPath);
    presignedUrlRequest.setRequestMethod(method);

    // Set the signature validity period to be 60s. Note that here is the signature
    presignedUrlRequest.setSignKeyTime(60);
    // Set not to sign `Host`
    presignedUrlRequest.addNoSignHeader("Host");

    String urlWithSign = cosXmlService.getPresignedURL(presignedUrlRequest);

} catch (CosXmlClientException e) {
    e.printStackTrace();
}
```

Note:

For the complete sample, go to GitHub.

Configuring Preflight Requests for Cross-origin Access

Last updated：2024-06-25 10:53:13

Overview

This document provides an overview of APIs and SDK code samples related to CORS preflight requests.

API	Operation	Description
Options Object	Configuring a preflight request for cross-origin access	Sends a preflight request to check whether a real cross-origin access request can be sent

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

Configuring a Preflight Request for Cross-origin Access

Description

This API is used to get the cross-origin access configuration for a preflight request.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; // The location identifier of the object in the b
String origin = "https://cloud.tencent.com";
String accessMethod = "PUT";
OptionObjectRequest optionObjectRequest = new OptionObjectRequest(bucket,
        cosPath, origin,
        accessMethod);
cosXmlService.optionObjectAsync(optionObjectRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult result) {
        OptionObjectResult optionObjectResult = (OptionObjectResult) result;
    }
```

```
        // If you use the Kotlin language to call this, please note that the exception
        // clientException is of type CosXmlClientException? and serviceException is of
        @Override
        public void onFail(CosXmlRequest cosXmlRequest,
                           @Nullable CosXmlClientException clientException,
                           @Nullable CosXmlServiceException serviceException) {
            if (clientException != null) {
                clientException.printStackTrace();
            } else {
                serviceException.printStackTrace();
            }
        }
    });
```

Note:

For more samples, please visit GitHub.

Server-Side Encryption

Last updated：2024-06-25 10:53:13

Overview

This document describes how to enable server-side encryption when uploading objects. There are three types of keys that can be used for server-side encryption:
COS-managed key
KMS-managed key
Customer-provided key

SDK API Reference

For the parameters and method descriptions of all the APIs in the SDK, please see SDK API Reference.

Using server-side encryption with COS-managed encryption keys (SSE-COS) to protect data

Description

With this method, your master key and data are managed by COS. COS can automatically encrypt your data when written into the IDC and automatically decrypt it when accessed. Currently, COS supports AES-256 encryption using a COS master key pair.

Sample code

```
PutObjectRequest putObjectRequest = new PutObjectRequest(bucket, cosPath, srcPath);
// Configure server-side encryption with COS-managed encryption keys (SSE-COS) to p
putObjectRequest.setCOSServerSideEncryption();

// Upload a file
COSXMLUploadTask cosxmlUploadTask = transferManager.upload(putObjectRequest, upload
```

Note:
For the complete sample, go to GitHub.

Using server-side encryption with KMS-managed encryption keys (SSE-KMS) to protect data

Description

SSE-KMS encryption is server-side encryption using keys managed by KMS, a Tencent Cloud security management service. KMS is designed to generate and protect your keys using third-party-certified hardware security modules (HSM). It allows you to easily create and manage keys for use in multiple applications and services, while meeting regulatory and compliance requirements. For information on how to activate KMS service, see Server-side Encryption Overview.

Sample code

```
// Server-side encryption key
String customKey = "customer master key (CMK)";
String encryptContext = "encryption context";
PutObjectRequest putObjectRequest = new PutObjectRequest(bucket, cosPath, srcPath);

// Configure server-side encryption with KMS customer master keys (SSE-KMS) to prot
try {
    putObjectRequest.setCOSServerSideEncryptionWithKMS(customKey, encryptContext);
} catch (CosXmlClientException e) {
    e.printStackTrace();
}
// Upload the files
COSXMLUploadTask cosxmlUploadTask = transferManager.upload(putObjectRequest, upload
```

Note:

For the complete sample, go to GitHub.

Using server-side encryption with customer-provided encryption keys (SSE-C) to protect data

Description

With this method, the encryption key is provided by the customer. To upload an object, COS will apply AES-256 encryption to the data using the customer-provided encryption key pair.

Note:

This type of encryption requires using HTTPS requests.

You need to provide a 32-byte string as the key, a combination of numbers, letters, and characters, with Chinese characters not supported.

If a file was key-encrypted when uploaded, you need to include the same key in your GET (download) or HEAD (query) request for it to succeed.

Sample code

```
// Server-side encryption key
String customKey = "Server-side encryption key";
PutObjectRequest putObjectRequest = new PutObjectRequest(bucket, cosPath, srcPath);
```

```
// Configure server-side encryption with customer-provided encryption keys (SSE-C)
try {
    putObjectRequest.setCOSServerSideEncryptionWithCustomerKey(customKey);
} catch (CosXmlClientException e) {
    e.printStackTrace();
}

// Upload the files
COSXMLUploadTask cosxmlUploadTask = transferManager.upload(putObjectRequest, upload
```

Note:

For the complete sample, go to GitHub.

Single-Connection Bandwidth Limit

Last updated：2024-06-25 10:53:13

Overview

This document describes how to limit the speed on a single URL when calling the upload or download API.

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

Directions

The speed range is **819200 to 838860800** (in bit/s), that is, 100 KB/s to 100 MB/s. If a value is not within this range, 400 will be returned.

Sample 1. Limiting single-URL speed on uploads

```
TransferConfig transferConfig = new TransferConfig.Builder().build();
// Initialize TransferManager
TransferManager transferManager = new TransferManager(cosXmlService,
        transferConfig);

// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; // Location identifier of the object in the bucke
String srcPath = new File(context.getCacheDir(), "exampleobject")
        .toString(); // Absolute path of the local file
// If there is an uploadId for the initialized multipart upload, assign the value o
String uploadId = null;

PutObjectRequest putObjectRequest = new PutObjectRequest(bucket, cosPath, srcPath);
// Set the bandwidth limit for a single request in bit/s. In the example, the limit
putObjectRequest.setTrafficLimit(1024 * 1024 * 8);

// Upload the object
COSXMLUploadTask cosxmlUploadTask = transferManager.upload(putObjectRequest, upload

// Set the upload progress callback
```

```
cosxmlUploadTask.setCosXmlProgressListener(new CosXmlProgressListener() {
    @Override
    public void onProgress(long complete, long target) {
        // todo Do something to update progress...
    }
});
// Set the response callback
cosxmlUploadTask.setCosXmlResultListener(new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        COSXMLUploadTask.COSXMLUploadTaskResult uploadResult =
                (COSXMLUploadTask.COSXMLUploadTaskResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest request,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
// Set the job status callback to view the job progress
cosxmlUploadTask.setTransferStateListener(new TransferStateListener() {
    @Override
    public void onStateChanged(TransferState state) {
        // todo notify transfer state
    }
});
```

Note:

For the complete sample, please visit GitHub.

Sample 2. Limiting single-URL speed on downloads

```
//.cssg-snippet-body-start:[transfer-download-object]
// The advanced download API supports checkpoint restart. Therefore, a HEAD request
// If you are using a temporary key or accessing with a sub-account, ensure that yo

// Initialize TransferConfig. The default configuration is used here. To customize
TransferConfig transferConfig = new TransferConfig.Builder().build();
// Initialize TransferManager
```

```
TransferManager transferManager = new TransferManager(cosXmlService,
        transferConfig);

// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; // Location identifier of the object in the bucke
// Path of the local directory
String savePathDir = context.getExternalCacheDir().toString();
// File name saved locally. If not specified (null), it will be the same as the COS
String savedFileName = "exampleobject";

GetObjectRequest getObjectRequest = new GetObjectRequest(bucket, cosPath, savePathD
// Set the bandwidth limit for a single URL in bit/s. In the example, the limit is
getObjectRequest.setTrafficLimit(1024 * 1024 * 8);

Context applicationContext = context.getApplicationContext(); // application
// context
COSXMLDownloadTask cosxmlDownloadTask =
        transferManager.download(applicationContext, getObjectRequest);

// Set the download progress callback
cosxmlDownloadTask.setCosXmlProgressListener(new CosXmlProgressListener() {
    @Override
    public void onProgress(long complete, long target) {
        // todo Do something to update progress...
    }
});
// Set the response callback
cosxmlDownloadTask.setCosXmlResultListener(new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        COSXMLDownloadTask.COSXMLDownloadTaskResult downloadTaskResult =
                (COSXMLDownloadTask.COSXMLDownloadTaskResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest request,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
// Set the job status callback to view the job progress
cosxmlDownloadTask.setTransferStateListener(new TransferStateListener() {
    @Override
    public void onStateChanged(TransferState state) {
        // todo notify transfer state
    }
});
```

Note:

For more samples, please visit GitHub.

Extracting Object Content

Last updated：2024-06-25 10:53:14

Overview

This document provides an overview of APIs and SDK code samples related to object content extraction.

API	Operation	Description
SELECT Object Content	Extracting object content	Extracts the content of a specified object (in CSV or JSON format)

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

Extracting Object Content

Description

COS Select supports extracting content from objects in the following formats:

CSV: The object’s data records are separated by a specific delimiter.

JSON: either a JSON file or a JSON list

Note:

To use COS Select, you must have the permission on `cos:GetObject` .

CSV and JSON objects need to be encoded in UTF-8.

COS Select supports extracting CSV and JSON objects compressed by gzip or bzip2.

COS Select supports extracting CSV and JSON objects encrypted with SSE-COS.

Sample code

```
String bucket = "examplebucket-1250000000";
// The object must be in JSON or CSV format
String cosPath = "exampleobject";
final String expression = "Select * from COSObject";

SelectObjectContentRequest selectObjectContentRequest = new SelectObjectContentRequ
        bucket, cosPath, expression, true,
        new InputSerialization(CompressionType.NONE, new JSONInput(JSONType.DOCUMEN
        new OutputSerialization(new JSONOutput(","))
);

// Set the result callback which may work repeatedly
selectObjectContentRequest.setSelectObjectContentProgressListener(new SelectObjectC
    @Override
    public void onProcess(SelectObjectContentEvent event) {

    }
});
cosXmlService.selectObjectContentAsync(selectObjectContentRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        SelectObjectContentResult selectObjectContentResult =
                (SelectObjectContentResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Remote Disaster Recovery Versioning

Last updated：2024-06-25 10:53:13

Overview

This document provides an overview of APIs and SDK code samples related to versioning.

API	Operation	Description
PUT Bucket versioning	Setting versioning	Sets versioning for a bucket
GET Bucket versioning	Querying versioning	Queries the versioning information of a bucket

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

Setting versioning

Description

This API is used to set the versioning configuration of a specified bucket. Once enabled, versioning can only be suspended but not disabled.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
PutBucketVersioningRequest putBucketVersioningRequest =
        new PutBucketVersioningRequest(bucket);
// true: enable versioning; false: suspend versioning
putBucketVersioningRequest.setEnableVersion(true);

cosXmlService.putBucketVersionAsync(putBucketVersioningRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        PutBucketVersioningResult putBucketVersioningResult =
                  (PutBucketVersioningResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Querying versioning

Description

This API is used to query the versioning configuration of a specified bucket.

To get the versioning status of a bucket, you need to have read permission for the bucket.

There are three versioning statuses: not enabled, enabled, and suspended.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
GetBucketVersioningRequest getBucketVersioningRequest =
        new GetBucketVersioningRequest(bucket);

cosXmlService.getBucketVersioningAsync(getBucketVersioningRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        GetBucketVersioningResult getBucketVersioningResult =
                (GetBucketVersioningResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Cross-region replication

Last updated：2024-06-25 10:53:13

Overview

This document provides an overview of APIs and SDK code samples related to cross-region replication.

API	Operation	Description
PUT Bucket replication	Setting a cross-region replication rule	Sets a cross-region replication rule for a bucket
GET Bucket replication	Querying a cross-region replication rule	Queries the cross-region replication rule of a bucket
DELETE Bucket replication	Deleting a cross-region replication rule	Deletes the cross-region replication rule from a bucket

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

Setting Cross-region Replication Rules

Description

This API is used to set the cross-region replication rules of a specified bucket.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
PutBucketReplicationRequest putBucketReplicationRequest =
        new PutBucketReplicationRequest(bucket);

string ownerUin = "100000000001"; //Replication initiator identifier: OwnerUin
string subUin = "100000000001"; //Replication initiator identifier: SubUin
putBucketReplicationRequest.setReplicationConfigurationWithRole(ownerUin,
        subUin);
```

```
PutBucketReplicationRequest.RuleStruct ruleStruct =
        new PutBucketReplicationRequest.RuleStruct();
// Identify the name of a specific rule
ruleStruct.id = "replication_01";
//Identify whether to enable the rule. true: enabled; false: disabled
ruleStruct.isEnable = true;
// Destination bucket region
ruleStruct.region = "ap-beijing";
// Destination bucket
ruleStruct.bucket = "destinationbucket-1250000000";
// Prefix matching policy
ruleStruct.prefix = "dir/";
putBucketReplicationRequest.setReplicationConfigurationWithRule(ruleStruct);

cosXmlService.putBucketReplicationAsync(putBucketReplicationRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        PutBucketReplicationResult putBucketReplicationResult =
                (PutBucketReplicationResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Querying Cross-region Replication Rules

Description

This API is used to query the cross-region replication rules of a specified bucket.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
GetBucketReplicationRequest getBucketReplicationRequest =
        new GetBucketReplicationRequest(bucket);

cosXmlService.getBucketReplicationAsync(getBucketReplicationRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        GetBucketReplicationResult getBucketReplicationResult =
                (GetBucketReplicationResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Deleting Cross-region Replication Rules

Description

This API is used to delete the cross-region replication rules of a specified bucket.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
DeleteBucketReplicationRequest deleteBucketReplicationRequest =
        new DeleteBucketReplicationRequest(bucket);
```

```
cosXmlService.deleteBucketReplicationAsync(deleteBucketReplicationRequest,
        new CosXmlResultListener() {
            @Override
            public void onSuccess(CosXmlRequest request, CosXmlResult result) {
                DeleteBucketReplicationResult deleteBucketReplicationResult =
                        (DeleteBucketReplicationResult) result;
            }

            // If you use the Kotlin language to call this, please note that the ex
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
                if (clientException != null) {
                    clientException.printStackTrace();
                } else {
                    serviceException.printStackTrace();
                }
            }
        });
```

Note:

For more samples, please visit GitHub.

Data Management

Lifecycle

Last updated：2024-06-25 10:53:13

Overview

This document provides an overview of APIs and SDK code samples related to lifecycles.

API	Operation	Description
PUT Bucket lifecycle	Setting lifecycle configuration	Sets lifecycle for a bucket
GET Bucket lifecycle	Querying a lifecycle configuration	Queries the lifecycle configuration of a bucket
DELETE Bucket lifecycle	Deleting a lifecycle configuration	Deletes the lifecycle configuration of a bucket

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

Setting a Lifecycle Configuration

Description

This API is used to set the lifecycle configuration of a specified bucket.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
PutBucketLifecycleRequest putBucketLifecycleRequest =
        new PutBucketLifecycleRequest(bucket);

// Specify a lifecycle configuration rule
LifecycleConfiguration.Rule rule = new LifecycleConfiguration.Rule();
rule.id = "Lifecycle ID";
LifecycleConfiguration.Filter filter = new LifecycleConfiguration.Filter();
// Specify the prefix to which the rule applies
filter.prefix = "dir/";
```

```
rule.filter = filter;
// Specify whether to enable the rule
rule.status = "Enabled";
// Specify the number of days after which the object is last modified that the acti
LifecycleConfiguration.Transition transition =
        new LifecycleConfiguration.Transition();
transition.days = 100;
transition.storageClass = COSStorageClass.STANDARD.getStorageClass();
rule.transition = transition;

putBucketLifecycleRequest.setRuleList(rule);

cosXmlService.putBucketLifecycleAsync(putBucketLifecycleRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        PutBucketLifecycleResult putBucketLifecycleResult =
                (PutBucketLifecycleResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Querying a Lifecycle Configuration

Description

This API is used to query the lifecycle management configuration of a bucket.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
GetBucketLifecycleRequest getBucketLifecycleRequest =
        new GetBucketLifecycleRequest(bucket);

cosXmlService.getBucketLifecycleAsync(getBucketLifecycleRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        GetBucketLifecycleResult getBucketLifecycleResult =
                (GetBucketLifecycleResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Deleting a Lifecycle Configuration

Description

This API is used to delete the lifecycle management configuration of a bucket.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
DeleteBucketLifecycleRequest deleteBucketLifecycleRequest =
        new DeleteBucketLifecycleRequest(bucket);

cosXmlService.deleteBucketLifecycleAsync(deleteBucketLifecycleRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        DeleteBucketLifecycleResult deleteBucketLifecycleResult =
                (DeleteBucketLifecycleResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Log Management

Last updated：2024-06-25 10:53:13

Overview

This document provides an overview of APIs and SDK code samples related to logging.

API	Operation	Description
PUT Bucket logging	Setting logging	Enables logging for a source bucket
GET Bucket logging	Querying logging configuration	Queries the logging configuration of a source bucket

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

Setting Logging Configuration

Description

This API is used to enable logging for a source bucket and store the access logs in a specified destination bucket.

Sample code

```
String srcBucket = "examplebucket-1250000000"; //Format: BucketName-APPID
String targetBucket = "examplebucket-1250000000"; //Format: BucketName-APPID
PutBucketLoggingRequest putBucketLoggingRequest =
        new PutBucketLoggingRequest(srcBucket);
// Destination bucket
putBucketLoggingRequest.setTargetBucket(targetBucket);
// Specified location in which to store the logs
putBucketLoggingRequest.setTargetPrefix("dir/");

cosXmlService.putBucketLoggingAsync(putBucketLoggingRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        PutBucketLoggingResult putBucketLoggingResult =
                (PutBucketLoggingResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Querying Logging Configuration

Description

This API is used to query the logging configuration of a specified bucket.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
GetBucketLoggingRequest getBucketLoggingRequest =
        new GetBucketLoggingRequest(bucket);

cosXmlService.getBucketLoggingAsync(getBucketLoggingRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        GetBucketLoggingResult getBucketLoggingResult =
                (GetBucketLoggingResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                            @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Object Tagging

Last updated：2024-06-25 10:53:13

Overview

This document provides an overview of APIs and SDK code samples related to object tagging.

API	Operation	Description
PUT Object tagging	Tagging an object	Tags an uploaded object.
GET Object tagging	Querying object tags	Queries all tags of an object.
DELETE Object tagging	Deleting object tags	Deletes all tags of an object.

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

Tagging an Object

Adding tags when uploading an object

Description

When uploading an object, you can add specific header information to the request to set tags for the object. For example, you can set `x-cos-tagging` to `Key1=Value1&Key2=Value2` . The tag keys and tag values in the set must be URL-encoded.

Sample code

```
// Initialize TransferConfig. The default configuration is used here. To customize
TransferConfig transferConfig = new TransferConfig.Builder().build();
// Initialize TransferManager
TransferManager transferManager = new TransferManager(cosXmlService,
        transferConfig);
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; // Location identifier of the object in the bucke
String srcPath = new File(context.getCacheDir(), "exampleobject")
```

```
        .toString(); // Absolute path of the local file
PutObjectRequest putObjectRequest = new PutObjectRequest(bucket, cosPath, srcPath);
try {
    // Set object tags. The tag keys and tag values in the set must be URL-encoded
    putObjectRequest.setRequestHeaders("x-cos-tagging", "Key1=Value&Key2=Value2", f
} catch (CosXmlClientException e) {
    e.printStackTrace();
}
// If there is an `uploadId` for an initialized multipart upload, assign the value
String uploadId = null;
// Upload the object
COSXMLUploadTask cosxmlUploadTask = transferManager.upload(bucket, cosPath,
        srcPath, uploadId);
// Set the response callback
cosxmlUploadTask.setCosXmlResultListener(new CosXmlResultListener() {

    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        COSXMLUploadTask.COSXMLUploadTaskResult uploadResult =
                (COSXMLUploadTask.COSXMLUploadTaskResult) result;
    }
    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest request,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Adding tags to an existing object

Description

This API is used to set tags for an existing object. It can help you group and manage existing object resources by adding key-value pairs as object tags.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; // Location identifier of the object in the bucke
PutObjectTaggingRequest putObjectTaggingRequest = new PutObjectTaggingRequest(bucke
putObjectTaggingRequest.addTag("key", "value");
try {
    PutObjectTaggingResult putObjectTaggingResult = cosXmlService.putObjectTagging(
} catch (CosXmlClientException clientException) {
    clientException.printStackTrace();
} catch (CosXmlServiceException serviceException) {
    serviceException.printStackTrace();
}
```

Note:

For more samples, please visit GitHub.

Querying Object Tags

Description

This API is used to query the existing tags of a specified object.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; // Location identifier of the object in the bucke
GetObjectTaggingRequest getObjectTaggingRequest = new GetObjectTaggingRequest(bucke
try {
    GetObjectTaggingResult getObjectTaggingResult = cosXmlService.getObjectTagging(
} catch (CosXmlClientException clientException) {
    clientException.printStackTrace();
} catch (CosXmlServiceException serviceException) {
    serviceException.printStackTrace();
}
```

Note:

For more samples, please visit GitHub.

