Cloud Object Storage

SDK Documentation

製品ドキュメント

カタログ：

SDK Documentation
Android SDK
かいそく入門

最終更新日：：2019-12-31 16:42:10

ダウンロードとインストール

関連資源

ソースコードとデモ

- COS Android SDKのソースコードについては、参照のことAndroid SDK Githubは.
- デモについては，参照のことCOS Android SDK例.

変更日誌

COS Android SDKの変更ログについては、ご参照くださいCOS Android SDK変更ログ.

環境要求

1. SDKはAndroid 2.2およびそれ以上のバージョンをサポートする。
2. 携帯電話はインターネット(GPRS、3 G、4 GまたはWi-Fi経由)に接続する必要がある。
3. モバイルデバイスが十分な記憶容量を持っていることを確保してください。そうでなければ、いくつかの機能が正常に作動しない
4. 上の事務局IDと事務局鍵。API鍵管理CAMコンソール内のappIdを取得する.顧客センター.

- “事務局”“事務局キーワード”“桶”などの用語の定義については、参照のこと。COS語彙.
- SDKには一般的なバッグが含まれています `com.tencent.cos.xml.*; com.tencent.cos.xml.exception.*; com.tencent.cos.xml.model.*; com.tencent.cos.xml.model.bucket.*; com.tencent.cos.xml.model.object.*; com.tencent.cos.xml.transfer.*; com.tencent.cos.xml.listener.*; and com.tencent.qcloud.core.auth.*` .

SDKのインストール

配置権限

SDKを使用するには、ネットワーク、ストレージなどに特定のアクセス権限を持つ必要があります。AndroidManifest.xmlに以下の権限宣言を追加してください(Android 5.0およびそれ以上のバージョンについては、動的に権限を取得する必要があります)：

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

SDK統合

SDKを2つの方法で統合することができます：自動化集積または人工集成.

自動統合(推奨)

1. 依存項をアプリケーションのルートディレクトリに追加したbuild.gradleファイル：

```
dependencies {
...
// Add this line
compile 'com.tencent.qcloud:cosxml:5.4.30'
}
```

2. アップロード、ダウンロード、コピー機能のみが必要な場合、ステップ1の依存項を以下の内容に変更することで、簡略化版のSDKを使用することができる：

```
dependencies {
...
// Add this line
compile 'com.tencent.qcloud:cosxml-lite:5.4.30'
}
```

3. SDKの品質を追跡し最適化し、より良いユーザー体験を得るために、SDKにMobile Tencent Analytics(MTA)を導入しました。この機能を禁止したい場合は、アプリケーションルートディレクトリのbuild.gradleファイルに以下の依存項を追加してください。

```
dependencies {
...
// Add this line
compile ('com.tencent.qcloud:cosxml:5.4.30'){
exclude group:'com.tencent.qcloud', module: 'mtaUtils' // Disable MTA reporting
}
}
```

SDKを簡略化するコードは以下の通り：

```
dependencies {
...
// Add this line
compile ('com.tencent.qcloud:cosxml-lite:5.4.30'){
exclude group:'com.tencent.qcloud', module: 'mtaUtils' // Disable MTA reporting
}
}
```

手動集積

以下の.jarパケットを項目に導入してlibsフォルダに格納する必要がある：

- COS-Android-sdk.jar
- Qcloud-Foundation.jar
- Bolts-tasks.jar
- okhttp.jar(3.9版とさらに高いバージョン)
- jar(1.13.0およびそれ以上のバージョン)
- mtaUtils.jar
- MID-sdk.jar

- mta-android-sdk.jar
- LogUtils.aar

あなたはすべての.jarパッケージをダウンロードすることができますここです。。最新リリースのパッケージをお勧めします。

かいそく入門

次節では、どのようにCOS Android SDKを使用して、クライアントの初期化、バケット作成、クエリーバケツリスト、アップロードオブジェクト、クエリオブジェクトリスト、ダウンロードオブジェクトおよび削除オブジェクトを実行するかについて説明する。

初期化

COSサービスに関する任意の要求を実行する前に、CosXmlServiceオブジェクトをインスタンス化する必要があります。

1. プロファイルクラスCosXmlServiceConfigを初期化する.
2. ライセンスクラスQCloudCredentialProviderを初期化する.
3. COSサービス系CosXmlServiceを初期化する.

初期化配置クラス

CosXmlServiceConfigCOSサービスのプロファイルクラスであり，以下のコードを用いて初期化できる：

```
String region = "The region where the bucket resides";

// Create a CosXmlServiceConfig object and modify the default configuration parameters as needed
CosXmlServiceConfig serviceConfig = new CosXmlServiceConfig.Builder()
.setRegion(region)
.isHttps(true) // Use HTTPS request. The default is HTTP request.
.setDebuggable(true)
.builder();
```

認可クラスの初期化

モバイルデバイスを永久鍵で認証する場合、鍵漏えいのリスクがある。AndroidおよびiOS用のCOS SDKは、一時鍵使用許可要求をサポートする。サービスを設定するだけで、一時鍵を返し、デバイス上で開始されるCOS要求を許可することができる。この方法を強く提案する。より多くの情報については、参照のこと。モバイルアプリ直接移植の実践.

一時キーによるライセンス(推奨)

- 標準応答機構の授権を使用する

一時鍵サービスを設定し、STSSDKで取得したJSONデータを応答体として使用した場合、以下のコードを使ってCOS SDKにライセンスクラスを作成することができます。

```
/**
* Get the URL address of the authorization service
*/
URL url = null; // URL address of the backend authorization service
try {
url = new URL("your_auth_server_url");
} catch (MalformedURLException e) {
e.printStackTrace();
```

```
}

/**
* Initialize the {@link QCloudCredentialProvider} object to provide a temporary key to the SDK
*/
QCloudCredentialProvider credentialProvider = new SessionCredentialProvider(new HttpRequest.Builder<String>()
.url(url)
.method("GET")
.build());
```

標準応答体を使用する場合、署名の開始時間は携帯電話上のシステム時間となります。そのため、デバイス時間の差が大きい(10分を超える)場合には、署名ミスが発生する可能性があります。この場合、カスタム対応体を用いて以下のような認可を行うことができます。

- カスタム対応体を使ったライセンス

より高い柔軟性を得るために、あなたは継承することができます `BasicLifecycleCredentialProvider` 分類してそれを実現する `fetchNewCredentials()` カスタム構成については、例えば、一時鍵サービスのHTTP応答体をカスタマイズして、デバイスの時差が大きくなることによる署名エラーを回避するために、サーバ時間をデバイスに返信することができる。また、他のプロトコルを用いて端末装置とサービス側との間で通信を行うことも選択することができる。

まず、定義 `MyCredentialProvider` カテゴリー：

```
public class MyCredentialProvider extends BasicLifecycleCredentialProvider {

@Override
protected QCloudLifecycleCredentials fetchNewCredentials() throws QCloudClientException {

// First, get the response containing the signature information from your temporary key server
....

// Then, parse the response to get the key information
String tmpSecretId = "COS_SECRETID"; // the secretId of the temporary key
String tmpSecretKey = "COS_SECRETKEY"; // the secretKey of the temporary key
String sessionToken = "TOKEN"; // the token of the temporary key
long expiredTime = 1556183496L;// End of the validity period of the temporary key

// Return server time as the start time of the signature
long beginTime = 1556182000L; // Start of the validity period of the temporary key

// todo something you want

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

そして、使用する `MyCredentialProvider` ライセンス要求のために定義された例：

```
/**
* Initialize the {@link QCloudCredentialProvider} object to provide a temporary key to the SDK
```

```
*/
QCloudCredentialProvider credentialProvider = new MyCredentialProvider();
```

永久鍵によるライセンス(推奨しない)

一時鍵サービスが設定されていない場合には、永続鍵を用いてライセンスクラスを初期化することができる。**私たちはこの方法を推薦しない。**.セキュリティ環境での一時テストにしか適用されていない.コードは以下のとおりである.

```
String secretId = "COS_SECRETID"; // the secretId of the permanent key
String secretKey ="COS_SECRETKEY"; // the secretKey of the permanent key

/**
* Initialize the {@link QCloudCredentialProvider} object to provide a temporary key to the SDK
*/
QCloudCredentialProvider credentialProvider = new ShortTimeCredentialProvider(secretId,
secretKey, 300);
```

CosXmlServiceサービスクラスを初期化中

CosXmlService様々なCOSサービスを操作するために利用可能なCOSサービスクラスである.プロファイルクラスやライセンスクラスをインスタンス化すると，以下のコードを用いて容易にCOSサービスクラスをインスタンス化することができる.

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

たるをつくる

```
String bucket = "examplebucket-1250000000"; // Format: BucketName-APPID
PutBucketRequest putBucketRequest = new PutBucketRequest(bucket);
// Send the request
cosXmlService.putBucketAsync(putBucketRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
// todo Put Bucket success
PutBucketResult putBucketResult = (PutBucketResult)result;
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceException service
Exception) {
// todo Put Bucket failed because of CosXmlClientException or CosXmlServiceException...
}
});
```

クエリーバケツリスト

```
GetServiceRequest getServiceRequest = new GetServiceRequest();
// Send the request
cosXmlService.getServiceAsync(getServiceRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
// todo Put Bucket Lifecycle success
GetServiceResult getServiceResult = (GetServiceResult)result;
}
```

```
@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceException service
Exception) {
// todo Put Bucket Lifecycle failed because of CosXmlClientException or CosXmlServiceException...
}
});
```

アップロード相手

TransferManager和**COSXMLUploadTask**簡単なアップロードおよび複数のアップロードAPIのための非同期的な要求をカプセル化し、アップロード要求を一時停止、復元、およびキャンセルすることをサポートし、オブジェクトアップロードのために提案する。例示的なコードは以下の通りである。

```
// Initialize TransferConfig
TransferConfig transferConfig = new TransferConfig.Builder().build();
/**
If you have special requirements, you can customize the initialization as follows. For example, for an object >= 2 MB
in size, use multipart upload where the size of each part is 1 MB, and for a source object larger than 5 MB, use mult
ipart copying where the size of each part is 5 MB.
TransferConfig transferConfig = new TransferConfig.Builder()
.setDividsionForCopy(5 * 1024 * 1024) // Minimum object size for multipart copying
.setSliceSizeForCopy(5 * 1024 * 1024) // The size of each part for multipart copying
.setDivisionForUpload(2 * 1024 * 1024) // Minimum object size for multipart upload
.setSliceSizeForUpload(1024 * 1024) // The size of each part for multipart upload
.build();
*/