Deleting Object Tags

Description

This API is used to delete the existing tags of a specified object.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; // Location identifier of the object in the bucke
DeleteObjectTaggingRequest deleteObjectTaggingRequest = new DeleteObjectTaggingRequ
try {
    DeleteObjectTaggingResult deleteObjectTaggingResult = cosXmlService.deleteObjec
} catch (CosXmlClientException clientException) {
    clientException.printStackTrace();
} catch (CosXmlServiceException serviceException) {
    serviceException.printStackTrace();
}
```

Note:

For more samples, please visit GitHub.

Bucket Tagging

Last updated：2024-06-25 10:53:13

Overview

This document provides an overview of APIs and SDK code samples related to bucket tagging.

API	Operation	Description
PUT Bucket tagging	Setting bucket tags	Sets tags for an existing bucket
GET Bucket tagging	Querying bucket tags	Queries the existing tags of a bucket
DELETE Bucket tagging	Deleting bucket tags	Deletes the tags of a bucket

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

Setting Bucket Tags

Description

This API is used to set tags for an existing bucket.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
PutBucketTaggingRequest putBucketTaggingRequest =
        new PutBucketTaggingRequest(bucket);
// Set a tag
putBucketTaggingRequest.addTag("key", "value");
putBucketTaggingRequest.addTag("hello", "world");

cosXmlService.putBucketTaggingAsync(putBucketTaggingRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        PutBucketTaggingResult putBucketTaggingResult =
                (PutBucketTaggingResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Querying Bucket Tags

Description

This API is used to query the existing tags of a specified bucket.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
GetBucketTaggingRequest getBucketTaggingRequest =
        new GetBucketTaggingRequest(bucket);

cosXmlService.getBucketTaggingAsync(getBucketTaggingRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        GetBucketTaggingResult getBucketTaggingResult =
                (GetBucketTaggingResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                        @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Deleting Bucket Tags

Description

This API is used to delete the existing tags from a bucket.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
DeleteBucketTaggingRequest deleteBucketTaggingRequest =
        new DeleteBucketTaggingRequest(bucket);

cosXmlService.deleteBucketTaggingAsync(deleteBucketTaggingRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        DeleteBucketTaggingResult getBucketTaggingResult =
                (DeleteBucketTaggingResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                        @Nullable CosXmlClientException clientException,
                        @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
        }
    });
```

Note:

For more samples, please visit GitHub.

Static Website

Last updated：2024-06-25 10:53:13

Overview

This document provides an overview of APIs and SDK code samples related to static website.

API	Operation	Description
PUT Bucket website	Setting a static website configuration	Configures a static website for a bucket
GET Bucket website	Querying a static website configuration	Queries the static website configuration of a bucket
DELETE Bucket website	Deleting a static website configuration	Deletes the static website configuration of a bucket

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

Setting Static Website Configuration

Description

This API is used to configure a static website for a bucket.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
PutBucketWebsiteRequest putBucketWebsiteRequest =
        new PutBucketWebsiteRequest(bucket);
// Set an index document
putBucketWebsiteRequest.setIndexDocument("index.html");

cosXmlService.putBucketWebsiteAsync(putBucketWebsiteRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        PutBucketWebsiteResult putBucketWebsiteResult =
                (PutBucketWebsiteResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Querying Static Website Configuration

Description

This API is used to query the static website configuration associated with a bucket.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
GetBucketWebsiteRequest getBucketWebsiteRequest =
        new GetBucketWebsiteRequest(bucket);
cosXmlService.getBucketWebsiteAsync(getBucketWebsiteRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        GetBucketWebsiteResult getBucketWebsiteResult =
                (GetBucketWebsiteResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Deleting Static Website Configuration

Description

This API is used to delete the static website configuration of a bucket.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
DeleteBucketWebsiteRequest deleteBucketWebsiteRequest =
        new DeleteBucketWebsiteRequest(bucket);

cosXmlService.deleteBucketWebsiteAsync(deleteBucketWebsiteRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        DeleteBucketWebsiteResult getBucketWebsiteResult =
                (DeleteBucketWebsiteResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
            }
        }
    });
```

Note:

For more samples, please visit GitHub.

Inventory

Last updated：2024-06-25 10:53:13

Overview

This document provides an overview of APIs and SDK code samples related to COS inventory.

API	Operation	Description
PUT Bucket inventory	Creating an inventory job	Creates an inventory job for a bucket
GET Bucket inventory	Querying inventory jobs	Queries the inventory jobs of a bucket
DELETE Bucket inventory	Deleting an inventory job	Deletes an inventory job from a bucket

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

Creating an Inventory Job

Description

This API (PUT Bucket inventory) is used to create an inventory job for a bucket.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
PutBucketInventoryRequest putBucketInventoryRequest =
        new PutBucketInventoryRequest(bucket);
putBucketInventoryRequest.setInventoryId("exampleInventoryId");
// Indicate whether to include object versions in the inventory:
// If set to All, all object versions are included in the inventory,
// with additional fields VersionId, IsLatest, and DeleteMarker
// If set to Current, no object versions are included in the inventory
putBucketInventoryRequest.setIncludedObjectVersions(InventoryConfiguration
        .IncludedObjectVersions.ALL);
// Backup frequency
putBucketInventoryRequest.setScheduleFrequency(InventoryConfiguration
        .SCHEDULE_FREQUENCY_DAILY);
```

```
// Backup path
putBucketInventoryRequest.setDestination("CSV", "1000000000",
        "examplebucket-1250000000", "region", "dir/");

cosXmlService.putBucketInventoryAsync(putBucketInventoryRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        PutBucketInventoryResult putBucketInventoryResult =
                (PutBucketInventoryResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Error codes

The following describes some common errors that may occur when you call this API:

Error Code	Description	Status Code
InvalidArgument	Invalid parameter value	HTTP 400 Bad Request
TooManyConfigurations	The number of inventories has reached the upper limit of 1,000	HTTP 400 Bad Request
AccessDenied	Unauthorized access. You most likely do not have access permission for the bucket	HTTP 403 Forbidden

Querying Inventory Jobs

Description

This API is used to query the inventory jobs of a bucket.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
GetBucketInventoryRequest getBucketInventoryRequest =
        new GetBucketInventoryRequest(bucket);
getBucketInventoryRequest.setInventoryId("exampleInventoryId");

cosXmlService.getBucketInventoryAsync(getBucketInventoryRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        GetBucketInventoryResult getBucketInventoryResult =
                (GetBucketInventoryResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Deleting an Inventory Job

Description

This API is used to delete a specified inventory job from a bucket.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
```

```
String bucket = "examplebucket-1250000000";
DeleteBucketInventoryRequest deleteBucketInventoryRequest =
        new DeleteBucketInventoryRequest(bucket);
deleteBucketInventoryRequest.setInventoryId("exampleInventoryId");

cosXmlService.deleteBucketInventoryAsync(deleteBucketInventoryRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        DeleteBucketInventoryResult deleteBucketInventoryResult =
                (DeleteBucketInventoryResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
    }
});
```

Note:

For more samples, please visit GitHub.

Cloud Access Management
Cross-Origin Resource Sharing

Last updated：2024-06-25 10:53:13

Overview

This document provides an overview of APIs and SDK sample codes related to cross-origin resource sharing (CORS).

API	Operation	Description
PUT Bucket cors	Setting CORS configuration	Sets the CORS permissions of bucket
GET Bucket cors	Querying CORS configuration	Queries the CORS configuration of a bucket
DELETE Bucket cors	Deleting CORS configuration	Deletes the CORS configuration of a bucket

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

Setting CORS Configuration

Description

This API is used to set the CORS configuration of a specified bucket.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
PutBucketCORSRequest putBucketCORSRequest = new PutBucketCORSRequest(bucket);

CORSConfiguration.CORSRule corsRule = new CORSConfiguration.CORSRule();

// Set the rule ID
corsRule.id = "123";
// Allowed origin in the format: `protocol://domain name[:port number]`. The wildca
corsRule.allowedOrigin = "https://cloud.tencent.com";
// Set the validity period of the OPTIONS request result
```

```
corsRule.maxAgeSeconds = 5000;

List<String> methods = new LinkedList<>();
methods.add("PUT");
methods.add("POST");
methods.add("GET");
// Allowed HTTP methods. Enumerated values: GET, PUT, HEAD, POST, DELETE
corsRule.allowedMethod = methods;

List<String> headers = new LinkedList<>();
headers.add("host");
headers.add("content-type");
// Notify the server which custom HTTP request headers are allowed for subsequent r
corsRule.allowedHeader = headers;

List<String> exposeHeaders = new LinkedList<>();
exposeHeaders.add("x-cos-meta-1");
// Set custom header information that the browser can receive from the server
corsRule.exposeHeader = exposeHeaders;

putBucketCORSRequest.addCORSRule(corsRule);

cosXmlService.putBucketCORSAsync(putBucketCORSRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        PutBucketCORSResult putBucketCORSResult = (PutBucketCORSResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Querying CORS Configuration

Description

This API is used to query the CORS configuration of a bucket.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
GetBucketCORSRequest getBucketCORSRequest = new GetBucketCORSRequest(bucket);
cosXmlService.getBucketCORSAsync(getBucketCORSRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        GetBucketCORSResult getBucketCORSResult = (GetBucketCORSResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Deleting CORS Configuration

Description

This API is used to delete the CORS configuration of a bucket.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
```

```
DeleteBucketCORSRequest deleteBucketCORSRequest =
        new DeleteBucketCORSRequest(bucket);
cosXmlService.deleteBucketCORSAsync(deleteBucketCORSRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        DeleteBucketCORSResult deleteBucketCORSResult =
                (DeleteBucketCORSResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Adding Domain Names

Last updated：2024-06-25 10:53:13

Overview

This document provides an overview of APIs and SDK code samples for custom domains.

API	Operation	Description
PUT Bucket domain	Setting a custom domain	Sets a custom domain for a bucket
GET Bucket domain	Querying a custom endpoint	Queries the custom endpoint of a bucket
DELETE Bucket domain	Deleting a custom domain	Deletes the custom domain configuration of a bucket

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

Setting Custom Domains

Feature description

This API is used to set a custom domain for a bucket.

Sample code

```
// Bucket name in the format of `BucketName-APPID` (`APPID` is required), which can
String bucket = "examplebucket-1250000000";
PutBucketDomainRequest putBucketDomainRequest =
        new PutBucketDomainRequest(bucket);
DomainConfiguration.DomainRule domainRule = new DomainConfiguration.DomainRule(
        DomainConfiguration.STATUS_ENABLED,
        "www.example.com",
        DomainConfiguration.TYPE_REST
);
domainRule.forcedReplacement = DomainConfiguration.REPLACE_CNAME;
putBucketDomainRequest.addDomainRule(domainRule);
```

```
cosXmlService.putBucketDomainAsync(putBucketDomainRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        PutBucketDomainResult putBucketDomainResult =
                (PutBucketDomainResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Error codes

The following describes some common errors that may occur when you call this API:

Status Code	Description
HTTP 409 Conflict	The domain record already exists, and forced overwrite is not specified in the request; OR the domain record does not exist, and forced overwrite is specified in the request
HTTP 451 Unavailable For Legal Reasons	The domain does not have an ICP filing in the Chinese mainland

Querying a Custom Domain

Feature description

This API is used to query the custom domain set for a bucket.

Sample code

```
// Bucket name in the format of `BucketName-APPID` (`APPID` is required), which can
String bucket = "examplebucket-1250000000";
GetBucketDomainRequest getBucketDomainRequest =
        new GetBucketDomainRequest(bucket);
cosXmlService.getBucketDomainAsync(getBucketDomainRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        GetBucketDomainResult getBucketTaggingResult =
                (GetBucketDomainResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Response parameters

Parameter	Description	API Type
x-cos-domain-txt-verification	Endpoint verification information. This field is an MD5 checksum of a character string in the format: cos[Region][BucketName-APPID][BucketCreateTime], where `Region` is the bucket region and `BucketCreateTime` is the time the bucket was created in GMT format	String

Deleting Custom Domains

Feature description

This API is used to delete the custom domain set for a bucket.

Note:

The COS Android SDK version should not be earlier than v5.9.8.

Sample code

```
// Bucket name in the format of `BucketName-APPID` (`APPID` is required), which can
String bucket = "examplebucket-1250000000";
DeleteBucketDomainRequest deleteBucketDomainRequest =
        new DeleteBucketDomainRequest(bucket);
cosXmlService.deleteBucketDomainAsync(deleteBucketDomainRequest,
        new CosXmlResultListener() {
            @Override
            public void onSuccess(CosXmlRequest request, CosXmlResult result) {
                // For detailed fields, see the API documentation or SDK source cod
                DeleteBucketDomainResult deleteBucketDomainResult =
                        (DeleteBucketDomainResult) result;
            }
            // If you use the Kotlin language to call this, please note that the ex
            // clientException is of type CosXmlClientException? and serviceExcepti
            @Override
            public void onFail(CosXmlRequest cosXmlRequest,
                               @Nullable CosXmlClientException clientException,
                               @Nullable CosXmlServiceException serviceException) {
                if (clientException != null) {
                    clientException.printStackTrace();
                } else {
                    serviceException.printStackTrace();
                }
            }
        });
```

Note:

For more samples, please visit GitHub.

Access Control

Last updated：2024-06-25 10:53:14

Overview

This document provides an overview of APIs and SDK code samples related to the access control lists (ACLs) for buckets and objects.

Bucket ACL

API	Operation	Description
PUT Bucket acl	Setting a bucket ACL	Sets an ACL for a bucket
GET Bucket acl	Querying a bucket ACL	Queries the ACL of a bucket

Object ACL

API	Operation	Description
PUT Object acl	Setting an object ACL	Sets an ACL for an object in a bucket
GET Object acl	Querying an object ACL	Queries the ACL of an object

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

Bucket ACL

Setting a bucket ACL

Description

This API is used to set an access control list (ACL) for a specified bucket.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
PutBucketACLRequest putBucketACLRequest = new PutBucketACLRequest(bucket);
```

```
// Set the bucket's access permissions
putBucketACLRequest.setXCOSACL("public-read");

// Grant read permission.
ACLAccount readACLS = new ACLAccount();
readACLS.addAccount("100000000001", "100000000001");
putBucketACLRequest.setXCOSGrantRead(readACLS);

// Grant write permission.
ACLAccount writeACLS = new ACLAccount();
writeACLS.addAccount("100000000001", "100000000001");
putBucketACLRequest.setXCOSGrantWrite(writeACLS);

// Grant read and write permission
ACLAccount writeandReadACLS = new ACLAccount();
writeandReadACLS.addAccount("100000000001", "100000000001");
putBucketACLRequest.setXCOSReadWrite(writeandReadACLS);

cosXmlService.putBucketACLAsync(putBucketACLRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        PutBucketACLResult putBucketACLResult = (PutBucketACLResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Querying a bucket ACL

Description

This API is used to query the access control list (ACL) of a specified bucket.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
GetBucketACLRequest getBucketACLRequest = new GetBucketACLRequest(bucket);
cosXmlService.getBucketACLAsync(getBucketACLRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        GetBucketACLResult getBucketACLResult = (GetBucketACLResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Object ACL

Setting an object ACL

Description

This API is used to set the access control list (ACL) for an object in a bucket.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; // The location identifier of the object in the b
PutObjectACLRequest putObjectACLRequest = new PutObjectACLRequest(bucket,
        cosPath);

// Set the object's access permissions
putObjectACLRequest.setXCOSACL("public-read");

// Grant read permission.
ACLAccount readACLS = new ACLAccount();
readACLS.addAccount("100000000001", "100000000001");
putObjectACLRequest.setXCOSGrantRead(readACLS);

// Grant read and write permission
ACLAccount writeandReadACLS = new ACLAccount();
writeandReadACLS.addAccount("100000000001", "100000000001");
putObjectACLRequest.setXCOSReadWrite(writeandReadACLS);

cosXmlService.putObjectACLAsync(putObjectACLRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        PutObjectACLResult putObjectACLResult = (PutObjectACLResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For the complete sample, go to GitHub.

Querying an object ACL

Description

This API is used to query the ACL of an object.

Sample code

```
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; // The location identifier of the object in the b
GetObjectACLRequest getBucketACLRequest = new GetObjectACLRequest(bucket,
        cosPath);
cosXmlService.getObjectACLAsync(getBucketACLRequest,
        new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        GetObjectACLResult getObjectACLResult = (GetObjectACLResult) result;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For the complete sample, go to GitHub.

Hotlink Protection

Last updated：2024-06-25 10:53:13

Overview

This document provides an overview of APIs and SDK code samples for hotlink protection.

API	Operation	Description
PUT Bucket referer	Setting hotlink protection	Sets hotlink protection for a bucket
GET Bucket referer	Querying the hotlink protection configuration	Queries the hotlink protection configuration of a bucket

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

Setting Hotlink Protection

Feature description

This API (`PUT Bucket referer`) is used to set hotlink protection for a bucket.

Sample code

```
// Bucket name in the format of `BucketName-APPID` (`APPID` is required), which can
String bucket = "examplebucket-1250000000";
PutBucketRefererRequest putBucketRefererRequest = new PutBucketRefererRequest(
        bucket, true, RefererConfiguration.RefererType.White);
putBucketRefererRequest.setAllowEmptyRefer(false);
ArrayList<RefererConfiguration.Domain> domainList = new ArrayList<>();
domainList.add(new RefererConfiguration.Domain("*.qq.com"));
domainList.add(new RefererConfiguration.Domain("*.qcloud.com"));
domainList.add(new RefererConfiguration.Domain("*.google.com"));
putBucketRefererRequest.setDomainList(domainList);
cosXmlService.putBucketRefererAsync(putBucketRefererRequest,
        new CosXmlResultListener() {
            @Override
            public void onSuccess(CosXmlRequest request, CosXmlResult result) {
                // For detailed fields, see the API documentation or SDK source cod
                PutBucketRefererResult putBucketRefererResult =
                        (PutBucketRefererResult) result;
            }
            // If you use the Kotlin language to call this, please note that the ex
            // clientException is of type CosXmlClientException? and serviceExcepti
            @Override
            public void onFail(CosXmlRequest cosXmlRequest,
                               @Nullable CosXmlClientException clientException,
                               @Nullable CosXmlServiceException serviceException) {
                if (clientException != null) {
                    clientException.printStackTrace();
                } else {
                    serviceException.printStackTrace();
                }
            }
        });
```

Note:

For more complete samples, visit GitHub.

Querying Hotlink Protection Configuration

Feature description

This API (`GET Bucket referer`) is used to query the hotlink protection configuration of a bucket.

Sample code

```
// Bucket name in the format of `BucketName-APPID` (`APPID` is required), which can
String bucket = "examplebucket-1250000000";
GetBucketRefererRequest getBucketRefererRequest = new GetBucketRefererRequest(bucke
cosXmlService.getBucketRefererAsync(getBucketRefererRequest,
        new CosXmlResultListener() {
            @Override
            public void onSuccess(CosXmlRequest request, CosXmlResult result) {
                // For detailed fields, see the API documentation or SDK source cod
                GetBucketRefererResult getBucketRefererResult =
                        (GetBucketRefererResult) result;
            }
            // If you use the Kotlin language to call this, please note that the ex
            // clientException is of type CosXmlClientException? and serviceExcepti
            @Override
            public void onFail(CosXmlRequest cosXmlRequest,
                               @Nullable CosXmlClientException clientException,
                               @Nullable CosXmlServiceException serviceException) {
                if (clientException != null) {
                    clientException.printStackTrace();
                } else {
                    serviceException.printStackTrace();
                }
            }
        });
```

Note:

For more complete samples, visit GitHub.

Bucket policy

Last updated：2024-06-25 10:53:13

Overview

This document provides an overview of APIs and SDK code samples related to bucket policies.

API	Operation	Description
PUT Bucket policy	Setting a bucket policy	Sets a permission policy for a bucket
GET Bucket policy	Querying a bucket policy	Queries the permission policy of a bucket
DELETE Bucket policy	Deleting a bucket policy	Deletes the permission policy of a bucket

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

Setting a bucket policy

Feature description

This API is used to set an access policy on a bucket.

Note:

The COS Android SDK version should not be earlier than v5.9.8.

Sample code

```
// Bucket name in the format of `BucketName-APPID` (`APPID` is required), which can
String bucket = "examplebucket-1250000000";
// Permission policy. For more information, visit https://cloud.tencent.com/documen
String policy = "{\\n" +
        "  \\"Statement\\": [\\n" +
        "    {\\n" +
        "      \\"Principal\\": {\\n" +
        "        \\"qcs\\": [\\n" +
        "          \\"qcs::cam::uin/100000000001:uin/100000000011\\"\\n" +
        "        ]\\n" +
        "      },\\n" +
        "      \\"Effect\\": \\"allow\\",\\n" +
        "      \\"Action\\": [\\n" +
        "        \\"name/cos:GetBucket\\"\\n" +
        "      ],\\n" +
        "      \\"Resource\\": [\\n" +
        "        \\"qcs::cos:ap-guangzhou:uid/1250000000:examplebucket-1250000000/*
        "      ]\\n" +
        "    }\\n" +
        "  ],\\n" +
        "  \\"version\\": \\"2.0\\"\\n" +
        "}";
PutBucketPolicyRequest putBucketPolicyRequest =
        new PutBucketPolicyRequest(bucket, policy);
cosXmlService.putBucketPolicyAsync(putBucketPolicyRequest,
        new CosXmlResultListener() {
            @Override
            public void onSuccess(CosXmlRequest request, CosXmlResult result) {
                // For detailed fields, see the API documentation or SDK source cod
                PutBucketPolicyResult putBucketPolicyResult =
                        (PutBucketPolicyResult) result;
            }
            // If you use the Kotlin language to call this, please note that the ex
            // clientException is of type CosXmlClientException? and serviceExcepti
            @Override
            public void onFail(CosXmlRequest cosXmlRequest,
                               @Nullable CosXmlClientException clientException,
                               @Nullable CosXmlServiceException serviceException) {
                if (clientException != null) {
                    clientException.printStackTrace();
                } else {
                    serviceException.printStackTrace();
                }
            }
        });
```

Note:

For more complete samples, visit GitHub.

Querying a bucket policy

Feature description

This API is used to query the access policy on a bucket.

Note:

The COS Android SDK version should not be earlier than v5.9.8.

Sample code

```
// Bucket name in the format of `BucketName-APPID` (`APPID` is required), which can
String bucket = "examplebucket-1250000000";
final GetBucketPolicyRequest getBucketPolicyRequest =
        new GetBucketPolicyRequest(bucket);
cosXmlService.getBucketPolicyAsync(getBucketPolicyRequest,
        new CosXmlResultListener() {
            @Override
            public void onSuccess(CosXmlRequest request, CosXmlResult result) {
                // For detailed fields, see the API documentation or SDK source cod
                GetBucketPolicyResult getBucketPolicyResult =
                        (GetBucketPolicyResult) result;
                String policy = getBucketPolicyResult.policy;
            }
            // If you use the Kotlin language to call this, please note that the ex
            // clientException is of type CosXmlClientException? and serviceExcepti
            @Override
            public void onFail(CosXmlRequest cosXmlRequest,
                               @Nullable CosXmlClientException clientException,
                               @Nullable CosXmlServiceException serviceException) {
                if (clientException != null) {
                    clientException.printStackTrace();
                } else {
                    serviceException.printStackTrace();
                }
            }
        });
```

Note:

For more complete samples, visit GitHub.

Deleting a bucket policy

Feature description

This API is used to delete the access policy from a specified bucket.

Note:

The COS Android SDK version should not be earlier than v5.9.8.

Sample code

```
// Bucket name in the format of `BucketName-APPID` (`APPID` is required), which can
String bucket = "examplebucket-1250000000";
DeleteBucketPolicyRequest deleteBucketPolicyRequest =
        new DeleteBucketPolicyRequest(bucket);
cosXmlService.deleteBucketPolicyAsync(deleteBucketPolicyRequest,
        new CosXmlResultListener() {
            @Override
            public void onSuccess(CosXmlRequest request, CosXmlResult result) {
                // For detailed fields, see the API documentation or SDK source cod
                DeleteBucketPolicyResult deleteBucketPolicyResult =
                        (DeleteBucketPolicyResult) result;
            }
            // If you use the Kotlin language to call this, please note that the ex
            // clientException is of type CosXmlClientException? and serviceExcepti
            @Override
            public void onFail(CosXmlRequest cosXmlRequest,
                               @Nullable CosXmlClientException clientException,
                               @Nullable CosXmlServiceException serviceException) {
                if (clientException != null) {
                    clientException.printStackTrace();
                } else {
                    serviceException.printStackTrace();
                }
            }
        });
```

Note:

For more complete samples, visit GitHub.

Data Verification

CRC64 Check

Last updated：2024-06-25 10:53:14

Overview

Errors may occur when data is transferred between the client and the server. COS can not only verify data integrity through MD5 and custom attributes, but also the CRC64 check code.
COS will calculate the CRC64 value of the newly uploaded object and store the result as object attributes. It will carry x-cos-hash-crc64ecma in the returned response header, which indicates the CRC64 value of the uploaded object calculated according to ECMA-182 standard. If an object already has a CRC64 value stored before this feature is activated, COS will not calculate its CRC64 value, nor will it be returned when the object is obtained.

Description

APIs that currently support CRC64 include:
APIs for simple upload
PUT Object and POST Object: you can get the CRC64 check value for your file from the response header.
Multipart upload APIs
Upload Part: you can compare and verify the CRC64 value returned by COS against the value calculated locally.
Complete Multipart Upload: returns a CRC64 value for the entire object only if each part has a CRC64 attribute. Otherwise, no value is returned.
The Upload Part - Copy operation returns a corresponding CRC64 value.
When you call the PUT Object - Copy, the CRC64 value is returned only if the source object has one.
The HEAD Object and GET Object operations return the CRC64 value provided the object has one. You can compare and verify the CRC64 value returned by COS against that calculated locally.

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

SDK Description

You can get the CRC64 value from the response header after a successful upload or download.

Note:

The COS Android SDK version should not be earlier than v5.7.5.

Upload request sample

```
// 1. Initialize TransferService. You should use the same TransferService for the s
TransferConfig transferConfig = new TransferConfig.Builder()
        .build();
TransferService transferService = new TransferService(cosXmlService, transferConfig

// 2. Initialize PutObjectRequest
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; // Location identifier of the object in the bucke
String srcPath = "examplefilepath"; // Absolute path to the local file
PutObjectRequest putObjectRequest = new PutObjectRequest(bucket,
        cosPath, srcPath);

// 3. Call the upload method to upload the file
final COSUploadTask uploadTask = transferService.upload(putObjectRequest);
uploadTask.setCosXmlResultListener(new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        // You can get the CRC64 value of the file after the upload is successful
        String crc64 = result.getHeader("x-cos-hash-crc64ecma");
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest request,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Samples for download requests

```
// 1. Initialize TransferService. You should use the same TransferService for the s
TransferConfig transferConfig = new TransferConfig.Builder()
        .build();
TransferService transferService = new TransferService(cosXmlService, transferConfig

// 2. Initialize GetObjectRequest
// Bucket name in the format of BucketName-APPID (APPID is required), which can be
String bucket = "examplebucket-1250000000";
String cosPath = "exampleobject"; // Location identifier of the object in the bucke
String savePathDir = context.getCacheDir().toString(); // Local directory path
// File name saved locally. If not specified (null), it will be the same as the COS
String savedFileName = "exampleobject";
GetObjectRequest getObjectRequest = new GetObjectRequest(bucket,
        cosPath, savePathDir, savedFileName);

// 3. Call the download method to download the file
final COSDownloadTask downloadTask = transferService.download(getObjectRequest);
downloadTask.setCosXmlResultListener(new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        // You can get the CRC64 value of the file after the download is successful
        String cosCRC64 = result.getHeader("x-cos-hash-crc64ecma");
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest request,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

CRC64 check

When you use `TransferService` for upload or download, the SDK verifies the data by default. If you still want to perform CRC64 check yourself, refer to the following code.

```
// 1. Refer to the above upload or download request sample code to get the CRC64 va
String cosCRC64 = "examplecoscrc64";

// 2. Calculate the CRC64 value of the local file
File localFile = new File("examplefilepath");
String localCRC64 = DigestUtils.getCRC64String(localFile);

// 3. Check whether localCRC64 and cosCRC64 are the same
if (localCRC64.equals(cosCRC64)) {
    // CRC64 values are the same
}
```

Note:

For more samples, please visit GitHub.

Image Processing
Persistent Image Processing

Last updated：2024-06-25 10:53:13

Overview

COS has integrated Cloud Infinite (CI), a one-stop professional multimedia solution that offers the image processing features outlined below. For more information, see Image Processing Overview.

Service	Feature	Description
Basic Image Processing	Scaling	Proportional scaling, scaling image to target width and height, and more
	Cropping	Cut (regular cropping), crop (scaling and cropping), iradius (inscribed circle cropping), and scrop (smart cropping)
	Rotation	Adaptive rotation and common rotation
	Format conversion	Format conversion, GIF optimization, and progressive display
	Quality conversion	Changes the quality of images in JPG and WEBP formats
	Gaussian blurring	Blurs images
	Sharpening	Sharpens images
	Watermarking	Image watermarks, text watermarks
	Obtaining image information	Basic information, EXIF data, average hue
	Removing metadata	Includes EXIF data
	Quick thumbnail template	Performs quick format conversion, scaling, and cropping to generate thumbnails
	Setting styles	Sets image styles to easily manage images for different purposes

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

Processing Image Upon Upload

The following example shows how to automatically process an image when you upload it to COS.
When the image is uploaded successfully, COS will save both the original and the processed images. You can later obtain the processing results using a general download request.

Sample code

```
List<PicOperationRule> rules = new LinkedList<>();
// Add a rule to convert the image to PNG format, and the processed image will be s
// examplepngobject
rules.add(new PicOperationRule("examplepngobject", "imageView2/format/png"));
PicOperations picOperations = new PicOperations(true, rules);

PutObjectRequest putObjectRequest = new PutObjectRequest(bucket, cosPath, srcPath);
putObjectRequest.setPicOperations(picOperations);

// If the upload is successful, you will get two images: the original and the proce
COSXMLUploadTask cosxmlUploadTask = transferManager.upload(putObjectRequest, upload
```

Note:

For more samples, go to GitHub.

Basic Image Processing

Last updated：2024-06-25 10:53:13

Overview

COS has integrated Cloud Infinite (CI), a one-stop professional multimedia solution that offers the image processing features outlined below. For more information, please see Image Processing Overview.

Service	Feature	Description
Image Processing-Basic Services	Scaling	Proportional scaling, scaling image to target width and height, and more
	Cropping	Cut (regular cropping), crop (scaling and cropping), iradius (inscribed circle cropping), and scrop (smart cropping)
	Rotation	Adaptive rotation and common rotation
	Format conversion	Format conversion, GIF optimization, and progressive display
	Quality conversion	Changes the quality of images in JPG and WEBP formats
	Gaussian blurring	Blurs images
	Sharpening	Sharpens images
	Watermarking	Image watermarks, text watermarks
	Obtaining image information	Basic information, EXIF data, average hue
	Removing metadata	Includes EXIF data
	Quick thumbnail template	Performs quick format conversion, scaling, and cropping to generate thumbnails

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

Scaling

Sample code

```
String bucket = "examplebucket-1250000000"; // Bucket, formatted as BucketName-APPI
String cosPath = "exampleobject"; // The location identifier of the object in the b
String savePath = context.getExternalCacheDir().toString(); // Local path

GetObjectRequest getObjectRequest = new GetObjectRequest(bucket, cosPath,
        savePath);
getObjectRequest.addQuery("imageMogr2/thumbnail/!50p", null);

cosXmlService.getObjectAsync(getObjectRequest, new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest cosXmlRequest,
                          CosXmlResult cosXmlResult) {
        GetObjectResult getObjectResult = (GetObjectResult) cosXmlResult;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Cropping

The following example shows how to process an in-cloud image and store the processing result in COS.

Sample code

```
String bucket = "examplebucket-1250000000"; // Bucket, formatted as BucketName-APPI
String cosPath = "exampleobject"; // The location identifier of the object in the b
String savePath = context.getExternalCacheDir().toString(); // Local path

GetObjectRequest getObjectRequest = new GetObjectRequest(bucket, cosPath,
        savePath);
getObjectRequest.addQuery("imageMogr2/iradius/150", null);

cosXmlService.getObjectAsync(getObjectRequest, new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest cosXmlRequest,
                          CosXmlResult cosXmlResult) {
        GetObjectResult getObjectResult = (GetObjectResult) cosXmlResult;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Rotating

The following sample shows how to process an image stored in COS upon the download:

Sample code

```
String bucket = "examplebucket-1250000000"; // Bucket, formatted as BucketName-APPI
String cosPath = "exampleobject"; // The location identifier of the object in the b
String savePath = context.getExternalCacheDir().toString(); // Local path

GetObjectRequest getObjectRequest = new GetObjectRequest(bucket, cosPath,
        savePath);
getObjectRequest.addQuery("imageMogr2/rotate/90", null);

cosXmlService.getObjectAsync(getObjectRequest, new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest cosXmlRequest,
                          CosXmlResult cosXmlResult) {
        GetObjectResult getObjectResult = (GetObjectResult) cosXmlResult;
    }

    // If you use the Kotlin language to call this, please note that the exception
    // clientException is of type CosXmlClientException? and serviceException is of
    @Override
    public void onFail(CosXmlRequest cosXmlRequest,
                       @Nullable CosXmlClientException clientException,
                       @Nullable CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Advanced Image Compression

Last updated：2024-06-25 10:53:13

Overview

This document provides an overview of APIs and SDK code samples related to advanced image compression.

API	Operation
Advanced image compression	Compresses images in a specified bucket

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

Advanced Image Compression

Description

Advanced image compression allows you to easily convert images into formats that provide a high compression ratio, such as TPG and HEIF. This effectively reduces the transmission time, loading time, and the use of bandwidth and traffic.

Sample code: performing advanced image compression upon download

```
String bucket = "examplebucket-1250000000"; // Bucket, formatted as BucketName-APPI
String cosPath = "exampleobject"; // The location identifier of the object in the b
String savePath = context.getExternalCacheDir().toString(); // Local path

GetObjectRequest getObjectRequest = new GetObjectRequest(bucket, cosPath,
        savePath);
getObjectRequest.addQuery("imageMogr2/format/tpg", null);

cosXmlService.getObjectAsync(getObjectRequest, new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest cosXmlRequest,
                          CosXmlResult cosXmlResult) {
        GetObjectResult getObjectResult = (GetObjectResult) cosXmlResult;
    }
```

```
      // If you use the Kotlin language to call this, please note that the exception
      // clientException is of type CosXmlClientException? and serviceException is of
      @Override
      public void onFail(CosXmlRequest cosXmlRequest,
                         @Nullable CosXmlClientException clientException,
                         @Nullable CosXmlServiceException serviceException) {
          if (clientException != null) {
              clientException.printStackTrace();
          } else {
              serviceException.printStackTrace();
          }
      }
  });
```

Note:

For more samples, please visit GitHub.

Blind Watermarking

Last updated：2024-06-25 10:53:14

Overview

This document provides an overview of APIs and SDK code samples related to COS blind watermarking.

API	Description
Blind watermarking	Adds blind watermarks to or extracts blind watermarks from local images and uploads them to a bucket

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

Adding Blind Watermarks

Description

COS allows you to add a blind watermark when uploading or downloading an object.

Sample 1. Adding a blind watermark when uploading

```
List<PicOperationRule> rules = new LinkedList<>();
// Add a rule for blind watermarking, and the processed image will be saved in the
// examplewatermarkobject
rules.add(new PicOperationRule("examplewatermarkobject",
        "watermark/3/type/1/image/aHR0cDovL2V4YW1wbGVzLTEyNTEwMDAw"));
PicOperations picOperations = new PicOperations(true, rules);

PutObjectRequest putObjectRequest = new PutObjectRequest(bucket, cosPath, srcPath);
putObjectRequest.setPicOperations(picOperations);

// If the upload is successful, you will get 2 images: the original and the process
COSXMLUploadTask cosxmlUploadTask = transferManager.upload(putObjectRequest, upload
```

Note:

For more samples, please visit GitHub.

Sample 2. Adding a blind watermark when downloading

```
GetObjectRequest getObjectRequest = new GetObjectRequest(bucket, cosPath, savePathD
// Add a text watermark
getObjectRequest.addQuery("watermark/3/type/3/text/dGVuY2VudCBjbG91ZA==", null);

COSXMLDownloadTask cosxmlDownloadTask =
        transferManager.download(applicationContext, getObjectRequest);
```

Note:

For more samples, please visit GitHub.

Setting Custom Headers

Last updated：2024-06-25 10:53:13

Overview

This document describes how to include custom headers in a request using the SDK.

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

Description

COS allows object upload requests to include custom headers that specify user-defined metadata. These headers start with `x-cos-meta-` , end with a custom suffix, and are saved as part of the object metadata.
If you have activated the Tencent Cloud CI service, you can specify the `Pic-Operations` header to enable automatic image processing. For detailed API instructions, see Persistence Processing.

Sample code

```
// The bucket region can be viewed in the COS console at https://console.intl.cloud
String region = "ap-beijing"; // Bucket region
String commonHeaderKey = "commonexamplekey"; // Key of the custom common header
String commonHeaderValue = "commonexamplevalue"; // Value of the custom common head
String requestHeaderKey = "requestexamplekey"; // Key of the custom request header
String requestHeaderValue = "requestexamplevalue"; // Value of the custom request h

CosXmlServiceConfig cosXmlServiceConfig = new CosXmlServiceConfig.Builder()
        .isHttps(true)
        .setRegion(region)
        .setDebuggable(false)
        // Add a custom common header to each request
        .addHeader(commonHeaderKey, commonHeaderValue)
        .builder();

CosXmlService cosXmlService = new CosXmlService(context, cosXmlServiceConfig,
        credentialProvider);

// Add a custom header with higher priority than common headers to a single request
HeadObjectRequest headObjectRequest = new HeadObjectRequest(bucket, cosPath);
try {
    headObjectRequest.setRequestHeaders(requestHeaderKey, requestHeaderValue, false
} catch (CosXmlClientException e) {
    e.printStackTrace();
}

// Initiate a request
cosXmlService.headObjectAsync(headObjectRequest, new CosXmlResultListener() {
    @Override
    public void onSuccess(CosXmlRequest request, CosXmlResult result) {
        HeadObjectResult headObjectResult = (HeadObjectResult) result;
    }

    @Override
    public void onFail(CosXmlRequest request, CosXmlClientException clientException
                       CosXmlServiceException serviceException) {
        if (clientException != null) {
            clientException.printStackTrace();
        } else {
            serviceException.printStackTrace();
        }
    }
});
```

Note:

For more samples, please visit GitHub.

Setting Access Domain Names (CDN/Global Acceleration)

Last updated：2024-06-25 10:53:13

Overview

This document describes how to request the COS service using a non-default endpoint.

SDK API References

For the parameters and method descriptions of all the APIs in the SDK, see SDK API Reference.

Default CDN acceleration domain name

For more information, see Enabling Default CDN Acceleration Domain Names.

The sample code below shows how to access a COS service using a default CDN acceleration domain name.

Sample code

```
// The bucket region can be viewed in the COS console at https://console.intl.cloud
String region = "ap-beijing"; // Bucket region
// Default CDN acceleration domain name of the bucket
String cdnDomain = "examplebucket-1250000000.file.myqcloud.com";

CosXmlServiceConfig cosXmlServiceConfig = new CosXmlServiceConfig.Builder()
        .isHttps(true)
        .setRegion(region)
        .setDebuggable(false)
        .setHostFormat(cdnDomain) // Modify the requested domain name
        .addHeader("Host", cdnDomain) // Modify the “Host” filed in the header
        .builder();

// The credentialProvider class is not provided. Instead, you add parameters to the
// for CDN permission verification
CosXmlService cosXmlService = new CosXmlService(context, cosXmlServiceConfig);
```

Note:

For more samples, please visit GitHub.