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

String bucket = "bucket name";
String cosPath = "object key"; // This is the absolute path to the object in COS in the format of cosPath = "text.tx
t";
String srcPath = "absolute path to the local file"; // For example, srcPath=Environment.getExternalStorageDirectory
().getPath() + "/text.txt";
String uploadId = null; // If there is an uploadId for initialized multipart upload, assigning the value of the uploa
dId here for resuming upload; otherwise, assign null.
// Upload the object
COSXMLUploadTask cosxmlUploadTask = transferManager.upload(bucket, cosPath, srcPath, uploadId);

/**
* To upload a byte array, you can call the upload(string, string, byte[]) method of TransferManager:
* byte[] bytes = "this is a test".getBytes(Charset.forName("UTF-8"));
* cosxmlUploadTask = transferManager.upload(bucket, cosPath, bytes);
*/

/**
* To upload a byte stream, you can call the upload(String, String, InputStream) method of TransferManager:
* InputStream inputStream = new ByteArrayInputStream("this is a test".getBytes(Charset.forName("UTF-8")));
* cosxmlUploadTask = transferManager.upload(bucket, cosPath, inputStream);
*/


// Set upload progress callback
```

```
cosxmlUploadTask.setCosXmlProgressListener(new CosXmlProgressListener() {
@Override
public void onProgress(long complete, long target) {
float progress = 1.0f * complete / target * 100;
Log.d("TEST", String.format("progress = %d%%", (int)progress));
}
});
// Set return result callback
cosxmlUploadTask.setCosXmlResultListener(new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
COSXMLUploadTaskResult cOSXMLUploadTaskResult = (COSXMLUploadTaskResult)result;
Log.d("TEST", "Success: " + cOSXMLUploadTaskResult.printResult());
}

@Override
public void onFail(CosXmlRequest request, CosXmlClientException exception, CosXmlServiceException serviceException) {
Log.d("TEST", "Failed: " + (exception == null ? serviceException.getMessage() : exception.toString()));
}
});
// Set task status callback where you can view the task process
cosxmlUploadTask.setTransferStateListener(new TransferStateListener() {
@Override
public void onStateChanged(TransferState state) {
Log.d("TEST", "Task state:" + state.name());
}
});

/**
If you have special requirements, you can do the following:
PutObjectRequest putObjectRequest = new PutObjectRequest(bucket, cosPath, srcPath);
putObjectRequest.setRegion(region); // Set the region where the bucket resides
putObjectRequest.setNeedMD5(true); // Whether to enable MD5 checksum
COSXMLUploadTask cosxmlUploadTask = transferManager.upload(putObjectRequest, uploadId);
*/

// Cancel upload
cosxmlUploadTask.cancel();


// Pause upload
cosxmlUploadTask.pause();

// Resume upload
cosxmlUploadTask.resume();
```

照会先リスト

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

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

// Maximum number of entries returned at a time; 1,000 by default
getBucketRequest.setMaxKeys(100);
```

```
// The delimiter is a symbol. If there is a prefix,
// the identical paths between prefix and delimiter are grouped into one class and defined as common prefixes,
// and then all common prefixes are listed. If there is no prefix, the listing starts from the beginning of the path
getBucketRequest.setDelimiter('/');

// Send the request
cosXmlService.getBucketAsync(getBucketRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
// todo Get Bucket success
GetBucketResult getBucketResult = (GetBucketResult)result;
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceException service
Exception) {
// todo Get Bucket failed because of CosXmlClientException or CosXmlServiceException...
}
});
```

ダウンロード対象

TransferManager和**COSXMLDownloadTask**APIをダウンロードする非同期要求をカプセル化し、ダウンロード要求を一時停止、回復、キャンセルすることをサポートします。これがお勧めのオブジェクトのダウンロード方法です。サンプルコードは以下の通りです。

```
Context applicationContext = "application context"; // getApplicationContext()
String bucket = "bucket name"; // Bucket where the object is stored
String cosPath = "object key"; // This is the absolute path to the object in COS in the format of cosPath = "text.tx
t";
String savedDirPath = "local folder path to the downloaded file";
String savedFileName = "local filename of the downloaded object"; // If this is null, the object name in COS will be
used
// Download the object
COSXMLDownloadTask cosxmlDownloadTask = transferManager.download(applicationContext, bucket, cosPath, savedDirPath, s
avedFileName);
// Set download progress callback
cosxmlDownloadTask.setCosXmlProgressListener(new CosXmlProgressListener() {
@Override
public void onProgress(long complete, long target) {
float progress = 1.0f * complete / target * 100;
Log.d("TEST", String.format("progress = %d%%", (int)progress));
}
});
// Set return result callback
cosxmlDownloadTask.setCosXmlResultListener(new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest request, CosXmlResult result) {
COSXMLDownloadTaskResult cOSXMLDownloadTaskResult = (COSXMLDownloadTaskResult)result;
Log.d("TEST", "Success: " + cOSXMLDownloadTaskResult.printResult());
}

@Override
public void onFail(CosXmlRequest request, CosXmlClientException exception, CosXmlServiceException serviceException) {
Log.d("TEST", "Failed: " + (exception == null ? serviceException.getMessage() : exception.toString()));
}
```

```
});
// Set task status callback where you can view the task process
cosxmlDownloadTask.setTransferStateListener(new TransferStateListener() {
@Override
public void onStateChanged(TransferState state) {
Log.d("TEST", "Task state:" + state.name());
}
});

/**
If you have special requirements, you can do the following:
GetObjectRequest getObjectRequest = new GetObjectRequest(bucket, cosPath, localDir, localFileName);
getObjectRequest.setRegion(region); // Set the region where the bucket resides
COSXMLDownloadTask cosxmlDownloadTask = transferManager.download(context, getObjectRequest);
*/

// Cancel download
cosxmlDownloadTask.cancel();

// Pause download
cosxmlDownloadTask.pause();

// Resume download
cosxmlDownloadTask.resume();
```

対象を削除する

```
String bucket = "examplebucket-1250000000"; // Bucket in the format of BucketName-APPID
String cosPath = "exampleobject"; // Location of the object in the bucket, i.e., the object key

DeleteObjectRequest deleteObjectRequest = new DeleteObjectRequest(bucket, cosPath);
// Send the request
cosXmlService.deleteObjectAsync(deleteObjectRequest, new CosXmlResultListener() {
@Override
public void onSuccess(CosXmlRequest cosXmlRequest, CosXmlResult result) {
// todo Delete Object success...
DeleteObjectResult deleteObjectResult = (DeleteObjectResult)result;
}

@Override
public void onFail(CosXmlRequest cosXmlRequest, CosXmlClientException clientException, CosXmlServiceException service
Exception) {
// todo Delete Object failed because of CosXmlClientException or CosXmlServiceException...
}
});
```

スピーディーな体験

最終更新日：：2019-12-31 16:42:11

背景.

モバイルインターネットの時代、アプリケーションはモバイルインターネットの基盤であり、通常は大量のデータのアップロードとダウンロードを必要とする。そのため、データセキュリティと信頼性は重要である。騰訊雲COSサービス本稿では，COSベースのApp Transferサービスを迅速に構築し，テントレクラウドCOS上でAppデータをアップロード・ダウンロードするために必要なことは，サーバ上に業務を展開し，一時鍵を生成・管理することで，データ蓄積を処理し，アプリケーションの業務ロジックに専念し，仕事量を削減し，開発効率を向上させることであり，COSベースのApp Transferサービスを迅速に構築し，騰訊雲COS上でAppデータをアップロードし，ダウンロードする方法を紹介した。

準備作業

- 一時鍵サービスの設定の詳細については、参照のこと一時鍵生成·利用ガイド.
- Android 4.0(APIレベル15)またはそれ以上のバージョンを持つ携帯電話。

> 一時鍵サービスを利用して許可することをお勧めします。もしあなたがこのサービスを設定できないならば、私達の実演と永久鍵を使うことができます。

アプリのダウンロードとインストール

Android携帯電話を使ってQRコードをスキャンすることで、直接デモをダウンロードできます：

完全コードのダウンロードからGitHub>>.

C SDK
かいそく入門

最終更新日：：2019-12-31 16:42:12

ダウンロードとインストール

関連資源

- COS XML C SDKソースコードをダウンロード：XML C SDK.
- ダウンロードデモ：XML C SDKデモ.

環境要求

必要なライブラリ：libcurl apr apr-util mini xml.

SDKのインストール

1. CMakeツールのダウンロード(2.6.0以上の使用を推奨)ここです。以下のように実装した.

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

2. libcurlのダウンロード(v 7.32.0とより高いバージョンがおすすめ)ここです。以下のように実装した.

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

3. APR(おすすめバージョン1.5.2とより高いバージョン)をダウンロードここです。以下のように実装した.

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

4. apr-util(推奨バージョン1.5.4とより高いバージョン)のダウンロードここです。以下のように実装した。`--with-apr` インストール過程中のオプション：

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

5. mini xml(おすすめバージョン2.8-2.12)をダウンロードここです。以下のように実装した.

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

5. COS C SDKをコンパイル。XML C SDKソースコード以下のコンパイル命令を実行する：

```
cmake .
make
make install
```

かいそく入門

以下はCOS XML C SDKを用いた一般的な過程である.

1. SDKを初期化する。
2. 要求オプションパラメータを設定する。COS語彙.
   - appIdは、テンセント·クラウド·アカウントを登録した後にシステムに割り当てられたアカウントIDの1つ。
   - “access_key_id”と“access_key_password”はアカウントAPIキーです。
   - 「サイト」はCOSアクセスドメイン名である.より多くの情報については，参照のこと.ドメイン·アクセスドメイン例えば、広州地域のエンドポイントは `cos.ap-guangzhou.myqcloud.com` .
3. APIに必要なパラメータを設定する。
4. SDKAPIを呼び出してリクエストを起動し、応答を取得する。

初期化

```
int main(int argc, char *argv[])
{
/* Call the cos_http_io_initialize method at the program entry to perform certain global resource initialization task
s internally covering network, memory, etc. */
if (cos_http_io_initialize(NULL, 0) != COSE_OK) {
exit(1);
}

/* Call the COS SDK API to upload or download a file */
/* ... user logic code which is omitted here */

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

初期化要求オプション

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

/* Create a new memory pool. The second parameter is NULL, indicating that it is not inherited from other memory pool
s */
cos_pool_create(&pool, NULL);
```

```
/* Create and initialize `options`. This parameter contains global configuration information such as endpoint, access
_key_id, acces_key_secret, is_cname, and curl
* The memory of `options` is allocated by the pool, and will be released upon pool release, so there is no need to re
lease the memory separately.
*/
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 the endpoint based on the COS service domain
name of your region
cos_str_set(&options->config->access_key_id, "<user's SecretId>"); // This is the SecretId obtained after you regist
er for the COS service
cos_str_set(&options->config->access_key_secret, "<user's SecretKey>"); // This is the SecretKey obtained after you
register for the COS service
cos_str_set(&options->config->appid, "<user's AppId>"); // This is the AppId obtained after you register for the COS
service

/* You can use a temporary key by setting sts_token. When you use a temporary key, you need to set access_key_id and
access_key_secret to its SecretId and SecretKey */
//cos_str_set(&options->config->sts_token, "MyTokenString");
/* Whether CNAME is used */
options->config->is_cname = 0;

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

/* Used to set whether to add Content-MD5 header automatically to the upload request. If `enable` is COS_FALSE, the h
eader will not be automatically added; if `enable` is COS_TRUE, the header will be automatically added. If this param
eter is not set, the header will be added by default */
cos_set_content_md5_enable(options->ctl, COS_FALSE);

/* Used to set the request routing address and port. Normally, you do not need to set this parameter. The request wil
l be routed according to the result of domain name resolution */
//cos_set_request_route(options->ctl, "192.168.12.34", 80);
```

たるをつくる

```
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 not inherited from other memory pool
s */
cos_pool_create(&p, NULL);

/* Create and initialize `options`. This parameter contains global configuration information such as endpoint, access
_key_id, acces_key_secret, is_cname, and curl
* The memory of `options` is allocated by the pool, and will be released upon pool release, so there is no need to re
lease the memory separately.
*/
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_secret, is_cname, and curl paramet
ers */
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);
```

照会先リスト

```
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;

/* Create a new memory pool. The second parameter is NULL, indicating that it is not inherited from other memory pool
s */
cos_pool_create(&p, NULL);

/* Create and initialize `options`. This parameter contains global configuration information such as endpoint, access
_key_id, acces_key_secret, is_cname, and curl
* The memory of `options` is allocated by the pool, and will be released upon pool release, so there is no need to re
lease the memory separately.
*/
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_secret, is_cname, and curl paramet
ers */
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);
```

アップロード相手

```
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 new memory pool. The second parameter is NULL, indicating that it is not inherited from other memory pool
s */
cos_pool_create(&p, NULL);

/* Create and initialize `options`. This parameter contains global configuration information such as endpoint, access
_key_id, acces_key_secret, is_cname, and curl
* The memory of `options` is allocated by the pool, and will be released upon pool release, so there is no need to re
lease the memory separately.
*/
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_secret, is_cname, and curl paramet
ers */
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);
```

ダウンロード対象

```
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 new memory pool. The second parameter is NULL, indicating that it is not inherited from other memory pool
s */
cos_pool_create(&p, NULL);

/* Create and initialize `options`. This parameter contains global configuration information such as endpoint, access
_key_id, acces_key_secret, is_cname, and curl
* The memory of `options` is allocated by the pool, and will be released upon pool release, so there is no need to re
lease the memory separately.
*/
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_secret, is_cname, and curl paramet
ers */
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_headers);
if (cos_status_is_ok(s)) {
printf("get object succeeded¥n");
} else {
printf("get object failed¥n");
}

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

対象を削除する

```
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 not inherited from other memory pool
s */
cos_pool_create(&p, NULL);

/* Create and initialize `options`. This parameter contains global configuration information such as endpoint, access
_key_id, access_key_secret, is_cname, and curl
* The memory of `options` is allocated by the pool, and will be released upon pool release, so there is no need to re
lease the memory separately.
*/
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_secret, is_cname, and curl paramet
ers */
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
かいそく入門

最終更新日：：2019-12-31 16:42:13

ダウンロードとインストール

関連資源

- COS XML Cをダウンロード++SDKソースコード：XML C++SDK.
- ダウンロードデモ：COS XML C++SDKデモ.

環境要求

- COS XML C++SDKはLinuxをサポートし、WindowsやMacシステムには対応しない。
- 必要な静的ライブラリ：jsoncpp Boost_system Boost_thread Poco(libフォルダ下)。
- 必要な動的ライブラリ：SSL暗号化RTz(実装が必要).
  JsonCppライブラリとヘッダファイルはSDKで利用可能です。自分でインストールしたい場合は、以下の手順でこれらのライブラリをインストールしてコンパイルし、SDKに対応するライブラリとヘッダファイルを差し替えてください。上のライブラリがシステムにインストールされている場合は、SDKの対応するライブラリとヘッダファイルを削除することもできます。

SDKのインストール

(一)CMakeの設置

```
yum install -y gcc gcc-c++ make automake
// Install required packages such as gcc (if they have already been installed, skip this step)
yum install -y wget

// CMake version should be above 3.5
wget https://cmake.org/files/v3.5/cmake-3.5.2.tar.gz
tar -zxvf cmake-3.5.2.tar.gz
cd cmake-3.5.2
./bootstrap --prefix=/usr
gmake
gmake install
```

2.Boostライブラリとヘッダファイルのインストール

```
wget http://sourceforge.net/projects/boost/files/boost/1.54.0/boost_1_54_0.tar.gz
tar -xzvf boost_1_54_0.tar.gz
cd boost_1_54_0
./bootstrap.sh --prefix=/usr/local
./b2 install --with=all
# The Boost libraries are installed in the /usr/local/lib directory
```

3.OpenSSLのインストール

方法1

```
yum install openssl openssl-devel
```

方法2

```
wget https://www.openssl.org/source/openssl-1.0.1t.tar.gz
tar -xzvf ./openssl-1.0.1t.tar.gz
cd openssl-1.0.1t/
./config --prefix=/usr/local/ssl --openssldir=/usr/local/ssl

cd /usr/local/
ln -s ssl openssl
echo "/usr/local/openssl/lib" >> /etc/ld.so.conf
ldconfig

# Add header file/library file search paths (which can be written to ~/.bashrc)
LIBRARY_PATH=/usr/local/ssl/lib/:$LIBRARY_PATH
CPLUS_INCLUDE_PATH=/usr/local/ssl/include/:$CPLUS_INCLUDE_PATH
```

4.Pocoライブラリとヘッダファイルのインストール

フルバージョンのPocoライブラリとヘッダファイルをダウンロードPOCOの公式サイト設置しています

```
./configure --omit=Data/ODBC,Data/MySQL
make
make install
```

部品中の以下の文は，ローカルBoostヘッダファイルパスを指定することができる.`CMakeList.txt` ファイル：

```
SET(BOOST_HEADER_DIR "/root/boost_1_61_0")
```

5.COS CPP SDKのコンパイル

ダウンロードする.XML C++SDKソースコードこれをあなたの開発環境に統合し、以下の命令を実行します：

```
cd ${cos-cpp-sdk}
mkdir -p build
cd build
cmake ..
make
```

はい。デモンストレーション一般的なAPIの例を見つけることができます。 `cos_demo` 直接実行することができます；生成された静的なライブラリは `libcossdk.a` ；生成したlibcossdk.aは `lib` カタログ；および生成の `include` ディレクトリは複製すべきだ `include` プロジェクトの経路。

かいそく入門

次節では，COS Cを用いて基本動作を行う方法について述べる.++例えば、クライアントの初期化、バケット作成、クエリーバケツリスト、アップロードオブジェクト、クエリオブジェクトリスト、ダウンロードオブジェクトおよび削除オブジェクトがある。

> “事務局”“事務局キーワード”“桶”などの用語の定義については、参照のこと。COS語彙.

初期化

プロファイルにおけるフィールドの説明：

```
// For SDK config files before V5.4.3, please use "AccessKey"
"SecretId":"COS_SECRETID",
"SecretKey":"COS_SECRETKEY",

// COS region. For regions and their abbreviations, see https://cloud.tencent.com/document/product/436/6224
"Region":"Region",

// Signature timeout period in seconds
"SignExpiredTime":360,

// Connection timeout period in milliseconds
"ConnectTimeoutInms":6000,

// HTTP request timeout period in milliseconds
"HttpTimeoutInms":60000,

// The size of each part in multipart upload; value range: 1 MB - 5 GB; default value: 1 MB
"UploadPartSize":1048576,

// Thread pool size for uploading a single file in multiple parts
"UploadThreadPoolSize":5,

// The size of each part in file download
"DownloadSliceSize":4194304,

// Thread pool size for downloading a single file
"DownloadThreadPoolSize":5,

// Thread pool size in async upload and download
"AsynThreadPoolSize":2,

// Log output type. 0: no output, 1: output to screen, 2: output to syslog
"LogoutType":1,

// Log level, 1: ERR, 2: WARN, 3: INFO, 4: DBG
"LogLevel":3,
```

たるをつくる

```
#include "cos_api.h"
#include "cos_sys_config.h"
#include "cos_defines.h"

int main(int argc, char *argv[]) {
```

```
// 1. Specify the path to the configuration file and initialize CosConfig
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);

// 2. Construct the request to create a bucket
std::string bucket_name = "examplebucket-1250000000"; // Bucket name
qcloud_cos::PutBucketReq req(bucket_name);
qcloud_cos::PutBucketResp resp;

// 3. Call the API to create a bucket
qcloud_cos::CosResult result = cos.PutBucket(req, &resp);

// 4. Process the result of the call
if (result.IsSucc()) {
// Created successfully
} else {
// Failed to create the bucket. You can call the member function of CosResult to output the error information such as
requestID
std::cout << "ErrorInfo=" << result.GetErrorInfo() << std::endl;
std::cout << "HttpStatus=" << result.GetHttpStatus() << std::endl;
std::cout << "ErrorCode=" << result.GetErrorCode() << std::endl;
std::cout << "ErrorMsg=" << result.GetErrorMsg() << std::endl;
std::cout << "ResourceAddr=" << result.GetResourceAddr() << std::endl;
std::cout << "XCosRequestId=" << result.GetXCosRequestId() << std::endl;
std::cout << "XCosTraceId=" << result.GetXCosTraceId() << std::endl;
}
}
```