Custom CDN acceleration domain name

For more information, see Enabling Custom CDN Acceleration Domain Names.

The sample code below shows how to access a COS service using a custom CDN acceleration domain name.

Sample code

```
// The bucket region can be viewed in the COS console at https://console.intl.cloud
String region = "ap-beijing"; // Bucket region
String cdnCustomDomain = "exampledomain.com"; // Custom CDN acceleration domain nam

CosXmlServiceConfig cosXmlServiceConfig = new CosXmlServiceConfig.Builder()
        .isHttps(true)
        .setRegion(region)
        .setDebuggable(false)
        .setHostFormat(cdnCustomDomain) // Modify the requested domain name
        .addHeader("Host", cdnCustomDomain) // Modify the "Host" filed in the heade
        .builder();
// The credentialProvider class is not provided. Instead, you add parameters to the
// for CDN permission verification
CosXmlService cosXmlService = new CosXmlService(context, cosXmlServiceConfig);
```

Note:

For more samples, please visit GitHub.

Custom origin server domain name

For more information, see Enabling Custom Origin Domain.

The sample code below shows how to access a COS service using a custom origin server domain name.

Sample code

```
// The bucket region can be viewed in the COS console at https://console.intl.cloud
String region = "ap-beijing"; // Bucket region
String customDomain = "exampledomain.com"; // Custom origin server domain name

CosXmlServiceConfig cosXmlServiceConfig = new CosXmlServiceConfig.Builder()
        .isHttps(true)
        .setRegion(region)
        .setDebuggable(false)
        .setHostFormat(customDomain) // Modify the requested domain name
        .builder();

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

Note:

For more samples, please visit GitHub.

Global acceleration endpoint

For more information on global acceleration, see Overview.

The sample code below shows how to access a COS service using a global acceleration endpoint.

Sample code

```
// The bucket region can be viewed in the COS console at https://console.intl.cloud
String region = "ap-beijing"; // Bucket region

CosXmlServiceConfig cosXmlServiceConfig = new CosXmlServiceConfig.Builder()
        .isHttps(true)
        .setRegion(region)
        .setDebuggable(false)
        .setAccelerate(true) // Enable a global acceleration endpoint
        .builder();

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

Note:

For more samples, please visit GitHub.

Troubleshooting

Last updated：2024-06-25 10:53:13

Overview

If a COS request fails when calling this SDK, it will throw a `CosXmlClientException` (client exception) or `CosXmlServiceException` (server exception).
The client exception, `CosXmlClientException` , results from unexpected interaction issues between the client and the COS server, such as a failure to connect to the server, a failure to parse the data returned by the server, or the occurrence of an IO exception when reading a local file.
The server exception, `CosXmlServiceException` , occurs when the client interacts with the COS server normally, but the operation fails. For example, the client accesses a bucket that does not exist, deletes a file that does not exist, or does not have the permission to perform an operation.

Client Exceptions

Inherited from `Exception` , `CosClientException` is used in the same way as `Exception` . An additional member `errorCode` is also added, as described below:

Member	Description	Type
errorCode	Client error code, such as 10000, which indicates a parameter verification failure. For more information, see Error Codes	int

Server Exceptions

The server exception, `CosXmlServiceException` , occurs when, for example, the client accesses a bucket that does not exist, deletes a file that does not exist, or does not have the permission to perform an operation, etc.
`CosXmlServiceException` contains the status code returned by the server, the `requestid` , the error details, and so on. After an exception is captured, we recommended that you print out the entire exception as it contains necessary factors for troubleshooting. The member variables of exception are described as follows:

Member	Description	Type
requestId	Request ID, used to identify a request. It is very important for troubleshooting.	string
statusCode	Status code in the response. 4xx indicates that the request failed due to a client	string

	exception. 5xx indicates that the request failed due to a server exception. For more information, see Error Codes.	
errorCode	Error code returned by the body when the request fails. For more information, see Error Codes.	string
errorMessage	Error message returned by the body when the request fails. For more information, see Error Codes.	string

Using the Diagnosis Tool

COS provides a self-help diagnosis tool to help you quickly locate request problems and debug code.

Directions

1. Copy the request ID (`RequestId`) returned when the request error occurs.
2. Click Diagnosis Tool.
3. Enter `RequestId` and click **Diagnose**.
4. Wait and view the diagnostic result.

C SDK

Getting Started

Last updated：2024-02-01 18:01:21

Download and Installation

Relevant resources

Download COS XML C SDK source code: XML C SDK.

Download demo: XML C SDK Demo.

For the SDK changelog, see Changelog.

For SDK FAQs, see C SDK FAQs.

Note:

If you encounter errors such as non-existent functions or methods when using the SDK, you can update the SDK to the latest version and try again.

Environmental dependencies

Dependent library: libcurl apr apr-util minixml.

Installing SDK

1. Download the CMake tool (v2.6.0 and higher recommended) here and install it as shown below:

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

2. Download libcurl (v7.32.0 or higher recommended) here and install it as shown below:

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

3. Download apr (v1.5.2 or higher recommended) here and install it as shown below:

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

4. Download apr-util (v1.5.4 or higher recommended) here and install it as shown below. You need to specify the `--with-apr` option during installation.

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

5. Download minixml (v2.8 - 2.12 recommended) here and install it as shown below:

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

6. Compile the COS C SDK. Download the XML C SDK source code and run the following compiling commands:

```
cmake .
make
make install
```

Getting Started

Below is the general process of using COS XML C SDK.
1. Initialize the SDK.
2. Set the request option parameters. For the definitions of parameters such as APPID, SecretId, SecretKey, and Bucket, see COS Glossary.
APPID is one of the account IDs assigned by the system after you register for a Tencent Cloud account.
`access_key_id` and `access_key_secret` are account API keys.
`endpoint` is the COS access domain name. For more information, see Regions and Access Endpoints. For example, the endpoint of the Guangzhou region is `cos.ap-guangzhou.myqcloud.com`, and the endpoint of a global acceleration domain name is `cos.accelerate.myqcloud.com`. You can add "http" or "https" to the endpoints. The SDK accesses COS via HTTP by default. For example, the endpoint for accessing the Guangzhou region via HTTPS is `https://cos.ap-guangzhou.myqcloud.com`.
`is_cname` specifies whether an endpoint is a custom domain name. If `is_cname` is set to 1, the endpoint is a custom domain name.
3. Set the parameters required for APIs.
4. Call the SDK API to initiate a request and get the response.

Initialization

Note:

We recommend you use a temporary key as instructed in Generating and Using Temporary Keys to call the SDK for security purposes. When you apply for a temporary key, follow the Notes on Principle of Least Privilege to avoid leaking resources besides your buckets and objects.

If you must use a permanent key, we recommend you follow the Notes on Principle of Least Privilege to limit the scope of permission on the permanent key.

```
int main(int argc, char *argv[])
{
    /* Call the cos_http_io_initialize method at the program entry to perform certa
    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
        exit(1);
    }

    /* Call a COS SDK API to upload/download files */
    /* ... User logic code, omitted here */

    /* Call the cos_http_io_deinitialize method to release global resources allocat
    cos_http_io_deinitialize();
    return 0;
}
```

Initializing request options

```
/* Equivalent to `apr_pool_t`, the memory pool for memory management. The implement
cos_pool_t *pool;
cos_request_options_t *options;

/* Create a new memory pool. The second parameter is NULL, indicating that it is no
cos_pool_create(&pool, NULL);

/* Create and initialize `options`. This parameter contains global configuration in
 * The memory of `options` is allocated by the pool, and will be released upon pool
 */
options = cos_request_options_create(pool);
options->config = cos_config_create(options->pool);

/* cos_str_set is a cos_string_t type initialized with a char* string */
cos_str_set(&options->config->endpoint, "<user's endpoint>");              // Enter
cos_str_set(&options->config->access_key_id, "<user's SecretId>");         // This
cos_str_set(&options->config->access_key_secret, "<user's SecretKey>");    // This
cos_str_set(&options->config->appid, "<user's AppId>");                    // This

/* You can use a temporary key by setting `sts_token`. When you use a temporary key
//cos_str_set(&options->config->sts_token, "MyTokenString");
/* Whether CNAME is used */
options->config->is_cname = 0;
/* Use a custom domain name to access COS */
/*
options->config->is_cname = 1;
```

```
cos_str_set(&options->config->endpoint, "<Custom domain name>");
*/

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

/* Used to set whether to add Content-MD5 header automatically to the upload reques
cos_set_content_md5_enable(options->ctl, COS_FALSE);

/* Used to set the request routing address and port. Normally, you do not need to s
//cos_set_request_route(options->ctl, "192.168.12.34", 80);
```

Note:

For more information about how to generate and use a temporary key, see Generating and Using Temporary Keys.

Creating a 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;

/* Create a new memory pool. The second parameter is NULL, indicating that it is no
cos_pool_create(&p, NULL);

/* Create and initialize `options`. This parameter contains global configuration in
 * The memory of `options` is allocated by the pool, and will be released upon pool
 */
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);

/* Set configuration information such as appid, endpoint, access_key_id, acces_key_
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);
/* Enter the bucket name in the format of BucketName-APPID. */
cos_str_set(&bucket, TEST_BUCKET_NAME);

/* Call an 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);
```

Querying objects

```
cos_pool_t *p = NULL;
int is_cname = 0;
cos_status_t *s = NULL;
cos_request_options_t *options = NULL;
cos_list_object_params_t *list_params = NULL;
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
cos_pool_create(&p, NULL);

/* Create and initialize `options`. This parameter contains global configuration in
 * The memory of `options` is allocated by the pool, and will be released upon pool
 */
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);

/* Set configuration information such as appid, endpoint, access_key_id, acces_key_
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 here must be in the format of BucketName-APPID. */
cos_str_set(&bucket, TEST_BUCKET_NAME);

/* Call an API to query object list */
list_params = cos_create_list_object_params(p);
cos_str_set(&list_params->encoding_type, "url");
s = cos_list_object(options, &bucket, list_params, &resp_headers);
if (cos_status_is_ok(s)) {
        printf("list object succeeded\\n");
} else {
        printf("list object failed\\n");
}

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

Uploading an object

```
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
cos_pool_create(&p, NULL);

/* Create and initialize `options`. This parameter contains global configuration in
 * The memory of `options` is allocated by the pool, and will be released upon pool
 */
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);

/* Set configuration information such as appid, endpoint, access_key_id, acces_key_
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 here must be in the format of BucketName-APPID. */
cos_str_set(&bucket, TEST_BUCKET_NAME);

/* Call an API to upload an 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);
```

Downloading an object

```
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
cos_pool_create(&p, NULL);

/* Create and initialize `options`. This parameter contains global configuration in
 * The memory of `options` is allocated by the pool, and will be released upon pool
 */
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);

/* Set configuration information such as appid, endpoint, access_key_id, acces_key_
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 here must be in the format of BucketName-APPID. */
cos_str_set(&bucket, TEST_BUCKET_NAME);

/* Call an API to download an 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_head
if (cos_status_is_ok(s)) {
        printf("get object succeeded\\n");
} else {
        printf("get object failed\\n");
}
```

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

Deleting an object

```
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 a new memory pool. The second parameter is NULL, indicating that it is no
cos_pool_create(&p, NULL);

/* Create and initialize `options`. This parameter contains global configuration in
 * The memory of `options` is allocated by the pool, and will be released upon pool
 */
options = cos_request_options_create(p);
options->config = cos_config_create(options->pool);
init_test_config(options->config, is_cname);

/* Set configuration information such as appid, endpoint, access_key_id, acces_key_
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 here must be in the format of BucketName-APPID. */
cos_str_set(&bucket, TEST_BUCKET_NAME);

/* Call an API to delete an 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);
```

C SDK

Last updated：2024-02-01 18:01:21

How do I implement checkpoint restart with C SDK?

You can use the advanced upload API of C SDK to implement the checkpoint restart feature. To use checkpoint restart, you need to set the upload control parameter to **COS_TRUE**, for example, `clt_params = cos_create_resumable_clt_params_content(p, 0, 1, COS_TRUE, NULL)` .

Why does the `HttpIOError` error occur when I use C SDK?

Error description: When you use the SDK, all APIs cannot be used or return `requestid` . By analyzing the captured packets, you find that no HTTP requests are sent successfully as shown in the following logs:

```
transport failure curl code:1 error:Unsupported protocol
status->code: -996
status->error_code: HttpIoError
status->error_msg: Unsupported protocol
status->req_id:
```

This error occurs because the HTTPS protocol is used, but the libcurl library doesn't support HTTPS. Therefore, the OpenSSL library is not used or the versions mismatch during libcurl compilation.

Solution: Check the running environment and reinstall the libcurl library (if you install it by compiling the source code, enable SSL) or update the OpenSSL library.

Bucket Operations

Last updated：2024-02-01 18:01:21

Overview

This document provides an overview of APIs and SDK sample codes related to basic bucket operations and access control lists (ACL).

Note:

We recommend you use a temporary key as instructed in Generating and Using Temporary Keys to call the SDK for security purposes. When you apply for a temporary key, follow the Notes on Principle of Least Privilege to avoid leaking resources besides your buckets and objects.

If you must use a permanent key, we recommend you follow the Notes on Principle of Least Privilege to limit the scope of permission on the permanent key.

Basic operations

API	Operation	Description
PUT Bucket	Creating a bucket	Creates a bucket under a specified account
DELETE Bucket	Deleting a bucket	Deletes an empty bucket from a specified account

Extracting a bucket and its permissions

API	Operation	Description
HEAD Bucket	Checking a bucket and its permissions	Checks whether a bucket exists and whether you have permission to access it

Basic operations

Creating a bucket

Feature description

This API is used to create a bucket under a 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);
```

Field description

Parameter	Description	Type
options	COS request options	Struct
bucket	Bucket name in the format: `BucketName-APPID`	String
cos_acl	Allow users to customize permissions. Valid values: `COS_ACL_PRIVATE(0)` (default), `COS_ACL_PUBLIC_READ(1)` , `COS_ACL_PUBLIC_READ_WRITE(2)`	Enum
resp_headers	Returns the HTTP response headers	Struct

Response description

Response Parameter	Description	Type
code	Error code	Int
error_code	Error code content	String
error_msg	Error code description	String
req_id	Request message ID	String

Sample

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"
#include <unistd.h>

// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
static char *TEST_ACCESS_KEY_ID;                //your secret_id
static char *TEST_ACCESS_KEY_SECRET;            //your secret_key
// The only user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";    //your appid
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";
```

```
void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    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);
}

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_create_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;

    // Initialize the 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);

    // 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 the memory pool
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    //set log level, default COS_LOG_WARN
    cos_log_set_level(COS_LOG_WARN);

    //set log output, default stderr
    cos_log_set_output(NULL);

    test_create_bucket();

    cos_http_io_deinitialize();

    return 0;
}
```

Deleting a bucket

Feature description

This API is used to delete an empty bucket under a specified account.

Method prototype

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

Field description

Parameter	Description	Type
options	COS request options	Struct

bucket	Bucket name in the format: `BucketName-APPID`	String
resp_headers	Returns the HTTP response headers	Struct

Response description

Response Parameter	Description	Type
code	Error code	Int
error_code	Error code content	String
error_msg	Error code description	String
req_id	Request message ID	String

Sample

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"
#include <unistd.h>

// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
static char *TEST_ACCESS_KEY_ID;                //your secret_id
static char *TEST_ACCESS_KEY_SECRET;            //your secret_key
// The only user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";    //your appid
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    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);
}

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_delete_bucket()
{
    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;

    // Initialize the 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);

    // Delete a 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 the memory pool
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }
    //set log level, default COS_LOG_WARN
    cos_log_set_level(COS_LOG_WARN);

    //set log output, default stderr
    cos_log_set_output(NULL);

    test_delete_bucket();

    cos_http_io_deinitialize();

    return 0;
}
```

Checking whether a bucket exists

Feature description

This API is used to check whether a bucket exists. It actually calls the `HEAD Bucket` API to perform the action.

Method prototype

```
cos_status_t *cos_check_bucket_exist(const cos_request_options_t *options,
                                     const cos_string_t *bucket,
                                     cos_bucket_exist_status_e *bucket_exist,
                                     cos_table_t **resp_headers)
```

Field description

Parameter	Description	Type
options	COS request options	Struct
bucket	Bucket name in the format: `BucketName-APPID`	String
bucket_exist	Whether a bucket exists. Enumerated values: `exist` , `non exist` , `unknown` (no explicit state is obtained)	Enum
resp_headers	Returns the HTTP response headers	Struct

Response description

Response Parameter	Description	Type
code	Error code	Int

error_code	Error code content	String
error_msg	Error code description	String
req_id	Request message ID	String

Sample

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"
#include <unistd.h>

// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
static char *TEST_ACCESS_KEY_ID;                //your secret_id
static char *TEST_ACCESS_KEY_SECRET;            //your secret_key
// The only user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";    //your appid
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    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);
}

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}
```

```
void test_check_bucket_exist()
{
    cos_pool_t *pool = NULL;
    int is_cname = 0;
    cos_status_t *status = NULL;
    cos_request_options_t *options = NULL;
    cos_string_t bucket;
    cos_table_t *resp_headers = NULL;
    cos_bucket_exist_status_e bucket_exist;

    // Create a memory pool
    cos_pool_create(&pool, NULL);

    // Initialize the request options
    options = cos_request_options_create(pool);
    init_test_request_options(options, is_cname);

    cos_str_set(&bucket, TEST_BUCKET_NAME);

    // Check whether a bucket exists
    status = cos_check_bucket_exist(options, &bucket, &bucket_exist, &resp_headers)
    if (bucket_exist == COS_BUCKET_NON_EXIST) {
        printf("bucket: %.*s non exist.\\n", bucket.len, bucket.data);
    } else if (bucket_exist == COS_BUCKET_EXIST) {
        printf("bucket: %.*s exist.\\n", bucket.len, bucket.data);
    } else {
        printf("bucket: %.*s unknown status.\\n", bucket.len, bucket.data);
        log_status(status);
    }

    cos_pool_destroy(pool);
}

int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    //set log level, default COS_LOG_WARN
    cos_log_set_level(COS_LOG_WARN);
    //set log output, default stderr
    cos_log_set_output(NULL);

    test_check_bucket_exist();

    cos_http_io_deinitialize();

    return 0;
}
```

Object Operations

Uploading Objects

Last updated：2024-02-01 18:01:21

Overview

This document provides an overview of APIs and SDK code samples related to object uploads.

Simple operations

API	Operation	Description
PUT Object	Uploading an object (creating a directory)	Uploads an object to a bucket.
APPEND Object	Appending parts	Uploads an object by appending parts

Multipart operations

API	Operation	Description
List Multipart Uploads	Querying multipart uploads	Queries in-progress multipart uploads.
Initiate Multipart Upload	Initializing a multipart upload	Initializes a multipart upload.
Upload Part	Uploading parts	Uploads a file in parts.
List Parts	Querying uploaded parts	Queries the uploaded parts of a multipart upload.
Complete Multipart Upload	Completing a multipart upload	Completes the multipart upload of a file.
Abort Multipart Upload	Aborting a multipart upload	Aborts a multipart upload and deletes the uploaded parts.

Advanced APIs (Recommended)

Uploading an object (checkpoint restart)

Description

The upload API automatically divides your data into parts based on the size of your file. It's easier to use, eliminating the need to follow each step of the multipart upload process. The part size is 1,048,576 (1 MB) by default and can be adjusted via the `part_size` parameter.

Method prototype

```
cos_status_t *cos_resumable_upload_file(cos_request_options_t *options,
                                          cos_string_t *bucket,
                                          cos_string_t *object,
                                          cos_string_t *filepath,
                                          cos_table_t *headers,
                                          cos_table_t *params,
                                          cos_resumable_clt_params_t *clt_params,
                                          cos_progress_callback progress_callback,
                                          cos_table_t **resp_headers,
                                          cos_list_t *resp_body)
```

Parameter description

Parameter	Description	Type
options	COS request options	Struct
bucket	Bucket name in the format: `BucketName-APPID`	String
object	Object name	String
filepath	The local file name of the object	String
headers	Headers attached to the COS request	Struct
params	Parameters for the COS request	Struct
clt_params	Control parameters for the upload operation	Struct
part_size	Part size in bytes. If you specify the part size to be below 1,048,576 (1 MB), the C SDK will divide your data based on the part size 1,048,576 (1 MB) by default. If the number of parts exceeds 10,000, the C SDK adjusts the part size according to the file size.	Int
thread_num	Number of threads, that is, size of the thread pool. Default: 1	Int
enable_checkpoint	Indicates whether to enable checkpoint restart	Int
checkpoint_path	Indicates the file path for which the upload progress is saved when checkpoint restart is enabled. Default: `<filepath>.cp` , where `filepath` is the local file name of the object	String

progress_callback	Callback function for upload progress	Function
resp_headers	Returns the HTTP response headers	Struct
resp_body	Saves the data returned by the `Complete Multipart Upload` request	Struct

Response description

Response Parameter	Description	Type
code	Error code	Int
error_code	Error code content	String
error_msg	Error code description	String
req_id	Request message ID	String

Sample

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"

// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
static char *TEST_ACCESS_KEY_ID;                    // Your SecretId
static char *TEST_ACCESS_KEY_SECRET;                // Your SecretKey
// A unique user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";     // Your APPID
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";
static char TEST_MULTIPART_FILE[] = "test.zip";
static char TEST_MULTIPART_OBJECT4[] = "multipart4.dat";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    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);
}
```

```
void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_resumable_upload_with_multi_threads()
{
    cos_pool_t *p = NULL;
    cos_string_t bucket;
    cos_string_t object;
    cos_string_t filename;
    cos_status_t *s = NULL;
    int is_cname = 0;
    cos_table_t *headers = NULL;
    cos_table_t *resp_headers = NULL;
    cos_request_options_t *options = NULL;
    cos_resumable_clt_params_t *clt_params;

    cos_pool_create(&p, NULL);
    options = cos_request_options_create(p);
    init_test_request_options(options, is_cname);
    headers = cos_table_make(p, 0);
    cos_str_set(&bucket, TEST_BUCKET_NAME);
    cos_str_set(&object, TEST_MULTIPART_OBJECT4);
    cos_str_set(&filename, TEST_MULTIPART_FILE);

    // upload
    clt_params = cos_create_resumable_clt_params_content(p, 1024 * 1024, 8, COS_FAL
    s = cos_resumable_upload_file(options, &bucket, &object, &filename, headers, NU
        clt_params, NULL, &resp_headers, NULL);

    if (cos_status_is_ok(s)) {
        printf("upload succeeded\\n");
    } else {
        printf("upload failed\\n");
    }
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    // Set the log level. Default value: `COS_LOG_WARN`
    cos_log_set_level(COS_LOG_WARN);

    // Set log output. Default value: `stderr`
    cos_log_set_output(NULL);

    test_resumable_upload_with_multi_threads();

    cos_http_io_deinitialize();

    return 0;
}
```

Simple Operations

Uploading an object (creating a directory) using simple upload

Description

This API is used to upload an object of up to 5 GB in size to a specified bucket. To upload objects larger than 5 GB, please use Multipart Upload or Advanced APIs.

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);
```

Parameter description

Parameter	Description	Type
options	COS request options	Struct
bucket	Bucket name in the format: `BucketName-APPID`	String
object	Object name	String
filename	Filename of the local object before it is uploaded to COS	String
headers	Additional headers of a COS request	Struct
resp_headers	Returns the HTTP response headers	Struct

Response description

Response Parameter	Description	Type
code	Error code	Int
error_code	Error code content	String
error_msg	Error code description	String
req_id	Request message ID	String

Sample 1. Uploading an object

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"

// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
static char *TEST_ACCESS_KEY_ID;                // Your SecretId
static char *TEST_ACCESS_KEY_SECRET;            // Your SecretKey
// A unique user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";     // Your APPID
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";
// A unique identifier of an object stored in COS. For more information about objec
static char TEST_OBJECT_NAME1[] = "1.txt";

void log_status(cos_status_t *s)
```

```
{
    cos_warn_log("status->code: %d", s->code);
    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);
}

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_put_object_from_file()
{
    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;
    cos_string_t file;
    int traffic_limit = 0;

    cos_pool_create(&p, NULL);
    options = cos_request_options_create(p);
    init_test_request_options(options, is_cname);
    cos_table_t *headers = NULL;
    if (traffic_limit) {
        // The speed range is 819200 to 838860800, that is 100 KB/s to 100 MB/s. If
        headers = cos_table_make(p, 1);
        cos_table_add_int(headers, "x-cos-traffic-limit", 819200);
    }
    cos_str_set(&bucket, TEST_BUCKET_NAME);
    cos_str_set(&file, TEST_OBJECT_NAME1);
    cos_str_set(&object, TEST_OBJECT_NAME1);
    s = cos_put_object_from_file(options, &bucket, &object, &file, headers, &resp_h
    log_status(s);

    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    // Set the log level. Default value: `COS_LOG_WARN`
    cos_log_set_level(COS_LOG_WARN);

    // Set log output. Default value: `stderr`
    cos_log_set_output(NULL);

    test_put_object_from_file();

    cos_http_io_deinitialize();

    return 0;
}
```

Sample 2. Creating a folder

COS uses slashes (/) to separate object paths to simulate the effect of directories. Therefore, you can upload an empty stream and append a slash to its name to create an empty directory in COS.

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"

// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
static char *TEST_ACCESS_KEY_ID;                // Your SecretId
static char *TEST_ACCESS_KEY_SECRET;            // Your SecretKey
// A unique user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";    // Your APPID
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";
```

```
void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    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);
}

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_create_dir()
{
    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;
    cos_table_t *headers = NULL;
    cos_list_t buffer;

    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);
    cos_str_set(&object, "folder/");

    // Upload a folder
    cos_list_init(&buffer);
    s = cos_put_object_from_buffer(options, &bucket, &object,
            &buffer, headers, &resp_headers);
    if (cos_status_is_ok(s)) {
        printf("put object succeeded\\n");
    } else {
        printf("put object failed\\n");
    }
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    // Set the log level. Default value: `COS_LOG_WARN`
    cos_log_set_level(COS_LOG_WARN);

    // Set log output. Default value: `stderr`
    cos_log_set_output(NULL);

    test_create_dir();

    cos_http_io_deinitialize();

    return 0;
}
```

Sample 3. Uploading an object to a COS directory

You can upload an object whose name is separated by slashes. In this way, the directory that contains this object will be created automatically. If you need to upload new objects to this COS directory, you can set the key’s prefix to the value of this directory.

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"

// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
static char *TEST_ACCESS_KEY_ID;                // Your SecretId
static char *TEST_ACCESS_KEY_SECRET;            // Your SecretKey
```

```
// A unique user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";    // Your APPID
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    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);
}

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_upload_file_to_dir()
{
    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;

    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);
    cos_str_set(&file, "examplefile");
    cos_str_set(&object, "folder/exampleobject");
    s = cos_put_object_from_file(options, &bucket, &object, &file, NULL, &resp_head
    if (cos_status_is_ok(s)) {
        printf("put object succeeded\\n");
    } else {
        printf("put object failed\\n");
    }
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    // Set the log level. Default value: `COS_LOG_WARN`
    cos_log_set_level(COS_LOG_WARN);

    // Set log output. Default value: `stderr`
    cos_log_set_output(NULL);

    test_upload_file_to_dir();

    cos_http_io_deinitialize();

    return 0;
}
```

Appending parts

Description

This API is used to upload an object by appending parts of the object. The default maximum size of each object part is 5 GB (no minimum size limit), and the size of the object uploaded using this API can be up to 5 GB. If the value of `position` is inconsistent with the object length, COS will return the 409 status code. If the object to append is of the "normal" type, COS will return "409 ObjectNotAppendable".

Method prototype

```
cos_status_t *cos_append_object_from_file(const cos_request_options_t *options,
                                          const cos_string_t *bucket,
                                          const cos_string_t *object,
                                          int64_t position,
                                          const cos_string_t *append_file,
                                          cos_table_t *headers,
                                          cos_table_t **resp_headers);
```

Parameter description

Parameter	Description	Type
options	COS request options	Struct
bucket	Bucket name in the format: `BucketName-APPID`	String
object	Object name	String
position	Starting point for the append operation (in bytes). For the first append, the value of this parameter is 0. For subsequent appends, the value is the `content-length` of the current object.	String
append_file	Filename of the local object before it is uploaded to COS	String
headers	Additional headers of a COS request	Struct
resp_headers	Returns the HTTP response headers	Struct

Response description

Response Parameter	Description	Type
code	Error code	Int
error_code	Error code content	String
error_msg	Error code description	String
req_id	Request message ID	String

Sample

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"

// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
static char *TEST_ACCESS_KEY_ID;                // Your SecretId
static char *TEST_ACCESS_KEY_SECRET;             // Your SecretKey
// A unique user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";    // Your APPID
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";
static char TEST_OBJECT_NAME3[] = "test3.dat";
static char *TEST_APPEND_NAMES[] = {"test.7z.001", "test.7z.002"};

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    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);
}

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_append_object()
{
    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 a memory pool
    cos_pool_create(&p, NULL);

    // Initialize the request options
    options = cos_request_options_create(p);
    init_test_request_options(options, is_cname);
    cos_str_set(&bucket, TEST_BUCKET_NAME);

    // Append parts
    cos_str_set(&object, TEST_OBJECT_NAME3);
    int32_t count = sizeof(TEST_APPEND_NAMES)/sizeof(char*);
    int32_t index = 0;
    int64_t position = 0;
    s = cos_head_object(options, &bucket, &object, NULL, &resp_headers);
    if(s->code == 200) {
        char *content_length_str = (char*)apr_table_get(resp_headers, COS_CONTENT_L
        if (content_length_str != NULL) {
            position = atol(content_length_str);
        }
    }
    for (; index < count; index++)
    {
        cos_str_set(&file, TEST_APPEND_NAMES[index]);
        s = cos_append_object_from_file(options, &bucket, &object,
                                        position, &file, NULL, &resp_headers);
        log_status(s);

        s = cos_head_object(options, &bucket, &object, NULL, &resp_headers);
        if(s->code == 200) {
            char *content_length_str = (char*)apr_table_get(resp_headers, COS_CONTE
            if (content_length_str != NULL) {
                position = atol(content_length_str);
            }
        }
    }

    // Destroy the memory pool.
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    // Set the log level. Default value: `COS_LOG_WARN`
    cos_log_set_level(COS_LOG_WARN);

    // Set log output. Default value: `stderr`
    cos_log_set_output(NULL);

    test_append_object();

    cos_http_io_deinitialize();

    return 0;
}
```

Multipart Operations

Querying multipart uploads

Description

This API is used to query in-progress multipart uploads. Up to 1,000 in-progress multipart uploads can be queried in a single request

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);
```

Parameter description

Parameter	Description	Type
options	COS request options	Struct
bucket	Bucket name in the format of `BucketName-APPID`	String
params	Parameters for the `List Multipart Uploads` operation	Struct
encoding_type	Specifies the encoding type of the returned value	String
prefix	Prefix to be matched, which is used to specify the prefix address of the files to be returned	String
delimiter	The delimiter is a symbol.	String

	If a particular prefix is specified, identical paths between the prefix and the delimiter will be grouped together and defined as a common prefix, and then all common prefixes will be listed. If no prefix is specified, the list will start from the beginning of the path.	
max_ret	The maximum number of returned entries per request; the default value is 1000	String
key_marker	Used together with `upload-id-marker` . If `upload-id-marker` is not specified, only the multipart uploads whose `ObjectName` is lexicographically greater than `key-marker` will be listed; If `upload-id-marker` is specified, the multipart uploads whose `ObjectName` is lexicographically greater than the specified `key-marker` will be listed, and any multipart upload whose `ObjectName` lexicographically equals `key-marker` and whose `UploadID` is greater than `upload-id-marker` 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, the multipart uploads whose `ObjectName` is lexicographically greater than the specified `key-marker` will be listed, and any multipart upload whose `ObjectName` lexicographically equals `key-marker` and whose `UploadID` is greater than `upload-id-marker` will also be listed	String
truncated	Indicates whether the returned entry is truncated. Valid value: `true` or `false`	Boolean
next_key_marker	If the returned list is truncated, the `NextKeyMarker` returned will be the starting point of the subsequent list.	String
next_upload_id_marker	If the returned list is truncated, the `UploadId` returned will be the starting point of the subsequent list.	String
upload_list	Lists all multipart uploads	Struct
key	Object name	String
upload_id	Identifies the ID of the current multipart upload	String
initiated	Indicates when the current multipart upload was initiated	String

resp_headers	Returns the HTTP response headers	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;
```

Response description

Response Parameter	Description	Type
code	Error code	Int
error_code	Error code content	String
error_msg	Error code description	String
req_id	Request message ID	String

Sample

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"

// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
static char *TEST_ACCESS_KEY_ID;                // Your SecretId
static char *TEST_ACCESS_KEY_SECRET;            // Your SecretKey
// A unique user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";    // Your APPID
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";
static char TEST_MULTIPART_OBJECT[] = "multipart.dat";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    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);
}
```

```
void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void list_multipart()
{
    cos_pool_t *p = NULL;
    cos_string_t bucket;
    cos_string_t object;
    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;
    cos_list_upload_part_params_t *list_upload_param = NULL;

    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_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_he
    log_status(s);

    list_upload_param = cos_create_list_upload_part_params(p);
    list_upload_param->max_ret = 1000;
    cos_string_t upload_id;
    cos_str_set(&upload_id,"149373379126aee264fecbf5fe8ddb8b9cd23b76c73ab1af0bcfd50
    cos_str_set(&object, TEST_MULTIPART_OBJECT);
    s = cos_list_upload_part(options, &bucket, &object, &upload_id,
                             list_upload_param, &resp_headers);
    if (cos_status_is_ok(s)) {
        printf("List upload part succeeded, upload_id::%.*s\\n",
               upload_id.len, upload_id.data);
        cos_list_part_content_t *part_content = NULL;
        cos_list_for_each_entry(cos_list_part_content_t, part_content, &list_upload
            printf("part_number = %s, size = %s, last_modified = %s, etag = %s\\n",
                   part_content->part_number.data,
                   part_content->size.data,
                   part_content->last_modified.data,
                   part_content->etag.data);
        }
    } else {
        printf("List upload part failed\\n");
    }

    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    // Set the log level. Default value: `COS_LOG_WARN`
    cos_log_set_level(COS_LOG_WARN);

    // Set log output. Default value: `stderr`
    cos_log_set_output(NULL);

    list_multipart();

    cos_http_io_deinitialize();

    return 0;
}
```

Initializing a multipart upload

Description

This API is used to initialize a multipart upload. After the request is executed successfully, the `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);
```

Parameter description

Parameter	Description	Type
options	COS request options	Struct
bucket	Bucket name in the format: `BucketName-APPID`	String
object	Object name	String
upload_id	Returns the ID of the multipart upload operation	String
headers	Additional headers of a COS request	Struct
resp_headers	Returns the HTTP response headers	Struct

Response description

Response Parameter	Description	Type
code	Error code	Int
error_code	Error code content	String
error_msg	Error code description	String
req_id	Request message ID	String

Sample

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"

// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
```

```
static char *TEST_ACCESS_KEY_ID;                    // Your SecretId
static char *TEST_ACCESS_KEY_SECRET;                // Your SecretKey
// A unique user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";     // Your APPID
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";
static char TEST_MULTIPART_OBJECT[] = "multipart.dat";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    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);
}

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_init_multipart_upload()
{
    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 upload_id;
    cos_table_t *resp_headers = NULL;

    // Create a memory pool
    cos_pool_create(&p, NULL);

    // Initialize the request options
    options = cos_request_options_create(p);
    init_test_request_options(options, is_cname);
    cos_str_set(&bucket, TEST_BUCKET_NAME);

    // Initialize the multipart upload
    cos_str_set(&object, TEST_MULTIPART_OBJECT);
    s = cos_init_multipart_upload(options, &bucket, &object,
                                  &upload_id, NULL, &resp_headers);
    if (cos_status_is_ok(s)) {
        printf("init multipart upload succeeded\\n");
    } else {
        printf("init multipart upload failed\\n");
    }

    // Destroy the memory pool.
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    // Set the log level. Default value: `COS_LOG_WARN`
    cos_log_set_level(COS_LOG_WARN);

    // Set log output. Default value: `stderr`
    cos_log_set_output(NULL);

    test_init_multipart_upload();

    cos_http_io_deinitialize();

    return 0;
}
```

Uploading parts

Description

This API is used to upload parts (possibly out of order) in an initiated multipart upload operation. This API supports the upload of between 1 to 10,000 parts, with each part ranging from 1 MB to 5 GB in size. The `Upload Part` request should include the `partNumber` and `uploadID` .

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);
```

Parameter description

Parameter	Description	Type
options	COS request options	Struct
bucket	Bucket name in the format: `BucketName-APPID`	String
object	Object name	String
upload_id	Upload task ID	String
part_num	Part number	Int
upload_file	Information on the file to be uploaded	Struct
resp_headers	Returns the HTTP response headers	Struct

Response description

Response Parameter	Description	Type
code	Error code	Int
error_code	Error code content	String
error_msg	Error code description	String
req_id	Request message ID	String

Sample

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"

// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
static char *TEST_ACCESS_KEY_ID;                  // Your SecretId
static char *TEST_ACCESS_KEY_SECRET;              // Your SecretKey
// A unique user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";     // Your APPID
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";
static char TEST_MULTIPART_OBJECT[] = "multipart.dat";
static char TEST_MULTIPART_FILE[] = "test.zip";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    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);
}

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_upload_part()
{
    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 upload_id = cos_string("xxxxxxxxxxxxxxxxxxx");  // Use your own `u
    cos_table_t *resp_headers = NULL;
    int part_num = 1;
    int64_t pos = 0;
    int64_t file_length = 0;

    // Create a memory pool
    cos_pool_create(&p, NULL);

    // Initialize the request options
    options = cos_request_options_create(p);
    init_test_request_options(options, is_cname);
    cos_str_set(&bucket, TEST_BUCKET_NAME);
    cos_str_set(&object, TEST_MULTIPART_OBJECT);

    // Upload the parts
    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 the memory pool.
    cos_pool_destroy(p);
}
```

```
int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    // Set the log level. Default value: `COS_LOG_WARN`
    cos_log_set_level(COS_LOG_WARN);

    // Set log output. Default value: `stderr`
    cos_log_set_output(NULL);

    test_upload_part();

    cos_http_io_deinitialize();

    return 0;
}
```

Querying uploaded parts

Description

This API is used to query the uploaded parts of a specified multipart upload.

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);
```

Parameter description

Parameter	Description	Type
options	COS request options	Struct
bucket	Bucket name in the format of `BucketName-APPID`	String

object	Object name	String
upload_id	Upload task ID	String
params	Parameters for the `List Parts` operation	Struct
part_number_marker	Marks the starting point of the list of parts; by default, entries are listed in UTF-8 binary order starting from this marker	String
encoding_type	Specifies the encoding type of the returned value	String
max_ret	The maximum number of returned entries per request; the default value is 1000	String
truncated	Indicates whether the returned entry is truncated. Valid value: `true` or `false`	Boolean
next_part_number_marker	Marks the starting point of the next entry if the returned entry is truncated	String
part_list	Lists all uploaded parts	Struct
part_number	Part number	String
size	Part size in bytes	String
etag	SHA-1 check value of the part	String
last_modified	The time the part was last modified	String
resp_headers	Returns the HTTP response headers	Struct

Response description

Response Parameter	Description	Type
code	Error code	Int
error_code	Error code content	String
error_msg	Error code description	String
req_id	Request message ID	String

Sample

```
#include "cos_http_io.h"
```

```
#include "cos_api.h"
#include "cos_log.h"

// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
static char *TEST_ACCESS_KEY_ID;                // Your SecretId
static char *TEST_ACCESS_KEY_SECRET;            // Your SecretKey
// A unique user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";    // Your APPID
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";
static char TEST_MULTIPART_OBJECT[] = "multipart.dat";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    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);
}

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_list_upload_part()
{
    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_list_upload_part_params_t *params = NULL;
    cos_list_t complete_part_list;
    cos_string_t upload_id = cos_string("xxxxxxxxxxxxxxxxxxx");  // Use your own `u
    cos_table_t *resp_headers = NULL;

    // Create a memory pool
    cos_pool_create(&p, NULL);

    // Initialize the request options
    options = cos_request_options_create(p);
    init_test_request_options(options, is_cname);
    cos_str_set(&bucket, TEST_BUCKET_NAME);
    cos_str_set(&object, TEST_MULTIPART_OBJECT);

    // Query the uploaded parts
    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");
    }

    // Destroy the memory pool.
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    // Set the log level. Default value: `COS_LOG_WARN`
    cos_log_set_level(COS_LOG_WARN);

    // Set log output. Default value: `stderr`
    cos_log_set_output(NULL);

    test_list_upload_part();
    cos_http_io_deinitialize();

    return 0;
}
```

Completing a multipart upload

Description

This API is used to complete the multipart upload of an entire file. You can use this API to complete the multipart upload when you have uploaded all the parts using the `Upload Parts` API. When using this API, you need to provide the `PartNumber` and `ETag` for every part in the body to verify the accuracy of the parts.