クエリーバケツリスト

```
#include "cos_api.h"
#include "cos_sys_config.h"
#include "cos_defines.h"

int main(int argc, char *argv[]) {
// 1. Specify the path to the configuration file and initialize CosConfig
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);

// 2. Construct the request to upload a file
qcloud_cos::GetServiceReq req;
qcloud_cos::GetServiceResp resp;
qcloud_cos::CosResult result = cos.GetService(req, &resp);

// 3. Get the response information
const qcloud_cos::Owner& owner = resp.GetOwner();
const std::vector<qcloud_cos::Bucket>& buckets = resp.GetBuckets();
std::cout << "owner.m_id=" << owner.m_id << ", owner.display_name=" << owner.m_display_name << std::endl;

for (std::vector<qcloud_cos::Bucket>::const_iterator itr = buckets.begin(); itr != buckets.end(); ++itr) {
const qcloud_cos::Bucket& bucket = *itr;
std::cout << "Bucket name=" << bucket.m_name << ", location="
<< bucket.m_location << ", create_date=" << bucket.m_create_date << std::endl;
}

// 4. Process the result of the call
```

```
if (result.IsSucc()) {
// File successfully uploaded
} else {
// Failed to upload the file. You can call the member function of CosResult to output the error information such as r
equestID
std::cout << "ErrorInfo=" << result.GetErrorInfo() << std::endl;
std::cout << "HttpStatus=" << result.GetHttpStatus() << std::endl;
std::cout << "ErrorCode=" << result.GetErrorCode() << std::endl;
std::cout << "ErrorMsg=" << result.GetErrorMsg() << std::endl;
std::cout << "ResourceAddr=" << result.GetResourceAddr() << std::endl;
std::cout << "XCosRequestId=" << result.GetXCosRequestId() << std::endl;
std::cout << "XCosTraceId=" << result.GetXCosTraceId() << std::endl;
}
}
```

アップロード相手

```
#include "cos_api.h"
#include "cos_sys_config.h"
#include "cos_defines.h"

int main(int argc, char *argv[]) {
// 1. Specify the path to the configuration file and initialize CosConfig
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);

// 2. Construct the request to upload a file
std::string bucket_name = "examplebucket-1250000000"; // Destination bucket name
std::string object_name = "exampleobject"; // exampleobject is the object key (`Key`) which is a unique ID for the ob
ject in the bucket. For example, in the object's access domain name examplebucket-1250000000.cos.ap-guangzhou.myqclou
d.com/doc/pic.jpg, the object key is `doc/pic.jpg`
// The local file path is required in the constructor of `request`.
qcloud_cos::PutObjectByFileReq req(bucket_name, object_name, "/path/to/local/file");
req.SetXCosStorageClass("STANDARD_IA"); // Call the `Set` method to set metadata
qcloud_cos::PutObjectByFileResp resp;

// 3. Call the API to upload the file
qcloud_cos::CosResult result = cos.PutObject(req, &resp);

// 4. Process the result of the call
if (result.IsSucc()) {
// File successfully uploaded
} else {
// Failed to upload the file. You can call the member function of CosResult to output the error information such as r
equestID
std::cout << "ErrorInfo=" << result.GetErrorInfo() << std::endl;
std::cout << "HttpStatus=" << result.GetHttpStatus() << std::endl;
std::cout << "ErrorCode=" << result.GetErrorCode() << std::endl;
std::cout << "ErrorMsg=" << result.GetErrorMsg() << std::endl;
std::cout << "ResourceAddr=" << result.GetResourceAddr() << std::endl;
std::cout << "XCosRequestId=" << result.GetXCosRequestId() << std::endl;
std::cout << "XCosTraceId=" << result.GetXCosTraceId() << std::endl;
}
}
```

照会先リスト

```
#include "cos_api.h"
#include "cos_sys_config.h"
#include "cos_defines.h"

int main(int argc, char *argv[]) {
// 1. Specify the path to the configuration file and initialize CosConfig
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);

// 2. Construct the request to create a bucket
std::string bucket_name = "examplebucket-1250000000"; // Destination bucket name
qcloud_cos::GetBucketReq req(bucket_name);
qcloud_cos::GetBucketResp resp;
qcloud_cos::CosResult result = cos.GetBucket(req, &resp);

std::vector<qcloud_cos::Content> cotents = resp.GetContents();
for (std::vector<qcloud_cos::Content>::const_iterator itr = cotents.begin(); itr != cotents.end(); ++itr) {
const qcloud_cos::Content& content = *itr;
std::cout << "key name=" << content.m_key << ", lastmodified ="
<< content.m_last_modified << ", size=" << content.m_size << std::endl;
}

// 3. Process the result of the call
if (result.IsSucc()) {
// File successfully uploaded
} else {
// Failed to upload the file. You can call the member function of CosResult to output the error information such as r
equestID
std::cout << "ErrorInfo=" << result.GetErrorInfo() << std::endl;
std::cout << "HttpStatus=" << result.GetHttpStatus() << std::endl;
std::cout << "ErrorCode=" << result.GetErrorCode() << std::endl;
std::cout << "ErrorMsg=" << result.GetErrorMsg() << std::endl;
std::cout << "ResourceAddr=" << result.GetResourceAddr() << std::endl;
std::cout << "XCosRequestId=" << result.GetXCosRequestId() << std::endl;
std::cout << "XCosTraceId=" << result.GetXCosTraceId() << std::endl;
}
}
```

ダウンロード対象

```
#include "cos_api.h"
#include "cos_sys_config.h"
#include "cos_defines.h"

int main(int argc, char *argv[]) {
// 1. Specify the path to the configuration file and initialize CosConfig
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);

// 2. Construct the request to create a bucket
std::string bucket_name = "examplebucket-1250000000"; // Destination bucket name
std::string object_name = "exampleobject"; // exampleobject is the object key (Key) which is a unique ID for the obje
ct in the bucket. For example, in the object's access domain name examplebucket-1250000000.cos.ap-guangzhou.myqcloud.
```

```
com/doc/pic.jpg, the object key is doc/pic.jpg
std::string local_path = "/tmp/exampleobject";
// Request needs to carry appid, bucketname, object, and local path (including the file name)
qcloud_cos::GetObjectByFileReq req(bucket_name, object_name, local_path);
qcloud_cos::GetObjectByFileResp resp;

// 3. Call the API to create a bucket
qcloud_cos::CosResult result = cos.GetObject(req, &resp);

// 4. Process the result of the call
if (result.IsSucc()) {
// File successfully downloaded
} else {
// Failed to download the file. You can call the member function of CosResult to output the error information such as
requestID
std::cout << "ErrorInfo=" << result.GetErrorInfo() << std::endl;
std::cout << "HttpStatus=" << result.GetHttpStatus() << std::endl;
std::cout << "ErrorCode=" << result.GetErrorCode() << std::endl;
std::cout << "ErrorMsg=" << result.GetErrorMsg() << std::endl;
std::cout << "ResourceAddr=" << result.GetResourceAddr() << std::endl;
std::cout << "XCosRequestId=" << result.GetXCosRequestId() << std::endl;
std::cout << "XCosTraceId=" << result.GetXCosTraceId() << std::endl;
}
}
```

対象を削除する

```
#include "cos_api.h"
#include "cos_sys_config.h"
#include "cos_defines.h"

int main(int argc, char *argv[]) {
// 1. Specify the path to the configuration file and initialize CosConfig
qcloud_cos::CosConfig config("./config.json");
qcloud_cos::CosAPI cos(config);

// 2. Construct the request to create a bucket
std::string bucket_name = "examplebucket-1250000000"; // Destination bucket name
std::string object_name = "exampleobject"; // exampleobject is the object key (Key) which is a unique ID for the obje
ct in the bucket. For example, in the object's access domain name examplebucket-1250000000.cos.ap-guangzhou.myqcloud.
com/doc/pic.jpg, the object key is doc/pic.jpg
// 3. Call the API to create a bucket
qcloud_cos::DeleteObjectReq req(bucket_name, object_name);
qcloud_cos::DeleteObjectResp resp;
qcloud_cos::CosResult result = cos.DeleteObject(req, &resp);

// 4. Process the result of the call
if (result.IsSucc()) {
// File successfully downloaded
} else {
// Failed to download the file. You can call the member function of CosResult to output the error information such as
requestID
std::cout << "ErrorInfo=" << result.GetErrorInfo() << std::endl;
std::cout << "HttpStatus=" << result.GetHttpStatus() << std::endl;
std::cout << "ErrorCode=" << result.GetErrorCode() << std::endl;
std::cout << "ErrorMsg=" << result.GetErrorMsg() << std::endl;
```

```
std::cout << "ResourceAddr=" << result.GetResourceAddr() << std::endl;
std::cout << "XCosRequestId=" << result.GetXCosRequestId() << std::endl;
std::cout << "XCosTraceId=" << result.GetXCosTraceId() << std::endl;
}
}
```

.Net SDK
かいそく入門

最終更新日：：2019-12-31 16:42:15

ダウンロードとインストール

関連資源

- .NET用のCOS XML SDKソースコードをダウンロードできるここです。.
- デモはダウンロードできますここです。.

環境依存性

- CoS XML SDK for.NET：.NET標準2.0に基づいて開発。
- Windows：インストール.NETCore 2.0または.NET Framework 4.6.1以上です。
- Linux/Mac：.NET Core 2.0またはそれ以上のバージョンをインストールする。

SDK追加

Nugetを統合する方法を提供し、それを追加することができる `csproj` プロジェクトファイル：

```
<PackageReference Include="Tencent.QCloud.Cos.Sdk" Version="5.4.5" />
```

.NET CLIを使うなら、以下のコマンドを実行してインストールしてください：

```
dotnet add package Tencent.QCloud.Cos.Sdk
```

sdkを手動でダウンロードすることもできます。ここです。.

その他の従属関係

第三者依存項としてNewtonsoft.Jsonを使用していますが、自動的にローカルで抽出されていなければ追加することができます `csproj` 手動ファイル：

```
<PackageReference Include="Newtonsoft.Json" Version="12.0.2" />
```

かいそく入門

次節では、クライアントの初期化、バケット作成、クエリーバケツリスト、アップロードオブジェクト、クエリオブジェクトリスト、ダウンロードオブジェクトおよび削除オブジェクトなど、COS SDK for.NETにおいて基本動作をどのように実行するかについて説明する。

- 事務局ID、事務局キーワード、樽その他の用語の定義については、参照のことCOS語彙.
- SDKでよく使われる命名空間は `using COSXML, using COSXML.Auth, using COSXML.Model.Object, using COSXML.Model.Bucket, and using COSXML.CosException` .

初期化

いずれかのCOS要求を実行する前に，3つのオブジェクトをインスタンス化する： `CosXmlConfig, QCloudCredentialProvider, and CosXmlServer` .

- `CosXmlConfig` SDKを構成するためのAPIを提供する。
- `QCloudCredentialProvider` キー情報を設定するためのAPIを提供する。
- `CosXmlServer` 各種COS APIを提供する。

```
// Initialize CosXmlConfig
string appid = "1250000000";// Set the APPID of your Tencent Cloud account
string region = "ap-beijing"; // Set the default bucket region
CosXmlConfig config = new CosXmlConfig.Builder()
.SetConnectionTimeoutMs(60000) // Set the connection timeout period in milliseconds, which is 45,000 ms by default
.SetReadWriteTimeoutMs(40000) // Set the read/write timeout period in milliseconds, which is 45,000 ms by default
.IsHttps(true) // Set HTTPS as default request method
.SetAppid(appid) // Set the APPID of your Tencent Cloud account
.SetRegion(region) // Set the default bucket region
.SetDebugLog(true) // Show the log
.Build(); // Create a CosXmlConfig object

// Initialize QCloudCredentialProvider. The SDK provides three methods: permanent key, temporary key, and custom
QCloudCredentialProvider cosCredentialProvider = null;

// Method 1: Permanent key
string secretId = "COS_SECRETID"; //"TencentCloud API key's SecretId";
string secretKey = "COS_SECRETKEY"; //"TencentCloud API key's SecretKey";
long durationSecond = 600; // Validity period of the secretKey in seconds
cosCredentialProvider = new DefaultQCloudCredentialProvider(secretId, secretKey, durationSecond);

// Method 2: Temporary key
string tmpSecretId = "COS_SECRETID"; // "Temporary key's SecretId";
string tmpSecretKey = "COS_SECRETKEY"; // "Temporary key's SecretKey";
string tmpToken = "COS_TOKEN"; // "Temporary key's token";
long tmpExpireTime = 1546862502;// End time of validity of the temporary key
cosCredentialProvider = new DefaultSessionQCloudCredentialProvider(tmpSecretId,tmpSecretKey,tmpExpireTime,tmpToken);

// Method 3: Custom key information, which inherits QCloudCredentialProvider and rewrites the GetQCloudCredentials()
method
public class MyQCloudCredentialProvider : QCloudCredentialProvider
{
public override QCloudCredentials GetQCloudCredentials()
{
string secretId = "COS_SECRETID"; // Key's SecretId
string secretKey = "COS_SECRETKEY"; // Key's SecretKey
string keyTime = "Validity period of the key"; //1546862502;1546863102
return new QCloudCredentials(secretId, secretKey, keyTime);
}

public override void Refresh()
{
// Update key information
}
}
cosCredentialProvider = new MyQCloudCredentialProvider();
```

```
// Initialize CosXmlServer
CosXmlServer cosXml = new CosXmlServer(config, cosCredentialProvider);
```

たるをつくる

```
try
{
string bucket = "examplebucket-1250000000"; // Bucket name in the format of BucketName-APPID
PutBucketRequest request = new PutBucketRequest(bucket);
// Set the validity period of the signature
request.SetSign(TimeUtils.GetCurrentTime(TimeUnit.SECONDS), 600);
// Execute the request
PutBucketResult result = cosXml.PutBucket(request);
// Request succeeded
Console.WriteLine(result.GetResultInfo());
}
catch (COSXML.CosException.CosClientException clientEx)
{
// Request failed
Console.WriteLine("CosClientException: " + clientEx.Message);
}
catch (COSXML.CosException.CosServerException serverEx)
{
// Request failed
Console.WriteLine("CosServerException: " + serverEx.GetInfo());
}
```

クエリーバケツリスト

```
try
{
GetServiceRequest request = new GetServiceRequest();
// Set the validity period of the signature
request.SetSign(TimeUtils.GetCurrentTime(TimeUnit.SECONDS), 600);
// Execute the request
GetServiceResult result = cosXml.GetService(request);
// Request succeeded
Console.WriteLine(result.GetResultInfo());
}
catch (COSXML.CosException.CosClientException clientEx)
{
// Request failed
Console.WriteLine("CosClientException: " + clientEx.Message);
}
catch (COSXML.CosException.CosServerException serverEx)
{
// Request failed
Console.WriteLine("CosServerException: " + serverEx.GetInfo());
}
```

アップロード相手

```
try
{
```

```
string bucket = "examplebucket-1250000000"; // Bucket in the format of BucketName-APPID
string key = "exampleobject"; // Location of the object in the bucket, i.e., the object key
string srcPath = @"F:¥exampleobject";// Absolute path to the local file
PutObjectRequest request = new PutObjectRequest(bucket, key, srcPath);
// Set the validity period of the signature
request.SetSign(TimeUtils.GetCurrentTime(TimeUnit.SECONDS), 600);
// Set progress callback
request.SetCosProgressCallback(delegate(long completed, long total)
{
Console.WriteLine(String.Format("progress = {0:##.##}%", completed * 100.0 / total));
});
// Execute the request
PutObjectResult result = cosXml.PutObject(request);
// Request succeeded
Console.WriteLine(result.GetResultInfo());
}
catch (COSXML.CosException.CosClientException clientEx)
{
// Request failed
Console.WriteLine("CosClientException: " + clientEx.Message);
}
catch (COSXML.CosException.CosServerException serverEx)
{
// Request failed
Console.WriteLine("CosServerException: " + serverEx.GetInfo());
}

**// Multipart upload() needs to be used for large files. For more information, see the TransferManager and COSXMLUpl
oadTask classes encapsulated in the SDK. Below is an example:**
TransferManager transferManager = new TransferManager(cosXml, new TransferConfig());
COSXMLUploadTask uploadTask = new COSXMLUploadTask(bucket, null, key);
uploadTask.SetSrcPath(srcPath);
uploadTask.progressCallback = delegate (long completed, long total)
{
Console.WriteLine(String.Format("progress = {0:##.##}%", completed * 100.0 / total));
};
uploadTask.successCallback = delegate (CosResult cosResult)
{
COSXML.Transfer.COSXMLUploadTask.UploadTaskResult result = cosResult as COSXML.Transfer.COSXMLUploadTask.UploadTaskRe
sult;
Console.WriteLine(result.GetResultInfo());
};
uploadTask.failCallback = delegate (CosClientException clientEx, CosServerException serverEx)
{
if (clientEx != null)
{
Console.WriteLine("CosClientException: " + clientEx.Message);
}
if (serverEx != null)
{
Console.WriteLine("CosServerException: " + serverEx.GetInfo());
}
};
transferManager.Upload(uploadTask);
```

照会先リスト

```
try
{
string bucket = "examplebucket-1250000000"; // Format: BucketName-APPID
GetBucketRequest request = new GetBucketRequest(bucket);
// Set the validity period of the signature
request.SetSign(TimeUtils.GetCurrentTime(TimeUnit.SECONDS), 600);
// Execute the request
GetBucketResult result = cosXml.GetBucket(request);
// Request succeeded
Console.WriteLine(result.GetResultInfo());
}
catch (COSXML.CosException.CosClientException clientEx)
{
// Request failed
Console.WriteLine("CosClientException: " + clientEx.Message);
}
catch (COSXML.CosException.CosServerException serverEx)
{
// Request failed
Console.WriteLine("CosServerException: " + serverEx.GetInfo());
}
```

ダウンロード対象

```
try
{
string bucket = "examplebucket-1250000000"; // Bucket in the format of BucketName-APPID
string key = "exampleobject"; // Location of the object in the bucket, i.e., the object key
string localDir = @"F:¥";// Download to the specified local folder
string localFileName = "exampleobject"; // Specify the name of the file to be saved in the local file system
GetObjectRequest request = new GetObjectRequest(bucket, key, localDir, localFileName);
// Set the validity period of the signature
request.SetSign(TimeUtils.GetCurrentTime(TimeUnit.SECONDS), 600);
// Set progress callback
request.SetCosProgressCallback(delegate(long completed, long total)
{
Console.WriteLine(String.Format("progress = {0:##.##}%", completed * 100.0 / total));
});
// Execute the request
GetObjectResult result = cosXml.GetObject(request);
// Request succeeded
Console.WriteLine(result.GetResultInfo());
}
catch (COSXML.CosException.CosClientException clientEx)
{
// Request failed
Console.WriteLine("CosClientException: " + clientEx.Message);
}
catch (COSXML.CosException.CosServerException serverEx)
{
// Request failed
Console.WriteLine("CosServerException: " + serverEx.GetInfo());
}
```

対象を削除する

```
try
{
string bucket = "examplebucket-1250000000"; // Bucket in the format of BucketName-APPID
string key = "exampleobject"; // Location of the object in the bucket, i.e., the object key
DeleteObjectRequest request = new DeleteObjectRequest(bucket, key);
// Set the validity period of the signature
request.SetSign(TimeUtils.GetCurrentTime(TimeUnit.SECONDS), 600);
// Execute the request
DeleteObjectResult result = cosXml.DeleteObject(request);
// Request succeeded
Console.WriteLine(result.GetResultInfo());
}
catch (COSXML.CosException.CosClientException clientEx)
{
// Request failed
Console.WriteLine("CosClientException: " + clientEx.Message);
}
catch (COSXML.CosException.CosServerException serverEx)
{
// Request failed
Console.WriteLine("CosServerException: " + serverEx.GetInfo());
}
```

Go SDK
かいそく入門

最終更新日：：2019-12-31 16:42:16

ダウンロードとインストール

関連資源

- GoソースコードのCOS XML SDKがダウンロードできるここです。.
- デモはダウンロードできますここです。.
- より多くの情報については，参照のことGo用COS SDK.

環境依存性

- GO：GO環境をダウンロード·インストールするために、GOの公式サイトからダウンロードできます

SDKのインストール

以下のコマンドを実行することで、GO用のCOS SDKをインストールできる：

```
go get -u github.com/tencentyun/cos-go-sdk-v5/
```

かいそく入門

次節では、クライアントの初期化、バケット作成、クエリーバケツリスト、アップロードオブジェクト、クエリオブジェクトリスト、ダウンロードオブジェクトおよび削除オブジェクトなど、COS Go SDKを用いて基本動作を実行する方法について説明する。

初期化

COSドメインを使ってGoのためのCOSクライアント構造を生成することができます。

方法原型

```
func NewClient(uri *BaseURL, httpClient *http.Client) *Client
```

サンプル請求

```
// Replace <BucketName-APPID> and <region> with your own information
// For example: http://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com
u, _ := url.Parse("http://<BucketName-APPID>.cos.<region>.myqcloud.com")
b := &cos.BaseURL{BucketURL: u}
// 1. Permanent key
client := cos.NewClient(b, &http.Client{
Transport: &cos.AuthorizationTransport{
SecretID: "COS_SECRETID",
SecretKey: "COS_SECRETKEY",
},
})
```

```
// 2. Temporary key
client := cos.NewClient(b, &http.Client{
Transport: &cos.AuthorizationTransport{
SecretID: "COS_SECRETID",
SecretKey: "COS_SECRETKEY",
SessionToken : "COS_SECRETTOKEN",
},
})
```