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);
```

Parameter description

Parameter	Description	Type
options	COS request options	Struct
bucket	Bucket name in the format: `BucketName-APPID`	String
object	Object name	String
upload_id	Upload task ID	String
part_list	Parameters for the `Complete Multipart Upload` operation	Struct
part_number	Part number	String
etag	ETag of the part, which is the `sha1` checksum value. It must be enclosed in double quotes, such as: `"3a0f1fd698c235af9cf098cb74aa25bc"` .	String
headers	Additional headers of a COS request	Struct
resp_headers	Returns the HTTP response headers	Struct

Response description

Response Parameter	Description	Type
code	Error code	Int
error_code	Error code content	String
error_msg	Error code description	String
req_id	Request message ID	String

Sample

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"

// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
static char *TEST_ACCESS_KEY_ID;                // Your SecretId
static char *TEST_ACCESS_KEY_SECRET;            // Your SecretKey
// A unique user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";    // Your APPID
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";
static char TEST_MULTIPART_OBJECT[] = "multipart.dat";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    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);
}

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
```

```
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_complete_multipart_upload()
{
    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_list_upload_part_params_t *params = NULL;
    cos_list_t complete_part_list;
    cos_string_t upload_id = cos_string("xxxxxxxxxxxxxxxxxxx");  // Use your own `u
    cos_table_t *resp_headers = NULL;
    cos_list_part_content_t *part_content = NULL;
    cos_complete_part_content_t *complete_part_content = NULL;

    // Create a memory pool
    cos_pool_create(&p, NULL);

    // Initialize the request options
    options = cos_request_options_create(p);
    init_test_request_options(options, is_cname);
    cos_str_set(&bucket, TEST_BUCKET_NAME);
    cos_str_set(&object, TEST_MULTIPART_OBJECT);

    // Query the uploaded parts
    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_li
        complete_part_content = cos_create_complete_part_content(p);
        cos_str_set(&complete_part_content->part_number, part_content->part_number.
        cos_str_set(&complete_part_content->etag, part_content->etag.data);
        cos_list_add_tail(&complete_part_content->node, &complete_part_list);
    }

    // Complete the multipart upload
    s = cos_complete_multipart_upload(options, &bucket, &object, &upload_id,
                                        &complete_part_list, NULL, &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 the memory pool.
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    // Set the log level. Default value: `COS_LOG_WARN`
    cos_log_set_level(COS_LOG_WARN);

    // Set log output. Default value: `stderr`
    cos_log_set_output(NULL);

    test_complete_multipart_upload();

    cos_http_io_deinitialize();

    return 0;
}
```

Aborting a multipart upload

Description

This API is used to abort a multipart upload operation and delete the uploaded parts. When this API is called, a failure is returned for any request using the `Upload Parts` API.

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);
```

Parameter description

Parameter	Description	Type
options	COS request options	Struct
bucket	Bucket name in the format: `BucketName-APPID`	String
object	Object name	String
upload_id	Upload task ID	String
resp_headers	Returns the HTTP response headers	Struct

Response description

Response Parameter	Description	Type
code	Error code	Int
error_code	Error code content	String
error_msg	Error code description	String
req_id	Request message ID	String

Sample

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"

// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
```

```
static char *TEST_ACCESS_KEY_ID;                    // Your SecretId
static char *TEST_ACCESS_KEY_SECRET;                // Your SecretKey
// A unique user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";     // Your APPID
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";
static char TEST_MULTIPART_OBJECT[] = "multipart.dat";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    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);
}

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void abort_multipart_upload()
{
    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;

    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);

    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;
    }

    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);
}

int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    // Set the log level. Default value: `COS_LOG_WARN`
    cos_log_set_level(COS_LOG_WARN);

    // Set log output. Default value: `stderr`
    cos_log_set_output(NULL);

    abort_multipart_upload();
    cos_http_io_deinitialize();

    return 0;
}
```

Copying and Moving Objects

Last updated：2024-02-01 18:01:21

Overview

This document provides an overview of APIs and SDK code samples related to object copy and movement.

Simple operations

API	Operation	Description
PUT Object - Copy	Copying an object (modifying attributes)	Copies an object to a destination path
DELETE Object	Deleting an object	Deletes a specified object from a bucket.

Multipart operations

API	Operation	Description
Upload Part - Copy	Copying a part	Copies an object as a part.

Simple Copy

Copying an object (modifying attributes)

Description

This API is used to copy an object to a destination path. You can also use this API to modify object attributes such as storage class.

Method prototype

```
cos_status_t *cos_copy_object(const cos_request_options_t *options,
                              const cos_string_t *src_bucket,
                              const cos_string_t *src_object,
                              const cos_string_t *src_endpoint,
                              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);
```

Parameter description

Parameter	Description	Type
options	COS request options	Struct
src_bucket	Source bucket	String
src_object	Name of the source object	String
src_endpoint	Endpoint of the source object	String
dest_bucket	Name of the destination bucket name in the format of `BucketName-APPID`	String
dest_object	Name of the destination object	String
headers	Headers attached to the COS request	Struct
copy_object_param	Parameters of the `Put Object Copy` operation	Struct
etag	Returns the MD5 checksum of the file	String
last_modify	Returns the time the file was last modified in GMT format	String
resp_headers	Returns the HTTP response headers	Struct

Response description

Response Parameter	Description	Type
code	Error code	Int
error_code	Error code content	String
error_msg	Error code description	String
req_id	Request message ID	String

Sample 1. Copying an object

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"

// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
```

```
static char *TEST_ACCESS_KEY_ID;                    // Your SecretId
static char *TEST_ACCESS_KEY_SECRET;                // Your SecretKey
// A unique user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";     // Your APPID
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";
// A unique identifier of an object stored in COS. For more information about objec
static char TEST_OBJECT_NAME1[] = "1.txt";
static char TEST_OBJECT_NAME2[] = "test2.dat";

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_copy()
{
    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 src_bucket;
    cos_string_t src_object;
    cos_string_t src_endpoint;
    cos_table_t *resp_headers = NULL;

    // Create a memory pool
    cos_pool_create(&p, NULL);

    // Initialize the request options
    options = cos_request_options_create(p);
    init_test_request_options(options, is_cname);
    cos_str_set(&bucket, TEST_BUCKET_NAME);
    // Set object replication
    cos_str_set(&object, TEST_OBJECT_NAME2);
    cos_str_set(&src_bucket, TEST_BUCKET_NAME);
    cos_str_set(&src_endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&src_object, TEST_OBJECT_NAME1);

    cos_copy_object_params_t *params = NULL;
    params = cos_create_copy_object_params(p);
    s = cos_copy_object(options, &src_bucket, &src_object, &src_endpoint, &bucket,
    if (cos_status_is_ok(s)) {
        printf("put object copy succeeded\\n");
    } else {
        printf("put object copy failed\\n");
    }

    // Destroy the memory pool.
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    // Set the log level. Default value: `COS_LOG_WARN`
    cos_log_set_level(COS_LOG_WARN);

    // Set log output. Default value: `stderr`
    cos_log_set_output(NULL);

    test_copy();

    cos_http_io_deinitialize();

    return 0;
}
```

Sample 2. Modifying storage class

Note:

You can change STANDARD to STANDARD_IA, INTELLIGENT TIERING, ARCHIVE, or DEEP ARCHIVE. To modify ARCHIVE or DEEP ARCHIVE to other storage classes, you need to call `cos_post_object_restore()` to restore objects in ARCHIVE or DEEP ARCHIVE first before calling this API. For more information, please see Storage Class Overview.

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"

// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
static char *TEST_ACCESS_KEY_ID;                // Your SecretId
static char *TEST_ACCESS_KEY_SECRET;            // Your SecretKey
// A unique user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";    // Your APPID
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";
// A unique identifier of an object stored in COS. For more information about objec
static char TEST_OBJECT_NAME1[] = "1.txt";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    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);
}

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_modify_storage_class()
```

```
{
    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 src_bucket;
    cos_string_t src_object;
    cos_string_t src_endpoint;
    cos_table_t *resp_headers = NULL;

    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);
    cos_str_set(&object, TEST_OBJECT_NAME1);
    cos_str_set(&src_bucket, TEST_BUCKET_NAME);
    cos_str_set(&src_endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&src_object, TEST_OBJECT_NAME1);

    // Set the `x-cos-metadata-directive` and `x-cos-storage-class` headers and rep
    cos_table_t *headers = cos_table_make(p, 2);
    apr_table_add(headers, "x-cos-metadata-directive", "Replaced");
    // Valid storage classes are NTELLIGENT_TIERING, MAZ_INTELLIGENT_TIERING, STAND
    apr_table_add(headers, "x-cos-storage-class", "ARCHIVE");

    cos_copy_object_params_t *params = NULL;
    params = cos_create_copy_object_params(p);
    s = cos_copy_object(options, &src_bucket, &src_object, &src_endpoint, &bucket,
    log_status(s);

    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    // Set the log level. Default value: `COS_LOG_WARN`
    cos_log_set_level(COS_LOG_WARN);
    // Set log output. Default value: `stderr`
    cos_log_set_output(NULL);

    test_modify_storage_class();

    cos_http_io_deinitialize();

    return 0;
}
```

Multipart Copy

Copying an object part

Description

This API is used to copy a part of an object.

Method prototype

```
cos_status_t *cos_upload_part_copy(const cos_request_options_t *options,
                                   cos_upload_part_copy_params_t *params,
                                   cos_table_t *headers,
                                   cos_table_t **resp_headers);
```

Parameter description

Parameter	Description	Type
options	COS request options	Struct
params	Parameters for the `Upload Part - Copy` operation	Struct
copy_source	Source file path	String
dest_bucket	Name of the destination bucket in the format of `BucketName-APPID`	String
dest_object	Name of the destination object	String
upload_id	Upload task ID	String
part_num	Part number	Int
range_start	The starting offset of the source file	Int

range_end	The ending offset of the source file	Int
rsp_content	The response of the `Upload Part - Copy` operation	Struct
etag	Returns the MD5 checksum of the file	String
last_modify	Returns the time the file was last modified in GMT format	String
resp_headers	Returns the HTTP response headers	Struct

Response description

Response Parameter	Description	Type
code	Error code	Int
error_code	Error code content	String
error_msg	Error code description	String
req_id	Request message ID	String

Sample

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"
#include <sys/stat.h>

// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
static char *TEST_ACCESS_KEY_ID;                // Your SecretId
static char *TEST_ACCESS_KEY_SECRET;            // Your SecretKey
// A unique user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";    // Your APPID
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    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);
}
```

```
void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void make_rand_string(cos_pool_t *p, int len, cos_string_t *data)
{
    char *str = NULL;
    int i = 0;
    str = (char *)cos_palloc(p, len + 1);
    for ( ; i < len; i++) {
        str[i] = 'a' + rand() % 32;
    }
    str[len] = '\\0';
    cos_str_set(data, str);
}

unsigned long get_file_size(const char *file_path)
{
    unsigned long filesize = -1;
    struct stat statbuff;

    if(stat(file_path, &statbuff) < 0){
        return filesize;
    } else {
        filesize = statbuff.st_size;
    }

    return filesize;
}

void test_part_copy()
{
    cos_pool_t *p = NULL;
    cos_request_options_t *options = NULL;
    cos_string_t bucket;
    cos_string_t object;
    cos_string_t file;
    int is_cname = 0;
    cos_string_t upload_id;
    cos_list_upload_part_params_t *list_upload_part_params = NULL;
    cos_upload_part_copy_params_t *upload_part_copy_params1 = NULL;
    cos_upload_part_copy_params_t *upload_part_copy_params2 = NULL;
    cos_table_t *headers = NULL;
    cos_table_t *query_params = NULL;
    cos_table_t *resp_headers = NULL;
    cos_table_t *list_part_resp_headers = NULL;
    cos_list_t complete_part_list;
    cos_list_part_content_t *part_content = NULL;
    cos_complete_part_content_t *complete_content = NULL;
    cos_table_t *complete_resp_headers = NULL;
    cos_status_t *s = NULL;
    int part1 = 1;
    int part2 = 2;
    char *local_filename = "test_upload_part_copy.file";
    char *download_filename = "test_upload_part_copy.file.download";
    char *source_object_name = "cos_test_upload_part_copy_source_object";
    char *dest_object_name = "cos_test_upload_part_copy_dest_object";
    FILE *fd = NULL;
    cos_string_t download_file;
    cos_string_t dest_bucket;
    cos_string_t dest_object;
    int64_t range_start1 = 0;
    int64_t range_end1 = 6000000;
    int64_t range_start2 = 6000001;
    int64_t range_end2;
    cos_string_t data;

    cos_pool_create(&p, NULL);
    options = cos_request_options_create(p);

    // create multipart upload local file
    make_rand_string(p, 10 * 1024 * 1024, &data);
    fd = fopen(local_filename, "w");
    fwrite(data.data, sizeof(data.data[0]), data.len, fd);
    fclose(fd);

    init_test_request_options(options, is_cname);
    cos_str_set(&bucket, TEST_BUCKET_NAME);
    cos_str_set(&object, source_object_name);
    cos_str_set(&file, local_filename);
    s = cos_put_object_from_file(options, &bucket, &object, &file, NULL, &resp_head
    log_status(s);

    // Initialize multipart upload
    cos_str_set(&object, dest_object_name);
    s = cos_init_multipart_upload(options, &bucket, &object,
                                  &upload_id, NULL, &resp_headers);
    log_status(s);

    // Upload part copy 1
    upload_part_copy_params1 = cos_create_upload_part_copy_params(p);
    cos_str_set(&upload_part_copy_params1->copy_source, "bucket-appid.cn-south.myqc
    cos_str_set(&upload_part_copy_params1->dest_bucket, TEST_BUCKET_NAME);
    cos_str_set(&upload_part_copy_params1->dest_object, dest_object_name);
    cos_str_set(&upload_part_copy_params1->upload_id, upload_id.data);
    upload_part_copy_params1->part_num = part1;
    upload_part_copy_params1->range_start = range_start1;
    upload_part_copy_params1->range_end = range_end1;
    headers = cos_table_make(p, 0);
    s = cos_upload_part_copy(options, upload_part_copy_params1, headers, &resp_head
    log_status(s);
    printf("last modified:%s, etag:%s\\n", upload_part_copy_params1->rsp_content->l

    // Upload part copy 2
    resp_headers = NULL;
    range_end2 = get_file_size(local_filename) - 1;
    upload_part_copy_params2 = cos_create_upload_part_copy_params(p);
    cos_str_set(&upload_part_copy_params2->copy_source, "bucket-appid.cn-south.myqc
    cos_str_set(&upload_part_copy_params2->dest_bucket, TEST_BUCKET_NAME);
    cos_str_set(&upload_part_copy_params2->dest_object, dest_object_name);
    cos_str_set(&upload_part_copy_params2->upload_id, upload_id.data);
    upload_part_copy_params2->part_num = part2;
    upload_part_copy_params2->range_start = range_start2;
    upload_part_copy_params2->range_end = range_end2;
    headers = cos_table_make(p, 0);
    s = cos_upload_part_copy(options, upload_part_copy_params2, headers, &resp_head
    log_status(s);
    printf("last modified:%s, etag:%s\\n", upload_part_copy_params1->rsp_content->l

    // List parts
    list_upload_part_params = cos_create_list_upload_part_params(p);
    list_upload_part_params->max_ret = 10;
    cos_list_init(&complete_part_list);

    cos_str_set(&dest_bucket, TEST_BUCKET_NAME);
    cos_str_set(&dest_object, dest_object_name);
    s = cos_list_upload_part(options, &dest_bucket, &dest_object, &upload_id,
                             list_upload_part_params, &list_part_resp_headers);
    log_status(s);
    cos_list_for_each_entry(cos_list_part_content_t, part_content, &list_upload_par
        complete_content = cos_create_complete_part_content(p);
        cos_str_set(&complete_content->part_number, part_content->part_number.data)
        cos_str_set(&complete_content->etag, part_content->etag.data);
        cos_list_add_tail(&complete_content->node, &complete_part_list);
    }

    // Complete multipart upload
    headers = cos_table_make(p, 0);
    s = cos_complete_multipart_upload(options, &dest_bucket, &dest_object,
            &upload_id, &complete_part_list, headers, &complete_resp_headers);
    log_status(s);

    // Check whether the upload copy part content is equal to the local file
    headers = cos_table_make(p, 0);
    cos_str_set(&download_file, download_filename);
    s = cos_get_object_to_file(options, &dest_bucket, &dest_object, headers,
                               query_params, &download_file, &resp_headers);
    log_status(s);
    printf("local file len = %"APR_INT64_T_FMT", download file len = %"APR_INT64_T_
    remove(download_filename);
    remove(local_filename);
    cos_pool_destroy(p);

    printf("test part copy ok\\n");
}

int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    // Set the log level. Default value: `COS_LOG_WARN`
    cos_log_set_level(COS_LOG_WARN);

    // Set log output. Default value: `stderr`
    cos_log_set_output(NULL);

    test_part_copy();

    cos_http_io_deinitialize();
    return 0;
}
```

Moving an Object

Moving an object

Description

Object movement involves copying the source object to the target location and deleting the source object.
Since COS uses the bucket name (`Bucket`) and object key (`ObjectKey`) to identify objects, moving an object will change the object identifier. Currently, COS's C SDK does not provide a standalone API to change object identifiers. However, you can still move the object with a combination of basic operations (**object copy** and **object delete**).
Copying an object
Deleting an object

Sample

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"

// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
static char *TEST_ACCESS_KEY_ID;                // Your SecretId
static char *TEST_ACCESS_KEY_SECRET;            // Your SecretKey
// A unique user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";    // Your APPID
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";
// A unique identifier of an object stored in COS. For more information about objec
static char TEST_OBJECT_NAME1[] = "1.txt";
static char TEST_OBJECT_NAME2[] = "test2.dat";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    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);
```

```
}

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_move()
{
    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 src_object;
    cos_string_t src_endpoint;
    cos_table_t *resp_headers = NULL;

    // Create a memory pool
    cos_pool_create(&p, NULL);

    // Initialize the request options
    options = cos_request_options_create(p);
    init_test_request_options(options, is_cname);
    cos_str_set(&bucket, TEST_BUCKET_NAME);

    // Set object replication
    cos_str_set(&object, TEST_OBJECT_NAME1);
    cos_str_set(&src_endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&src_object, TEST_OBJECT_NAME2);

    cos_copy_object_params_t *params = NULL;
    params = cos_create_copy_object_params(p);
    s = cos_copy_object(options, &bucket, &src_object, &src_endpoint, &bucket, &obj
    log_status(s);
    if (cos_status_is_ok(s)) {
        s = cos_delete_object(options, &bucket, &src_object, &resp_headers);
        log_status(s);
        printf("move object succeeded\\n");
    } else {
        printf("move object failed\\n");
    }

    // Destroy the memory pool.
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    // Set the log level. Default value: `COS_LOG_WARN`
    cos_log_set_level(COS_LOG_WARN);

    // Set log output. Default value: `stderr`
    cos_log_set_output(NULL);

    test_move();

    cos_http_io_deinitialize();

    return 0;
}
```

Downloading Objects

Last updated：2024-02-01 18:01:21

Overview

This document provides an overview of APIs and SDK code samples related to object downloads.

Simple operations

API	Operation	Description
GET Object	Downloading an object	Downloads an object to the local file system.

Advanced APIs (Recommended)

Downloading an object (checkpoint restart)

Description

The multipart download API automatically downloads data concurrently with `Range` according to the object size. The part size is 1,048,576 (1 MB) by default and can be adjusted via the `part_size` parameter.