たるをつくる

```
package main
import (
"context"
"io/ioutil"
"net/http"
"net/url"
"os"
"time"

"github.com/tencentyun/cos-go-sdk-v5"
)

func main() {
// Replace <BucketName-APPID> and <region> with your own information
// For example: http://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com
u, _ := url.Parse("http://<BucketName-APPID>.cos.<region>.myqcloud.com")
b := &cos.BaseURL{BucketURL: u}
c := cos.NewClient(b, &http.Client{
Transport: &cos.AuthorizationTransport{
SecretID: "COS_SECRETID",
SecretKey: "COS_SECRETKEY",
},
})

_, err := c.Bucket.Put(context.Background(), nil)
if err != nil {
panic(err)
}
}
```

クエリーバケツリスト

```
package main

import (
"context"
"fmt"
"os"
"net/http"

"github.com/tencentyun/cos-go-sdk-v5"
)

func main() {
```

```
c := cos.NewClient(nil, &http.Client{
Transport: &cos.AuthorizationTransport{
SecretID: "COS_SECRETID",
SecretKey: "COS_SECRETKEY",
},
})

s, _, err := c.Service.Get(context.Background())
if err != nil {
panic(err)
}

for _, b := range s.Buckets {
fmt.Printf("%#v¥n", b)
}
}
```

アップロード相手

```
package main
import (
"context"
"net/url"
"os"
"strings"
"net/http"

"github.com/tencentyun/cos-go-sdk-v5"
)

func main() {
// Replace <BucketName-APPID> and <region> with your own information
// For example: http://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com
u, _ := url.Parse("http://<BucketName-APPID>.cos.<region>.myqcloud.com")
b := &cos.BaseURL{BucketURL: u}
c := cos.NewClient(b, &http.Client{
Transport: &cos.AuthorizationTransport{
SecretID: "COS_SECRETID",
SecretKey: "COS_SECRETKEY",
},
})
// An object key is the unique identifier of an object in a bucket.
// For example, in the access domain name `examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/test/objectPut.go`,
the object key is test/objectPut.go
name := "test/objectPut.go"
// 1.Normal put string content
f := strings.NewReader("test")

_, err := c.Object.Put(context.Background(), name, f, nil)
if err != nil {
panic(err)
}
// 2.Put object by local file path
_, err = c.Object.PutFromFile(context.Background(), name, "./test", nil)
if err != nil {
panic(err)
```

```
}
}
```

照会先リスト

```
package main

import (
"context"
"fmt"
"os"
"net/url"
"net/http"

"github.com/tencentyun/cos-go-sdk-v5"
)

func main() {
// Replace <BucketName-APPID> and <region> with your own information
// For example: http://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com
u, _ := url.Parse("http://<BucketName-APPID>.cos.<region>.myqcloud.com")
b := &cos.BaseURL{BucketURL: u}
c := cos.NewClient(b, &http.Client{
Transport: &cos.AuthorizationTransport{
SecretID: "COS_SECRETID",
SecretKey: "COS_SECRETKEY",
},
})

opt := &cos.BucketGetOptions{
Prefix: "test",
MaxKeys: 3,
}
v, _, err := c.Bucket.Get(context.Background(), opt)
if err != nil {
panic(err)
}

for _, c := range v.Contents {
fmt.Printf("%s, %d¥n", c.Key, c.Size)
}
}
```

ダウンロード対象

```
package main

import (
"context"
"fmt"
"net/url"
"os"
"io/ioutil"
"net/http"
```

```
"github.com/tencentyun/cos-go-sdk-v5"
)

func main() {
// Replace <BucketName-APPID> and <region> with your own information
// For example: http://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com
u, _ := url.Parse("http://<BucketName-APPID>.cos.<region>.myqcloud.com")
b := &cos.BaseURL{BucketURL: u}
c := cos.NewClient(b, &http.Client{
Transport: &cos.AuthorizationTransport{
SecretID: "COS_SECRETID",
SecretKey: "COS_SECRETKEY",
},
})
// 1.Get object content by resp body.
name := "test/hello.txt"
resp, err := c.Object.Get(context.Background(), name, nil)
if err != nil {
panic(err)
}
bs, _ := ioutil.ReadAll(resp.Body)
resp.Body.Close()
fmt.Printf("%s¥n", string(bs))
// 2.Get object to local file path.
_, err = c.Object.GetToFile(context.Background(), name, "example", nil)
if err != nil {
panic(err)
}
}
```

対象を削除する

```
package main

import (
"context"
"fmt"
"net/url"
"os"
"io/ioutil"
"net/http"

"github.com/tencentyun/cos-go-sdk-v5"
)

func main() {
// Replace <BucketName-APPID> and <region> with your own information
// For example: http://examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com
u, _ := url.Parse("http://<BucketName-APPID>.cos.<region>.myqcloud.com")
b := &cos.BaseURL{BucketURL: u}
c := cos.NewClient(b, &http.Client{
Transport: &cos.AuthorizationTransport{
SecretID: "COS_SECRETID",
SecretKey: "COS_SECRETKEY",
},
})
```

```
name := "test_object"
_, err := c.Object.Delete(context.Background(), name)
if err != nil {
panic(err)
}
}
```

iOS SDK
かいそく入門

最終更新日：：2019-12-31 16:42:17

発展準備作業.

SDK取得

- 以下のサイトからCOSサービスのIOS SDKリソースをダウンロード：XML IOS SDK.
- 以下の位置から必要なバージョンを選択することで、Framework形式のPackatedSDKをダウンロードすることができます。釈放する..
- より多くの例については、デモを参照のこと。XML IOS SDKデモ.
- COSごとのXMLバージョンの変更については、参照のことXML IOS SDK変更ログ。

発展の準備

- SDKはIOS 8.0またはそれ以上のバージョンをサポートする。
- あなたの携帯電話はネットワーク(GPRS、3 G、Wi-Fiなど)に接続しなければなりません。
- 従COS V 5コンソール.

> 事務局ID、事務局キーワード、樽その他の用語の定義及びこれらの用語の入手方法については、参照のこと。COS語彙.

構成SDK

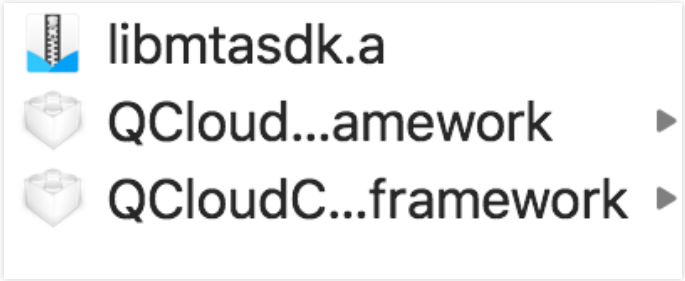
SDKの導入

CocoaPodsやパッケージをダウンロードしたダイナミックなライブラリでSDKを統合することができます。私たちはCocoaPodsを使って導入することをお勧めします。

- **CocoaPodsを使って導入(おすすめ)**
  Podfileでは、使用：

```
pod 'QCloudCOSXML'
```

- **パッケージ動質ライブラリを使って導入(手動インテグレーション)**
  曳航**QCloudCOSXML.framework、QCloudCore.frameworkとlibmtasdk.a**あなたのプロジェクトは、以下のとおりです：



以下の相関ライブラリを追加する：

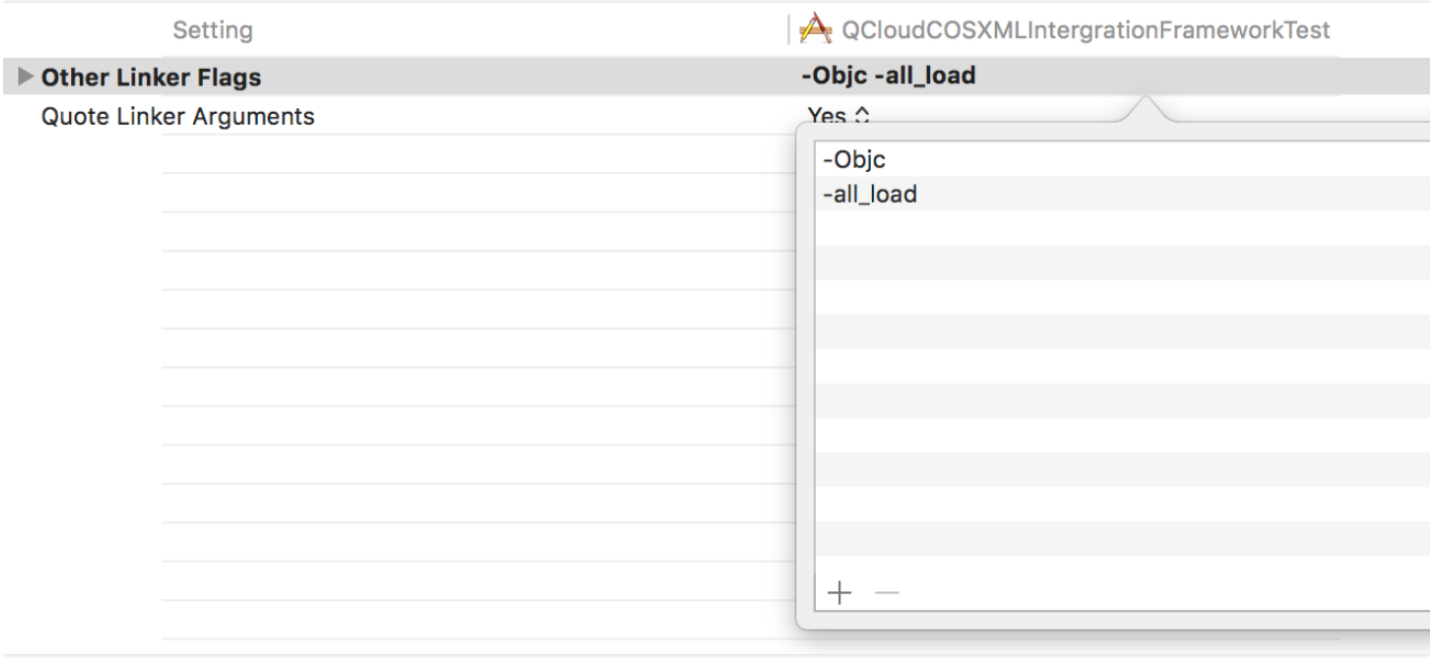
1. コア電話
2. 基礎
3. システム構成
4. 庫++.tbd

配置項目

“Build Settings”に“Other Linker Flag”を設定し、パラメータを追加する：

```
-ObjC
-all_load
```

次の図を参照。



- 騰訊雲COS XML iOS SDKはHTTPプロトコルを使い、iOS上でSDKを動作させ、HTTP転送を可能にする。
  次の2つの方法でHTTP転送を開始することができます：
  - **手動配置**
    プロジェクトの“info.plist”ファイルに“App Transport Security Settings”を追加し、“App Transport Security Settings”の下に“任意ロードを許可”を追加し、そのタイプを“Boolean”、値を“yes”に設定する。
  - **コード構成で**
    App Integration SDKのinfo.plistに以下のコードを追加できます：

```
NSAppTransportSecurity

NSExceptionDomains
```

```
myqcloud.com

NSIncludesサブドメイン

NSTemporaryExceptionAllowsInsecureHTTPLoads
```

```
- Mobile Tencent Analytics (MTA) is also introduced in the SDK. If you want to disable this feature, follow the steps below:
- Import the header file ` #import <QCloudCore/MTAConfig.h>`
- After registering the default cosxml service, add the following code:
`[TACMTAConfig getInstance].statEnable = NO;`
```

初期化

SDKの特性を使用する前に、必要なヘッダファイルをいくつか導入して初期化を実行する。

SDKのヘッダファイルを導入·アップロード：

```
QCloudCore.h,
QCloudCOSXML/QCloudCOSXML.h
```

1. SDKを利用する前に、デフォルトのクラウドサービス配置オブジェクトQCloudServiceConfigurationをインスタンス化し、QCloudCOSXMLServiceとQClodCOSTransferManagerServiceオブジェクトをインスタンス化する。
2. QCloudServiceConfigurationを変更すれば、新しいQCloudCOSTransferManagerServiceを登録できる `registerCOSTransferMangerWithConfiguration:(QCloudServiceConfiguration*)configuration withKey:(NSString*)key` ただし，1つのQCloudCOSTransferManagerServiceをデフォルト値に設定するしかない.

方法原型

インスタンス化QCloudServiceConfigurationオブジェクト：

```
QCloudServiceConfiguration* configuration = [QCloudServiceConfiguration new];
configuration.appID = @""//Project ID.
```

インスタンス化QCloudCOSXMLServiceオブジェクト：

```
+ (QCloudCOSXMLService*) registerDefaultCOSXMLWithConfiguration:(QCloudServiceConfiguration*)configuration;
```

インスタンス化QCloudCOSTransferManagerServiceオブジェクト：

```
+ (QCloudCOSTransferMangerService*) registerDefaultCOSTransferMangerWithConfiguration:(QCloudServiceConfiguration*)configuration;
```

QCloudServiceConfigurationパラメータ

パラメータ名	説明する.	タイプ	ひっくるめ書き
応用プログラム	項目ID，すなわちappIdである.	NSString*	いえ。
エンドポイント	エンドポイントに関する情報を配置する	QCloudCOSXMLEndPoint*	はい。

QCloudCOSXMLEndPointパラメータ

パラメータ名	説明する.	タイプ	ひっくるめ書き
地域名	サービスエリア	NSString*	はい。
サービス名	ドメイン。デフォルト値はmyqcloud.comです。	NSString*	いえ。
HTTPSを使用	HTTPSサービスを使用するかどうかを指示する.デフォルト値はNoである.	ブル	いえ。
接尾辞	カスタム対応http：//ketketname.接尾辞	NSString	いえ。

初期化例

以下の例では、appId、uneId、およびuneKeyは、COS v 5コンソール.

```
//AppDelegate.m

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
QCloudServiceConfiguration* configuration = [QCloudServiceConfiguration new];
configuration.appID = @"*****";
configuration.signatureProvider = self;
QCloudCOSXMLEndPoint* endpoint = [[QCloudCOSXMLEndPoint alloc] init];
endpoint.regionName = @"ap-beijing";//Service region name. See the notes for available regions.
configuration.endpoint = endpoint;
[QCloudCOSXMLService registerDefaultCOSXMLWithConfiguration:configuration];
[QCloudCOSTransferMangerService registerDefaultCOSTransferMangerWithConfiguration:configuration];
}
```

かいそく入門

以下の例では，アップロードとダウンロードの基本プロセスを示す.XML IOS SDKデモ各APIの使用方法については、デモで提供されたセルテストファイルを参照されたい。

> このステップを実行する前に、あなたは騰訊雲制御台でCOSサービスのappIdを申請しなければなりません。

初期化

> QCloudSignatureProviderプロトコルは，QCloudServiceConfigurationのExpressureProviderオブジェクトのために実現する必要がある.

ステップ1：

```
//AppDelegate.m
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
QCloudServiceConfiguration* configuration = [QCloudServiceConfiguration new];
configuration.appID = @"*****";
configuration.signatureProvider = self;
QCloudCOSXMLEndPoint* endpoint = [[QCloudCOSXMLEndPoint alloc] init];
endpoint.regionName = @"ap-beijing";//Service region name. See the notes for available regions.
configuration.endpoint = endpoint;
[QCloudCOSXMLService registerDefaultCOSXMLWithConfiguration:configuration];
[QCloudCOSTransferMangerService registerDefaultCOSTransferMangerWithConfiguration:configuration];
}
```

ステップ2：

```
//AppDelegate.m
- (void) signatureWithFields:(QCloudSignatureFields*)fileds
request:(QCloudBizHTTPRequest*)request
urlRequest:(NSURLRequest*)urlRequst
compelete:(QCloudHTTPAuthentationContinueBlock)continueBlock
{
//The process of implementing signature. We recommend implementing this process on the server. For more information, see the "Generating a Signature" section below.
}
```

書類をアップロードする

この例では、あなたがあなたの企業のためにBucketを申請したと仮定します。実際には、すべてのSDK要求には、それらの要求クラスがあります。要求を生成して関連属性を設定すると、要求をQCloudCOSTransferMangerServiceオブジェクトに渡して、必要な操作を完了させます。要求の本体部分には、アップロードするファイルのローカルURL(NSURL)を入力します。*タイプ)。

ファイルをアップロードするためのAPIは，署名を用いて認証を行う.送信された要求は，初期化期間中に指定されたオブジェクトから自動的に署名を要求し，そのオブジェクトはQCloudSignatureProviderプロトコルに従う.署名をどのように生成するかに関するより多くの情報は，以下の署名生成部を参照されたい.

> アップロード中にURL指向のファイルを変更できないとエラーになります。

実例

```
QCloudCOSXMLUploadObjectRequest* put = [QCloudCOSXMLUploadObjectRequest new];
NSURL* url = [NSURL fileURLWithPath:@"filePathString"] /*URL of file*/;
put.object = @"filename.jpg";
put.bucket = @"test-123456789";
put.body = url;
[put setSendProcessBlock:^(int64_t bytesSent, int64_t totalBytesSent, int64_t totalBytesExpectedToSend) {
NSLog(@"upload %lld totalSend %lld aim %lld", bytesSent, totalBytesSent, totalBytesExpectedToSend);
}];
[put setFinishBlock:^(id outputObject, NSError* error) {

}];
[[QCloudCOSTransferMangerService defaultCOSTransferManager] UploadObject:put];
```

QCloudCOSXMLUploadObjectRequestパラメータ

パラメータ名	説明する.	タイプ	ひっくるめ書き
相手.	ObjectKeyは，リポジトリ中のオブジェクトの一意識別子である.たとえば，オブジェクトのアクセスドメイン「bucket 1—12500000.cos.ap—guangzhou.myqcloud.com/doc 1/PIc 1.jpg」では，ObjectKeyは「doc 1/PIc 1.jpg」である.オブジェクト記述.	NSString*	はい。
水おけ	貯蔵桶の名称は、COS V 5コンソール形式は<桶の名称>-<応用プログラム>例えば、testBucket-1253653367です。	NSString*	はい。
主体.	アップロードするファイルのパスです。NSURL*タイプの変数を入力してください。	BodyType	はい。
StorageClass	対象のストレージ類	QCloudCOSStorageClass	はい。
キャッシュコントロール	RFC 2616で定義されているキャッシュポリシー	NSString*	いえ。
内容配置	RFC 2616で定義されているファイル名	NSString*	いえ。
期待する.	Expect=@“100-Continue”を使用すると、サーバからの応答が届くまで、要求内容は送信されない。	NSString*	いえ。
期限が切れる	RFC 2616で定義されている満期時間	NSString*	いえ。
MultipleUploadFinishBlockを初期化	要求が複数のアップロード要求を生成する場合は、複数のアップロードが初期化された後、当該ブロックを介してコールバックが実行され、マルチロードが完了すると、このコールバックブロックからバケット、キー、アップロードID、および後続の失敗アップロードを回復するためのResumeDataを取得することができる。	ブロック	いえ。
accessControlList	定義対象のACL属性.有効値：私有，公共読み書き，公共読み出し；デフォルト値：プライベート	NSString*	いえ。
grantRead	ライセンスユーザに読み込み権限を付与する.フォーマット：id=「，id=」.子アカウントの許可に対して，id=「qcs」カムuin/<OwnerUin>：uin/<SubUin>“；ルートアカウントのライセンスについては、id=”qcs“カムuin/<OwnerUin>：uin/<OwnerUin>“では、OwnerUinはルートアカウントのIDを指し、SubUinは固定子アカウントのIDを指す。	NSString*	いえ。
GRANT Write	授権ユーザーには書き込み権限を付与する。フォーマットは上と同じである。	NSString*	いえ。
grantFullControl	授権ユーザーに読み書き権限を付与する。フォーマットは上と同じである。	NSString*	いえ。

ファイルをダウンロードする

実例

```
QCloudGetObjectRequest* request = [QCloudGetObjectRequest new];
//Set the URL for download. If it has been set, the file is downloaded to the specified path.
request.downloadingURL = [NSURL URLWithString:QCloudTempFilePathWithExtension(@"downding")];
request.object = @ “Your Object-Key” ;
request.bucket = @"test-123456789";
[request setFinishBlock:^(id outputObject, NSError ¥*error) {
//additional actions after finishing
}];
[request setDownProcessBlock:^(int64_t bytesDownload, int64_t totalBytesDownload, int64_t totalBytesExpectedToDownloa
d) {
//Progress of download
}];
[[QCloudCOSXMLService defaultCOSXML] GetObject:request];
```

署名を生成する

SDK中のいかなる要求にも署名が必要であり，アクセスユーザの身元を検証し，アクセスの安全性を確保する必要がある.署名が正しくない場合は，多くのCOSサービスがアクセスできず，403個のエラーを返す.SDKで署名を生成することができる.各要求に対して，QCloudServiceConfigurationオブジェクトの中のEclipureProviderオブジェクトから署名を要求することができる.署名を生成するためのオブジェクトは，開始時にEcliureProviderに割り当てられ，QCloudSignatureProviderプロトコルに従って署名生成方法を実現すべきである.