Method prototype

```
cos_status_t *cos_resumable_download_file(cos_request_options_t *options,
                                          cos_string_t *bucket,
                                          cos_string_t *object,
                                          cos_string_t *filepath,
                                          cos_table_t *headers,
                                          cos_table_t *params,
                                          cos_resumable_clt_params_t *clt_params,
                                          cos_progress_callback progress_callback);
```

Parameter description

Parameter	Description	Type
options	COS request options	Struct
bucket	Bucket name in the format: `BucketName-APPID`	String
object	Object name	String

filepath	The local file name of the object	String
headers	Headers attached to the COS request	Struct
params	Parameters for the COS request	Struct
clt_params	Control parameters for the download operation	Struct
part_size	Part size in bytes. If you specify the part size to be below 4,194,304 (4 MB), the system will divide your data based on the part size 4,194,304 (4 MB).	Int
thread_num	Number of threads, that is, size of the thread pool. Default: 1	Int
enable_checkpoint	Indicates whether to enable checkpoint restart	Int
checkpoint_path	Indicates the file path for which the upload progress is saved when checkpoint restart is enabled. Default: `<filepath>.cp` , where `filepath` is the local file name of the object	String
progress_callback	Callback function for the download progress	Function

Response

Response Parameter	Description	Type
code	Error code	Int
error_code	Error code content	String
error_msg	Error code description	String
req_id	Request message ID	String

Sample

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"

// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
static char *TEST_ACCESS_KEY_ID;                // Your SecretId
static char *TEST_ACCESS_KEY_SECRET;            // Your SecretKey
// A unique user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";    // Your APPID
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
```

```
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";
static char TEST_DOWNLOAD_NAME4[] = "multipart_download.dat";
static char TEST_MULTIPART_OBJECT4[] = "multipart4.dat";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    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);
}

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_resumable()
{
    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 filepath;
    cos_resumable_clt_params_t *clt_params;

    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);
    cos_str_set(&object, TEST_MULTIPART_OBJECT4);
    cos_str_set(&filepath, TEST_DOWNLOAD_NAME4);

    clt_params = cos_create_resumable_clt_params_content(p, 5*1024*1024, 3, COS_FAL
    s = cos_resumable_download_file(options, &bucket, &object, &filepath, NULL, NUL
    log_status(s);

    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    // Set the log level. Default value: `COS_LOG_WARN`
    cos_log_set_level(COS_LOG_WARN);

    // Set log output. Default value: `stderr`
    cos_log_set_output(NULL);

    test_resumable();

    cos_http_io_deinitialize();

    return 0;
}
```

Batch downloading files (downloading a COS directory)

Description

This API is used to download a COS directory and the files in it to the local disk.

Sample

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"

// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
static char *TEST_ACCESS_KEY_ID;                // Your SecretId
static char *TEST_ACCESS_KEY_SECRET;            // Your SecretKey
// A unique user-level resource identifier for COS access. It can be obtained at ht
```

```
static char TEST_APPID[] = "<APPID>";    // Your APPID
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    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);
}

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_download_directory()
{
    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 file_name;
    cos_string_t suffix = cos_string("/");
    cos_table_t *resp_headers;
    cos_table_t *headers = NULL;
    cos_table_t *params = NULL;
    int is_truncated = 1;
    cos_string_t marker;
    apr_status_t status;

    // Initialize the 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 object (get bucket)
    cos_list_object_params_t *list_params = NULL;
    list_params = cos_create_list_object_params(p);
    cos_str_set(&list_params->prefix, "folder/");   // Use your own directory name
    cos_str_set(&marker, "");
    while (is_truncated) {
        list_params->marker = marker;
        s = cos_list_object(options, &bucket, list_params, &resp_headers);
        log_status(s);
        if (!cos_status_is_ok(s)) {
            printf("list object failed, req_id:%s\\n", s->req_id);
            break;
        }
        cos_list_object_content_t *content = NULL;
        cos_list_for_each_entry(cos_list_object_content_t, content, &list_params->o
            cos_str_set(&file_name, content->key.data);
            if (cos_ends_with(&content->key, &suffix)) {
                // If you need to create the directory first, you can modify the 0x
                status = apr_dir_make(content->key.data, 0x0755, options->pool);
                if (status != APR_SUCCESS && !APR_STATUS_IS_EEXIST(status)) {
                    printf("mkdir: %s failed, status: %d\\n", content->key.data, st
                }
            } else {
                // Download the object to a local directory, which is the current p
                s = cos_get_object_to_file(options, &bucket, &content->key, headers
                if (!cos_status_is_ok(s)) {
                    printf("get object[%s] failed, req_id:%s\\n", content->key.data
                }
            }
        }
        is_truncated = list_params->truncated;
        marker = list_params->next_marker;
    }

    // Destroy the memory pool.
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");
    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    // Set the log level. Default value: `COS_LOG_WARN`
    cos_log_set_level(COS_LOG_WARN);

    // Set log output. Default value: `stderr`
    cos_log_set_output(NULL);

    test_download_directory();

    cos_http_io_deinitialize();

    return 0;
}
```

Simple Operations

Downloading an object

Description

This API is used to download an object to the local file system. This operation requires that you have read permission for the object, or that the object has public read permission enabled.

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);
```

Parameter description

Parameter	Description	Type
options	COS request options	Struct
bucket	Bucket name in the format: `BucketName-APPID`	String

object	Object name	String
headers	Additional headers of a COS request	Struct
params	Parameters for the COS request operation	Struct
filename	Filename of the local object before it is uploaded to COS	String
resp_headers	Returns the HTTP response headers	Struct

Response description

Response Parameter	Description	Type
code	Error code	Int
error_code	Error code content	String
error_msg	Error code description	String
req_id	Request message ID	String

Sample: simple download

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"

// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
static char *TEST_ACCESS_KEY_ID;                    // Your SecretId
static char *TEST_ACCESS_KEY_SECRET;                // Your SecretKey
// A unique user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";     // Your APPID
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";
// A unique identifier of an object stored in COS. For more information about objec
static char TEST_OBJECT_NAME1[] = "1.txt";
static char TEST_DOWNLOAD_NAME[] = "download_test3.dat";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    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);
}

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_download_object()
{
    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 a memory pool
    cos_pool_create(&p, NULL);

    // Initialize the request options
    options = cos_request_options_create(p);
    init_test_request_options(options, is_cname);
    cos_str_set(&bucket, TEST_BUCKET_NAME);

    // Get the object
    cos_str_set(&file, TEST_DOWNLOAD_NAME);
    cos_str_set(&object, TEST_OBJECT_NAME1);
    s = cos_get_object_to_file(options, &bucket, &object, NULL, NULL, &file, &resp_
    if (cos_status_is_ok(s)) {
        printf("get object succeeded\\n");
    } else {
        printf("get object failed\\n");
    }
    // Destroy the memory pool.
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    // Set the log level. Default value: `COS_LOG_WARN`
    cos_log_set_level(COS_LOG_WARN);

    // Set log output. Default value: `stderr`
    cos_log_set_output(NULL);

    test_download_object();

    cos_http_io_deinitialize();

    return 0;
}
```

Listing Objects

Last updated：2024-02-01 18:01:21

Overview

This document provides an overview of APIs and SDK code samples related to object listing.

API	Operation	Description
GET Bucket (List Objects)	Querying objects	Queries some or all the objects in a bucket.

Querying an Object List

Description

This API is used to query some or all the objects in a bucket.

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);
```

Parameter description

Parameter	Description	Type
options	COS request options	Struct
Bucket	Bucket name in the format of `BucketName-APPID`	String
params	Parameters for the list operation	Struct
encoding_type	Specifies the encoding type of the returned value	String
prefix	Prefix to be matched, which is used to specify the prefix address of the files to be returned	String
marker	Marks the starting point of the object list; by default, entries are listed in UTF-8 binary order starting from this marker	String

delimiter	A query separator, used to group object keys	String
max_ret	The maximum number of returned entries per request; the default value is 1000	Struct
truncated	Indicates whether the returned entry is truncated. Valid value: `true` or `false`	Boolean
next_marker	Marks the starting point of the next entry if the returned entry is truncated	String
object_list	Object list returned by the `Get Bucket` operation	Struct
key	Name (key) of the object returned by the `Get Bucket` operation	Struct
last_modified	The last modified time of the object returned by the `Get Bucket` operation	Struct
etag	SHA-1 check value of the object returned by the `Get Bucket` operation	Struct
size	The size in bytes of the object returned by the `Get Bucket` operation	Struct
owner_id	UID of the owner of the object returned by the `Get Bucket` operation	Struct
storage_class	The storage class of the object returned by the `Get Bucket` operation	Struct
common_prefix_list	The identical paths between a particular prefix and the delimiter are grouped together and defined as a common prefix	Struct
resp_headers	Returns the HTTP response headers	Struct

Response description

Response Parameter	Description	Type
code	Error code	Int
error_code	Error code content	String
error_msg	Error code description	String
req_id	Request message ID	String

Sample 1. Listing objects on the first page

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"
```

```
// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
static char *TEST_ACCESS_KEY_ID;                // Your SecretId
static char *TEST_ACCESS_KEY_SECRET;            // Your SecretKey
// A unique user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";    // Your APPID
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_list_objects()
{
    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 a memory pool
    cos_pool_create(&p, NULL);

    // Initialize the request options
    options = cos_request_options_create(p);
    init_test_request_options(options, is_cname);
    cos_str_set(&bucket, TEST_BUCKET_NAME);

    // Get the object list
    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->o
           printf("object: %.*s\\n", content->key.len, content->key.data);
        }
    } else {
        printf("list object failed\\n");
    }

    // Destroy the memory pool.
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    // Set the log level. Default value: `COS_LOG_WARN`
    cos_log_set_level(COS_LOG_WARN);

    // Set log output. Default value: `stderr`
    cos_log_set_output(NULL);

    test_list_objects();

    cos_http_io_deinitialize();

    return 0;
}
```

Sample 2. Listing the objects in a directory

COS does not have the concept of folder, but you can use slashes (/) as the delimiter to stimulate folders.

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"

// `endpoint` is the COS access domain name. For more information, see https://intl
```

```
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
static char *TEST_ACCESS_KEY_ID;                // Your SecretId
static char *TEST_ACCESS_KEY_SECRET;            // Your SecretKey
// A unique user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";    // Your APPID
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_list_directory()
{
    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;
    int is_truncated = 1;
    cos_string_t marker;

    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 object (get bucket)
    cos_list_object_params_t *list_params = NULL;
    list_params = cos_create_list_object_params(p);
    // The prefix indicates that the key of the object to be listed must start with
    cos_str_set(&list_params->prefix, "folder/");
    // Set the delimiter to "/" to list objects in the current directory. To list a
    cos_str_set(&list_params->delimiter, "/");
    // Set the maximum number of traversed objects (up to 1,000 per listobject requ
    list_params->max_ret = 1000;
    cos_str_set(&marker, "");
    while (is_truncated) {
        list_params->marker = marker;
        s = cos_list_object(options, &bucket, list_params, &resp_headers);
        if (!cos_status_is_ok(s)) {
            printf("list object failed, req_id:%s\\n", s->req_id);
            break;
        }
        // `list_params->object_list` returns the following objects
        cos_list_object_content_t *content = NULL;
        cos_list_for_each_entry(cos_list_object_content_t, content, &list_params->o
            printf("object: %s\\n", content->key.data);
        }
        // `list_params->common_prefix_list` indicates paths that end with the deli
        cos_list_object_common_prefix_t *common_prefix = NULL;
        cos_list_for_each_entry(cos_list_object_common_prefix_t, common_prefix, &li
            printf("common prefix: %s\\n", common_prefix->prefix.data);
        }

        is_truncated = list_params->truncated;
        marker = list_params->next_marker;
    }
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    // Set the log level. Default value: `COS_LOG_WARN`
    cos_log_set_level(COS_LOG_WARN);

    // Set log output. Default value: `stderr`
    cos_log_set_output(NULL);

    test_list_directory();

    cos_http_io_deinitialize();
    return 0;
}
```

Sample 3. Listing all objects in a bucket

A maximum of 1,000 objects can be listed at a time. When the number of objects in a bucket exceeds 1,000, you need to list all objects in the bucket repeatedly.

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"

// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
static char *TEST_ACCESS_KEY_ID;                    // Your SecretId
static char *TEST_ACCESS_KEY_SECRET;                // Your SecretKey
// A unique user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";     // Your APPID
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_list_all_objects()
{
    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;
    int is_truncated = 1;
    cos_string_t marker;

    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 object (get bucket)
    cos_list_object_params_t *list_params = NULL;
    list_params = cos_create_list_object_params(p);
    // Set the maximum number of traversed objects (up to 1,000 per listobject requ
    list_params->max_ret = 1000;
    cos_str_set(&marker, "");
    while (is_truncated) {
        list_params->marker = marker;
        cos_list_init(&list_params->object_list);
        s = cos_list_object(options, &bucket, list_params, &resp_headers);
        if (!cos_status_is_ok(s)) {
            printf("list object failed, req_id:%s\\n", s->req_id);
            break;
        }
        // `list_params->object_list` returns the following objects
        cos_list_object_content_t *content = NULL;
        cos_list_for_each_entry(cos_list_object_content_t, content, &list_params->o
            printf("object: %s\\n", content->key.data);
        }

        is_truncated = list_params->truncated;
        marker = list_params->next_marker;
    }
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    // Set the log level. Default value: `COS_LOG_WARN`
    cos_log_set_level(COS_LOG_WARN);
    // Set log output. Default value: `stderr`
    cos_log_set_output(NULL);

    test_list_all_objects();

    cos_http_io_deinitialize();

    return 0;
}
```

Deleting Objects

Last updated：2024-02-01 18:01:21

Overview

This document provides an overview of APIs and SDK code samples related to object deletion.

API	Operation	Description
DELETE Object	Deleting an object	Deletes a specified object from a bucket.
DELETE Multiple Objects	Deleting multiple objects	Deletes multiple objects from a bucket.

Deleting a Single Object

Description

This API is used to delete a specified object from a bucket.

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);
```

Parameter description

Parameter	Description	Type
options	COS request options	Struct
bucket	Bucket name in the format: `BucketName-APPID`	String
object	Object name	String
resp_headers	Returns the HTTP response headers	Struct

Response description

Response Parameter	Description	Type

code	Error code	Int
error_code	Error code content	String
error_msg	Error code description	String
req_id	Request message ID	String

Sample 1. Deleting an object

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"

// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
static char *TEST_ACCESS_KEY_ID;                    // Your SecretId
static char *TEST_ACCESS_KEY_SECRET;                // Your SecretKey
// A unique user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";    // Your APPID
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";
// A unique identifier of an object stored in COS. For more information about objec
static char TEST_OBJECT_NAME1[] = "1.txt";

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_delete_object()
{
    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 a memory pool
    cos_pool_create(&p, NULL);

    // Initialize the request options
    options = cos_request_options_create(p);
    init_test_request_options(options, is_cname);
    cos_str_set(&bucket, TEST_BUCKET_NAME);

    // Delete the single object
    cos_str_set(&object, TEST_OBJECT_NAME1);
    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 the memory pool.
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    // Set the log level. Default value: `COS_LOG_WARN`
    cos_log_set_level(COS_LOG_WARN);

    // Set log output. Default value: `stderr`
    cos_log_set_output(NULL);

    test_delete_object();

    cos_http_io_deinitialize();
    return 0;
}
```

Sample 2. Deleting a folder

This request does not delete objects in the folder but only the specified key.

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"

// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
static char *TEST_ACCESS_KEY_ID;                // Your SecretId
static char *TEST_ACCESS_KEY_SECRET;            // Your SecretKey
// A unique user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";    // Your APPID
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_delete_dir()
{
    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 a memory pool
    cos_pool_create(&p, NULL);

    // Initialize the request options
    options = cos_request_options_create(p);
    init_test_request_options(options, is_cname);
    cos_str_set(&bucket, TEST_BUCKET_NAME);

    // Delete the directory object
    cos_str_set(&object, "folder/");
    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 the memory pool.
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    // Set the log level. Default value: `COS_LOG_WARN`
    cos_log_set_level(COS_LOG_WARN);

    // Set log output. Default value: `stderr`
    cos_log_set_output(NULL);

    test_delete_dir();

    cos_http_io_deinitialize();

    return 0;
}
```

Deleting Multiple Objects

Description

This API is used to delete multiple objects from a bucket in a single request. It can delete up to 1,000 objects in a single request. This API supports two response modes: Verbose and Quiet. Verbose mode returns the information on the deletion of each object, whereas Quiet mode only returns the information on the objects for which deletion errors were reported.

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);
```

Parameter description

Parameter	Description	Type
options	COS request options	Struct
bucket	Bucket name in the format of `BucketName-APPID`	String
object_list	The list of objects to be deleted	Struct
key	Name of the object to be deleted	String
is_quiet	Indicates whether Quiet mode is enabled. True(1): Quiet mode is enabled; False(0): Verbose mode is enabled. The default value is False(0).	Boolean
resp_headers	Returns the HTTP response headers	Struct
deleted_object_list	The list of deleted objects	Struct

Response description

Response Parameter	Description	Type
code	Error code	Int
error_code	Error code content	String
error_msg	Error code description	String

req_id	Request message ID	String

Sample 1. Deleting multiple objects

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"

// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
static char *TEST_ACCESS_KEY_ID;                    // Your SecretId
static char *TEST_ACCESS_KEY_SECRET;                // Your SecretKey
// A unique user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";    // Your APPID
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";
// A unique identifier of an object stored in COS. For more information about objec
static char TEST_OBJECT_NAME2[] = "test2.dat";
static char TEST_OBJECT_NAME3[] = "test3.dat";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    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);
}

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_delete_objects()
```

```
{
    cos_pool_t *p = NULL;
    int is_cname = 0;
    cos_string_t bucket;
    cos_status_t *s = NULL;
    cos_table_t *resp_headers = NULL;
    cos_request_options_t *options = NULL;
    char *object_name1 = TEST_OBJECT_NAME2;
    char *object_name2 = TEST_OBJECT_NAME3;
    cos_object_key_t *content1 = NULL;
    cos_object_key_t *content2 = NULL;
    cos_list_t object_list;
    cos_list_t deleted_object_list;
    int is_quiet = COS_TRUE;

    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);

    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);

    s = cos_delete_objects(options, &bucket, &object_list, is_quiet,
        &resp_headers, &deleted_object_list);
    log_status(s);

    cos_pool_destroy(p);

    if (cos_status_is_ok(s)) {
        printf("delete objects succeeded\\n");
    } else {
        printf("delete objects failed\\n");
    }
}

int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");
    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    // Set the log level. Default value: `COS_LOG_WARN`
    cos_log_set_level(COS_LOG_WARN);

    // Set log output. Default value: `stderr`
    cos_log_set_output(NULL);

    test_delete_objects();

    cos_http_io_deinitialize();

    return 0;
}
```

Sample 2. Deleting a folder and the objects contained

COS does not have the concept of folder, but you can use slashes (/) as the delimiter to stimulate folders.
In COS, deleting a folder and the objects contained actually means deleting objects that have the same specified prefix. Currently, COS's C SDK does not provide a standalone API to perform this operation. However, you can still do so with a combination of basic operations (**query object list** and **batch delete objects**).

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"

// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
static char *TEST_ACCESS_KEY_ID;                // Your SecretId
static char *TEST_ACCESS_KEY_SECRET;            // Your SecretKey
// A unique user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";    // Your APPID
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    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);
}
```

```
void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_delete_directory()
{
    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;
    int is_truncated = 1;
    cos_string_t marker;
    cos_list_t deleted_object_list;
    int is_quiet = COS_TRUE;

    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 object (get bucket)
    cos_list_object_params_t *list_params = NULL;
    list_params = cos_create_list_object_params(p);
    cos_str_set(&list_params->prefix, "folder/");
    cos_str_set(&marker, "");
    while (is_truncated) {
        list_params->marker = marker;
        s = cos_list_object(options, &bucket, list_params, &resp_headers);
        if (!cos_status_is_ok(s)) {
            printf("list object failed, req_id:%s\\n", s->req_id);
            break;
        }
        s = cos_delete_objects(options, &bucket, &list_params->object_list, is_quie
        log_status(s);
        if (!cos_status_is_ok(s)) {
            printf("delete objects failed, req_id:%s\\n", s->req_id);
        }

        is_truncated = list_params->truncated;
        marker = list_params->next_marker;
    }
    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    // Set the log level. Default value: `COS_LOG_WARN`
    cos_log_set_level(COS_LOG_WARN);

    // Set log output. Default value: `stderr`
    cos_log_set_output(NULL);

    test_delete_directory();

    cos_http_io_deinitialize();

    return 0;
}
```

Object Access URL

Last updated：2024-02-01 18:01:21

Overview

This document provides an overview of the API and sample code for getting an object access URL.

Getting an Object Access URL

Description

The API is used to get object access URLs for anonymous download or distribution.

Method prototype

```
const char *cos_gen_object_url(const cos_request_options_t *options,
                               const cos_string_t *bucket,
                               const cos_string_t *object)
```

Parameter description

Parameter	Description	Type	Required
options	COS request options	Struct	Yes
Bucket	Bucket name in the format of `BucketName-APPID`	Struct	Yes
object	Object key, the unique identifier of an object in a bucket. For example, if the object endpoint is `examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/doc/pic.jpg` , its object key is `doc/pic.jpg`	String	Yes

Response description

An object access URL is returned upon success.

Sample request

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"
```

```
// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
static char *TEST_ACCESS_KEY_ID;                // Your SecretId
static char *TEST_ACCESS_KEY_SECRET;            // Your SecretKey
// A unique user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";    // Your APPID
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";
// A unique identifier of an object stored in COS. For more information about objec
static char TEST_OBJECT_NAME1[] = "1.txt";

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_gen_object_url()
{
    cos_pool_t *p = NULL;
    int is_cname = 0;
    cos_request_options_t *options = NULL;
    cos_string_t bucket;
    cos_string_t object;

    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);
    cos_str_set(&object, TEST_OBJECT_NAME1);

    printf("url:%s\\n", cos_gen_object_url(options, &bucket, &object));

    cos_pool_destroy(p);
}
```

```
int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    // Set the log level. Default value: `COS_LOG_WARN`
    cos_log_set_level(COS_LOG_WARN);

    // Set log output. Default value: `stderr`
    cos_log_set_output(NULL);

    test_gen_object_url();

    cos_http_io_deinitialize();

    return 0;
}
```

Restoring Archived Objects

Last updated：2024-02-01 18:01:21

Overview

This document provides an overview of APIs and SDK code samples related to restoring an archived object.

API	Operation	Description
POST Object restore	Restoring an archived object	Restores an archived object for access.

Restoring an Archived Object

Description

This API is used to restore an archived object for access.

Method prototype

```
cos_status_t *cos_post_object_restore(const cos_request_options_t *options,
                                      const cos_string_t *bucket,
                                      const cos_string_t *object,
                                      cos_object_restore_params_t *restore_params,
                                      cos_table_t *headers,
                                      cos_table_t *params,
                                      cos_table_t **resp_headers);
```

Parameter description

Parameter	Description	Type
options	COS request options	Struct
bucket	Bucket name in the format of `BucketName-APPID`	String
object	Object name	String
restore_params	Parameters for the `Post Object Restore` operation	Struct
days	The number of days before a temporary copy restored using `Post Object Restore` expires	Int

tier	Specifies one of the three CAS restoration modes for Post Object Restore: `Expedited` , `Standard` , `Bulk`	String
headers	Headers attached to the COS request	Struct
params	Parameters for the COS request operation	Struct
resp_headers	Returns the HTTP response headers	Struct

Response description

Response Parameter	Description	Type
code	Error code	Int
error_code	Error code content	String
error_msg	Error code description	String
req_id	Request message ID	String

Sample

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"

// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
static char *TEST_ACCESS_KEY_ID;                // Your SecretId
static char *TEST_ACCESS_KEY_SECRET;            // Your SecretKey
// A unique user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";    // Your APPID
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";

void log_status(cos_status_t *s)
{
    cos_warn_log("status->code: %d", s->code);
    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);
}

void init_test_config(cos_config_t *config, int is_cname)
```

```
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_object_restore()
{
    cos_pool_t *p = NULL;
    cos_string_t bucket;
    cos_string_t object;
    int is_cname = 0;
    cos_table_t *resp_headers = NULL;
    cos_request_options_t *options = NULL;
    cos_status_t *s = NULL;

    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);
    cos_str_set(&object, "test_restore.dat");

    cos_object_restore_params_t *restore_params = cos_create_object_restore_params(
    restore_params->days = 30;
    cos_str_set(&restore_params->tier, "Standard");
    s = cos_post_object_restore(options, &bucket, &object, restore_params, NULL, NU
    log_status(s);

    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
        exit(1);
    }

    // Set the log level. Default value: `COS_LOG_WARN`
    cos_log_set_level(COS_LOG_WARN);

    // Set log output. Default value: `stderr`
    cos_log_set_output(NULL);

    test_object_restore();

    cos_http_io_deinitialize();

    return 0;
}
```

Querying Object Metadata

Last updated：2024-02-01 18:01:21

Overview

This document provides an overview of APIs and SDK code samples related to querying object metadata.

API	Operation	Description
HEAD Object	Querying object metadata	Queries the metadata of an object

Querying Object Metadata

Description

This API is used to query the metadata of an 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);
```

Parameter description

Parameter	Description	Type
options	COS request options	Struct
bucket	Bucket name in the format: `BucketName-APPID`	String
object	Object name	String
headers	Additional headers of a COS request	Struct
resp_headers	Returns the HTTP response headers	Struct

Response description

Response Parameter	Description	Type

code	Error code	Int
error_code	Error code content	String
error_msg	Error code description	String
req_id	Request message ID	String

Sample

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"

// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
static char *TEST_ACCESS_KEY_ID;                // Your SecretId
static char *TEST_ACCESS_KEY_SECRET;            // Your SecretKey
// A unique user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";    // Your APPID
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";
// A unique identifier of an object stored in COS. For more information about objec
static char TEST_OBJECT_NAME1[] = "1.txt";

static void print_headers(cos_table_t *headers)
{
    const cos_array_header_t *tarr;
    const cos_table_entry_t *telts;
    int i = 0;

    if (apr_is_empty_table(headers)) {
        return;
    }

    tarr = cos_table_elts(headers);
    telts = (cos_table_entry_t*)tarr->elts;

    printf("headers:\\n");
    for (; i < tarr->nelts; i++) {
        telts = (cos_table_entry_t*)(tarr->elts + i * tarr->elt_size);
        printf("%s: %s\\n", telts->key, telts->val);
    }
}
```

```
void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_head_object()
{
    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 a memory pool
    cos_pool_create(&p, NULL);

    // Initialize the request options
    options = cos_request_options_create(p);
    init_test_request_options(options, is_cname);
    cos_str_set(&bucket, TEST_BUCKET_NAME);

    // Get object metadata
    cos_str_set(&object, TEST_OBJECT_NAME1);
    s = cos_head_object(options, &bucket, &object, NULL, &resp_headers);
    print_headers(resp_headers);
    if (cos_status_is_ok(s)) {
        printf("head object succeeded\\n");
    } else {
        printf("head object failed\\n");
    }

    // Destroy the memory pool.
    cos_pool_destroy(p);
}
```

```
int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    // Set the log level. Default value: `COS_LOG_WARN`
    cos_log_set_level(COS_LOG_WARN);

    // Set log output. Default value: `stderr`
    cos_log_set_output(NULL);

    test_head_object();

    cos_http_io_deinitialize();

    return 0;
}
```

Getting Pre-Signed URLs

Last updated：2024-02-01 18:01:21

Overview

The C SDK provides APIs to obtain pre-signed URLs. For detailed directions, see the description and examples below.

Note:

You are advised to use a temporary key to generate a pre-signed URL for the security of your requests such as uploads and downloads. When you apply for a temporary key, follow the Principle of Least Privilege to avoid leaking resources besides your buckets and objects.

If you need to use a permanent key to generate a pre-signed URL, you are advised to limit the permission of the permanent key to uploads and downloads only to avoid risks.

Getting a Pre-signed Request URL

Generating pre-signed URL

Description

This API is used to generate a pre-signed URL.

Method prototype

```
int cos_gen_presigned_url(const cos_request_options_t *options,
                          const cos_string_t *bucket,
                          const cos_string_t *object,
                          const int64_t expire,
                          http_method_e method,
                          cos_string_t *presigned_url);
```

Parameter description

Parameter	Description	Type
options	COS request options	Struct
bucket	Bucket name in the format of `BucketName-APPID`	String
object	Object name	String

expire	Validity time of the signature, in seconds	Int
method	HTTP request method. Enumerated values: `HTTP_GET` , `HTTP_HEAD` , `HTTP_PUT` , `HTTP_POST` , `HTTP_DELETE`	Enum
presigned_url	The generated pre-signed URL	String

Response description

Response	Description	Type
code	Error code	Int

Pre-signed request samples

You can obtain a pre-signed URL by setting a permanent or temporary key using the `options` parameter.

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"

// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
static char *TEST_ACCESS_KEY_ID;                    // Your SecretId
static char *TEST_ACCESS_KEY_SECRET;                // Your SecretKey
// A unique user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";     // Your APPID
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";
// A unique identifier of an object stored in COS. For more information about objec
static char TEST_OBJECT_NAME1[] = "1.txt";

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_presigned_url()
{
    cos_pool_t *p = NULL;
    int is_cname = 0;
    cos_request_options_t *options = NULL;
    cos_string_t bucket;
    cos_string_t object;
    cos_string_t presigned_url;

    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);
    cos_str_set(&object, TEST_OBJECT_NAME1);

    cos_gen_presigned_url(options, &bucket, &object, 300, HTTP_GET, &presigned_url)
    printf("presigned_url: %s\\n", presigned_url.data);

    cos_pool_destroy(p);

}

int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    // Set the log level. Default value: `COS_LOG_WARN`
    cos_log_set_level(COS_LOG_WARN);

    // Set log output. Default value: `stderr`
    cos_log_set_output(NULL);

    test_presigned_url();

    cos_http_io_deinitialize();

    return 0;
}
```

Securely Getting a Pre-signed Request URL

Securely generating a pre-signed URL

Description

This API is used to securely generate a pre-signed URL.

Method prototype

```
int cos_gen_presigned_url_safe(const cos_request_options_t *options,
                               const cos_string_t *bucket,
                               const cos_string_t *object,
                               const int64_t expire,
                               http_method_e method,
                               cos_table_t *headers,
                               cos_table_t *params,
                               int sign_host,
                               cos_string_t *presigned_url);
```

Parameter description

Parameter	Description	Type
options	COS request options	Struct
bucket	Bucket name in the format of `BucketName-APPID`	String
object	Object name	String
expire	Validity time of the signature, in seconds	Int
method	HTTP request method. Enumerated values: `HTTP_GET` , `HTTP_HEAD` , `HTTP_PUT` , `HTTP_POST` , `HTTP_DELETE`	Enum
headers	Additional headers of a COS request	Struct
params	Parameters for the COS request	Struct
sign_host	Whether to sign the signature to the host header. You are strongly advised to enable it for security’s sake.	Int
presigned_url	The generated pre-signed URL	String

Response description

Response	Description	Type
code	Error code	Int

Sample request 1. Generate a pre-signed URL with `host` in the signature

You can obtain a pre-signed URL by setting a permanent or temporary key using the `options` parameter.

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"

// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
static char *TEST_ACCESS_KEY_ID;                // Your SecretId
static char *TEST_ACCESS_KEY_SECRET;            // Your SecretKey
// A unique user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";    // Your APPID
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";
// A unique identifier of an object stored in COS. For more information about objec
static char TEST_OBJECT_NAME1[] = "1.txt";

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_safe_presigned_url()
{
    cos_pool_t *p = NULL;
    int is_cname = 0;
    cos_request_options_t *options = NULL;
    cos_string_t bucket;
    cos_string_t object;
    cos_string_t presigned_url;
    cos_table_t *params = NULL;
    cos_table_t *headers = NULL;
    int sign_host = 1;

    cos_pool_create(&p, NULL);
    options = cos_request_options_create(p);
    init_test_request_options(options, is_cname);
    /* You can use a temporary key by setting `sts_token`. When you use a temporary
    //cos_str_set(&options->config->sts_token, "MyTokenString");
    cos_str_set(&bucket, TEST_BUCKET_NAME);
    cos_str_set(&object, TEST_OBJECT_NAME1);

    // You are strongly advised to set `sign_host` to `1` to add the `host` header
    cos_gen_presigned_url_safe(options, &bucket, &object, 300, HTTP_GET, headers, p
    printf("presigned_url_safe: %s\\n", presigned_url.data);

    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    // Set the log level. Default value: `COS_LOG_WARN`
    cos_log_set_level(COS_LOG_WARN);

    // Set log output. Default value: `stderr`
    cos_log_set_output(NULL);

    test_safe_presigned_url();

    cos_http_io_deinitialize();

    return 0;
}
```

Sample request 2. Generate the pre-signed URL with the signature containing `request param` and `request header`

You can obtain a pre-signed URL by setting a permanent or temporary key using the `options` parameter.

```
#include "cos_http_io.h"
#include "cos_api.h"
#include "cos_log.h"

// `endpoint` is the COS access domain name. For more information, see https://intl
static char TEST_COS_ENDPOINT[] = "cos.ap-guangzhou.myqcloud.com";
// A developer-owned secret ID/key used for the project. It can be obtained at http
static char *TEST_ACCESS_KEY_ID;                // Your SecretId
static char *TEST_ACCESS_KEY_SECRET;            // Your SecretKey
// A unique user-level resource identifier for COS access. It can be obtained at ht
static char TEST_APPID[] = "<APPID>";    // Your APPID
// COS bucket name, in the format of [bucket]-[appid], for example `mybucket-125366
static char TEST_BUCKET_NAME[] = "<bucketname-appid>";
// A unique identifier of an object stored in COS. For more information about objec
static char TEST_OBJECT_NAME1[] = "1.txt";

void init_test_config(cos_config_t *config, int is_cname)
{
    cos_str_set(&config->endpoint, TEST_COS_ENDPOINT);
    cos_str_set(&config->access_key_id, TEST_ACCESS_KEY_ID);
    cos_str_set(&config->access_key_secret, TEST_ACCESS_KEY_SECRET);
    cos_str_set(&config->appid, TEST_APPID);
    config->is_cname = is_cname;
}

void init_test_request_options(cos_request_options_t *options, int is_cname)
{
    options->config = cos_config_create(options->pool);
    init_test_config(options->config, is_cname);
    options->ctl = cos_http_controller_create(options->pool, 0);
}

void test_safe_presigned_url()
{
    cos_pool_t *p = NULL;
    int is_cname = 0;
    cos_request_options_t *options = NULL;
    cos_string_t bucket;
    cos_string_t object;
    cos_string_t presigned_url;
    cos_table_t *params = NULL;
    cos_table_t *headers = NULL;
    int sign_host = 1;

    cos_pool_create(&p, NULL);
    options = cos_request_options_create(p);
    init_test_request_options(options, is_cname);
    /* You can use a temporary key by setting `sts_token`. When you use a temporary
    //cos_str_set(&options->config->sts_token, "MyTokenString");
    cos_str_set(&bucket, TEST_BUCKET_NAME);
    cos_str_set(&object, TEST_OBJECT_NAME1);

    // Add your `params` and `headers`
    params = cos_table_make(options->pool, 0);
    //cos_table_add(params, "param1", "value");
    headers = cos_table_make(options->pool, 0);
    //cos_table_add(headers, "header1", "value");

    // You are strongly advised to set `sign_host` to `1` to add the `host` header
    cos_gen_presigned_url_safe(options, &bucket, &object, 300, HTTP_GET, headers, p
    printf("presigned_url_safe: %s\\n", presigned_url.data);

    cos_pool_destroy(p);
}

int main(int argc, char *argv[])
{
    // Get SecretId and SecretKey from environment variables
    TEST_ACCESS_KEY_ID     = getenv("COS_SECRETID");
    TEST_ACCESS_KEY_SECRET = getenv("COS_SECRETKEY");

    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }

    // Set the log level. Default value: `COS_LOG_WARN`
    cos_log_set_level(COS_LOG_WARN);

    // Set log output. Default value: `stderr`
    cos_log_set_output(NULL);

    test_safe_presigned_url();

    cos_http_io_deinitialize();

    return 0;
}
```

Server-Side Encryption

Last updated：2024-02-01 18:01:21

Overview

You can encrypt uploaded objects in the following ways.

Using SSE-COS Encryption

Description

With this method, your master key and data are managed by COS. COS can automatically encrypt your data when written into the IDC and automatically decrypt it when accessed. Currently, COS supports AES-256 encryption using a COS master key pair.
In the C SDK, you need to configure the encryption by specifying the `x-cos-server-side-encryption` header.

Sample

```
int main(int argc, char *argv[])
{
    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;
    cos_table_t *headers = NULL;
    cos_string_t file;
    cos_string_t download_file;
    // Initialize
    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }
    cos_pool_create(&p, NULL);
    // Initialize the request options
    options = cos_request_options_create(p);
    options->config = cos_config_create(options->pool);
    cos_str_set(&options->config->endpoint, "cos.ap-guangzhou.myqcloud.com"); // Yo
    cos_str_set(&options->config->access_key_id, "SECRETID");// Your SecretId
    cos_str_set(&options->config->access_key_secret, "SECRETKEY"); // Your SecretKe
    cos_str_set(&options->config->appid, "APPID");// Your APPID
    options->config->is_cname = is_cname;
    options->ctl = cos_http_controller_create(options->pool, 0);
    cos_str_set(&bucket, "<BucketName-APPID>");// Your bucket name in the format: B

    // Set headers for server-side SSE-COS encryption
    headers = cos_table_make(p, 1);
    apr_table_add(headers, "x-cos-server-side-encryption", "AES256");

    cos_str_set(&file, "example1.txt");// Your filename
    cos_str_set(&object, "example_object");// Your object name

    // Upload an object
    s = cos_put_object_from_file(options, &bucket, &object, &file, headers, &resp_h
    if (cos_status_is_ok(s)) {
        printf("put object succeeded\\n");
    } else {
        printf("put object failed\\n");
    }

    // Download an object
    cos_str_set(&download_file, "example2.txt");
    s = cos_get_object_to_file(options, &bucket, &object, NULL, NULL, &download_fil
    if (cos_status_is_ok(s)) {
        printf("get object succeeded\\n");
    } else {
        printf("get object failed\\n");
    }

    cos_pool_destroy(p);

    cos_http_io_deinitialize();
    return 0;
}
```

Using SSE-C Encryption

The encryption key is provided by the customer. When you upload an object, COS will apply AES-256 encryption to your data using the customer-provided encryption key pair. In the C SDK, you need to configure the encryption by specifying the `x-cos-server-side-encryption-customer-*` header.

Description

Note:

This type of encryption requires using HTTPS requests.

customerKey: the key provided by the user; this key should be a 32-byte string consisting of numbers, letters, and symbols. Chinese characters are not supported.

If this encryption method was used when you uploaded the source file, you should also use it when you GET (download) or HEAD (query) this file.

Sample

```
int main(int argc, char *argv[])
{
    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;
    cos_table_t *headers = NULL;
    cos_string_t file;
    cos_string_t download_file;
    // Initialize
    if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
       exit(1);
    }
    cos_pool_create(&p, NULL);
    // Initialize the request options
    options = cos_request_options_create(p);
    options->config = cos_config_create(options->pool);
    cos_str_set(&options->config->endpoint, "https://cos.ap-guangzhou.myqcloud.com"
    cos_str_set(&options->config->access_key_id, "SECRETID");// Your SecretId
    cos_str_set(&options->config->access_key_secret, "SECRETKEY"); // Your SecretKe
    cos_str_set(&options->config->appid, "APPID");// Your APPID
    options->config->is_cname = is_cname;
    options->ctl = cos_http_controller_create(options->pool, 0);
    cos_str_set(&bucket, "<BucketName-APPID>");// Your bucket name in the format: B

    // Set headers for SSE-C encryption
    headers = cos_table_make(p, 3);
    apr_table_add(headers, "x-cos-server-side-encryption-customer-algorithm", "AES2
    apr_table_add(headers, "x-cos-server-side-encryption-customer-key", "MDEyMzQ1Nj
    apr_table_add(headers, "x-cos-server-side-encryption-customer-key-MD5", "U5L61r

    cos_str_set(&file, "example1.txt");// Your file name
    cos_str_set(&object, "example_object");// Your object name
    // Upload an object
    s = cos_put_object_from_file(options, &bucket, &object, &file, headers, &resp_h
    if (cos_status_is_ok(s)) {
        printf("put object succeeded\\n");
    } else {
        printf("put object failed\\n");
    }

    // Download an object, which also requires the headers for server-side encrypti
    cos_str_set(&download_file, "example2.txt");
    s = cos_get_object_to_file(options, &bucket, &object, headers, NULL, &download_
    if (cos_status_is_ok(s)) {
        printf("get object succeeded\\n");
    } else {
        printf("get object failed\\n");
    }

    cos_pool_destroy(p);

    cos_http_io_deinitialize();
    return 0;
}
```

Checking Whether Objects Exist

Last updated：2024-02-01 18:01:21

Overview

This document provides an overview of the API and sample code for quickly checking whether an object exists in a bucket.

API	Operation	Description
HEAD Object	Querying object metadata	Queries the metadata of an object

Checking Whether an Object Exists

Description

This API is used to check whether an object exists in a bucket. It actually calls the `HEAD Object` API to perform the action.

Method prototype

```
cos_status_t *cos_check_object_exist
```