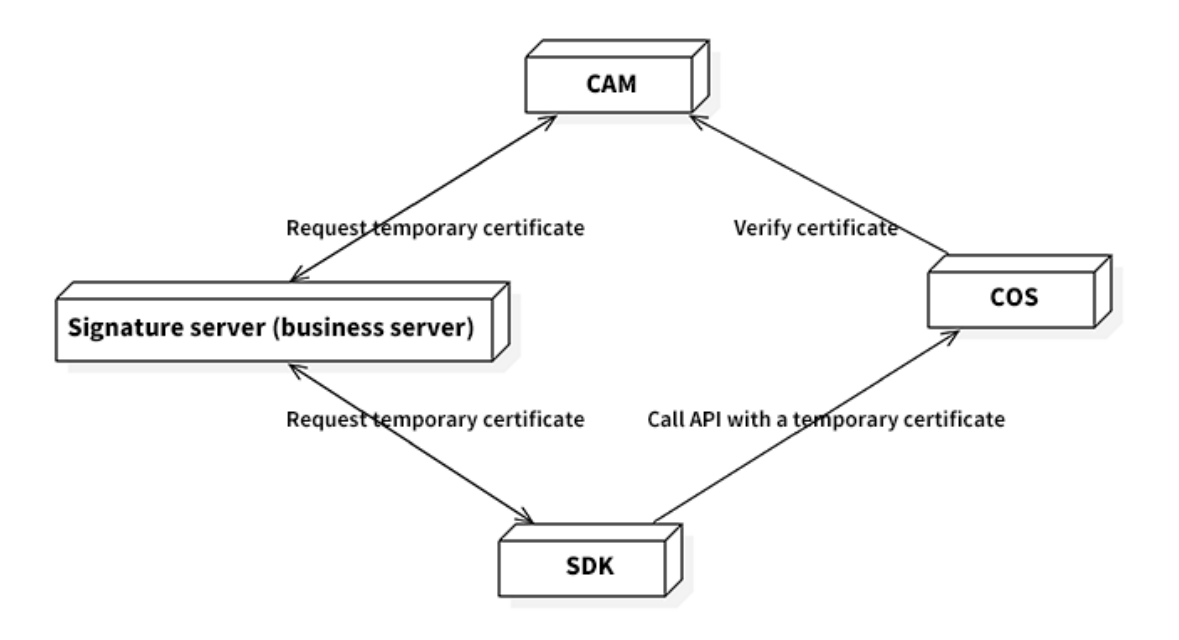
```
- (void) signatureWithFields:(QCloudSignatureFields* )fileds
request:(QCloudBizHTTPRequest* )request
urlRequest:(NSURLRequest* )urlRequst
compelete:(QCloudHTTPAuthentationContinueBlock)continueBlock
```

CAMシステムに接続して仮署名(推奨)

Permanent事務局IDと事務局鍵を持つ署名を生成するAPIは、ローカルで提供されているが、Permanent事務局IDと事務局鍵をローカルに格納することは非常に危険であり、漏洩による不必要な損失が生じる可能性があるため、サーバ上で署名を実現してセキュリティを確保することを推奨している。

> 署名の開始時間としてサーバに戻る時間を強く提案し，携帯電話のローカル時間と標準時間のずれによる署名誤りを回避する.

署名フローを実現するために、自身の署名サーバにテントレクラウドのCAM(クラウドアクセスマネージャ)を接続することをお勧めします。



CAMシステムに接続するために署名サーバをどのように設定するかについての詳細は参照されたいモバイルアプリ直接移植の実践.

署名サーバがCAMシステムに接続された後，クライアントがサーバから署名を要求すると，サーバはCAMシステムから仮証明書を要求し，それをクライアントに返す.CAMシステムは，仮の秘密ID，鍵，トークンを生成し，あなたの永久事務局IDと事務局鍵から署名を生成する.これにより，安全性が最大となる.これらの一時鍵の情報を受け取った後，端末はこれらの鍵を用いてQCloudCredentialオブジェクトを構築し，QClodCredinatedオブジェクトを介してQCloudSphereationCreatorを生成し，生成する.

```
- (void) signatureWithFields:(QCloudSignatureFields*)fileds
request:(QCloudBizHTTPRequest*)request
urlRequest:(NSURLRequest*)urlRequst
compelete:(QCloudHTTPAuthentationContinueBlock)continueBlock
{
/*Request the temporary Secret ID, Secret Key and Token from signature server*/
QCloudCredential* credential = [QCloudCredential new];
credential.secretID = @"The temporary Secret ID obtained from the CAM system";
credential.secretKey = @"The temporary Secret Key obtained from the CAM system";
credential.token = @"The Token (session ID) returned from the CAM system"
/*It is strongly recommended to return the server time as the start time of the signature, to avoid incorrect signatu
re caused by large deviation of mobile phone's local time from the standard time. */
credential.startDate = /*Returned server time*/
credential.expiretionDate = /*Signature expiration time*/
QCloudAuthentationV5Creator* creator = [[QCloudAuthentationV5Creator alloc] initWithCredential:credential];
QCloudSignature* signature = [creator signatureForData:urlRequst];
continueBlock(signature, nil);
}
```

端末の永久鍵を用いて署名を生成する(推奨しない)

永久鍵を使って端末で署名を生成しないことをお勧めします。これはデータ漏えいにつながる可能性があります。

例コードを以下に示す.

```
- (void) signatureWithFields:(QCloudSignatureFields*)fileds
request:(QCloudBizHTTPRequest*)request
urlRequest:(NSMutableURLRequest*)urlRequst
compelete:(QCloudHTTPAuthentationContinueBlock)continueBlock
{

QCloudCredential* credential = [QCloudCredential new];
credential.secretID = @"Permanent SecretID";
credential.secretKey = @"Permanent SecretKey";
QCloudAuthentationV5Creator* creator = [[QCloudAuthentationV5Creator alloc] initWithCredential:credential];
QCloudSignature* signature = [creator signatureForData:urlRequst];
continueBlock(signature, nil);
}
```

組み立てツールを使って非同期署名を管理する

今ではSDKで署名を生成してAPIを利用することができますまた、仮署名を実現しやすくするためのツールを提供し、サーバーから仮署名に必要な情報を取得することができます例えば、上のコードに従って署名を生成することができ、私たちの構築ツールQCloudCredentailFenceQueueを使って仮署名を得ることもできます。QCloudCredentailFenceQueueを使って署名を取得すれば、どの署名要求も一時署名を得ることができません。フェンシングの仕組みを提供していますので、QCloudCredentailFenceQueueを使って署名を得ることができます。
QCloudCredentailFenceQueueを用いるには，まず1つのインスタンスを生成する.

```
//AppDelegate.m
//AppDelegate must follow QCloudCredentailFenceQueueDelegate protocol
//
- (BOOL)application:(UIApplication * )application didFinishLaunchingWithOptions:(NSDictionary * )launchOptions {
// init step
self.credentialFenceQueue = [QCloudCredentailFenceQueue new];
self.credentialFenceQueue.delegate = self;
return YES;
}
```

次に，QCloudCredentailFenceQueueを呼び出すクラスは，QCloudCredentailFenceQueueDelegateを用いてプロトコルで定義された方法を実現しなければならない：

```
- (void) fenceQueue:(QCloudCredentailFenceQueue * )queue requestCreatorWithContinue:(QCloudCredentailFenceQueueContin
ue)continueBlock
```

QCloudCredentailFenceQueueを用いて署名を得る場合，SDKに署名が必要なすべての要求は，プロトコル定義の方法で署名に必要なパラメータを獲得し，有効な署名を生成するまでは実行されない.以下の例を参照されたい.

```
//AppDelegate.m
- (void) fenceQueue:(QCloudCredentailFenceQueue * )queue requestCreatorWithContinue:(QCloudCredentailFenceQueueContin
ue)continueBlock
{
QCloudCredential* credential = [QCloudCredential new];
//You can synchronize the process to obtain from the server the secretID, secretKey, expirationDate and token paramet
ers needed for the temporary signature.
credential.secretID = @"****";
credential.secretKey = @"****";
```

```
/*It is strongly recommended to return the server time as the start time of the signature, to avoid incorrect signatu
re caused by large deviation of mobile phone's local time from the standard time.*/
credential.startDate =[NSDate dateWithTimeIntervalSince1970:@"Returned server time"]
credential.experationDate = [NSDate dateWithTimeIntervalSince1970:1504183628];
credential.token = @"****";
QCloudAuthentationV5Creator* creator = [[QCloudAuthentationV5Creator alloc] initWithCredential:credential];
continueBlock(creator, nil);
}
- (void) signatureWithFields:(QCloudSignatureFields*)fileds
request:(QCloudBizHTTPRequest*)request
urlRequest:(NSMutableURLRequest*)urlRequst
compelete:(QCloudHTTPAuthentationContinueBlock)continueBlock
{
[self.credentialFenceQueue performAction:^(QCloudAuthentationCreator *creator, NSError *error) {
if (error) {
continueBlock(nil, error);
} else {
QCloudSignature* signature = [creator signatureForData:urlRequst];
continueBlock(signature, nil);
}
}];
}
```

この時、あなたは私たちが提供する構築ツールを使って、臨時署名を生成することができ、あなたは自分で署名過程を実現することができます。

SDK上のユーザーガイドを簡略化

簡略化されたSDKはアップロードとダウンロード機能のみを利用しているユーザーに提供されており、小さなサイズのSDKしかインストールできず、この簡略化されたSDKは全体のSDKサイズの半分である。

簡略化されたSDKは、CocoaPodsのSubspecで実現されているため、CocoaPodsのみで統合されています。簡略化されたSDKを使用するには、Podfileに以下のものを追加してください。

```
pod 'QCloudCOSXML/Transfer'
```

> モバイル回線ユーザーの場合、TACStorageが**使用を禁ずる**そして公式の情報源タラ置かれる**すべての情報源の前で**これをPodfileの最初の行に置くことを提案する。他のユーザは、この注釈を無視することができる。

簡略化されたSDKはヘッダファイルがないQCloudCOSXML.h.初期化時に以下のヘッダファイルを導入する.

```
#import <QCloudCOSXML/QCloudCOSXMLTransfer.h>
#import <QCloudCore/QCloudCore.h>
```

SDKの初期化を簡素化し、アップロードおよびダウンロードのためのAPIは、完全SDKの初期化とAPIと同じだ。

スピーディーな体験

最終更新日：：2019-12-31 16:42:18

同前.

すべてのツールとデモはすべて記憶しているGitHubストレージ.

本稿では，ユーザのアプリケーションサーバとユーザのアプリケーションクライアントを紹介した。

ユーザのアプリケーションサーバを設定する

本例では、ユーザのアプリケーションサーバは、1つの一時鍵のみを提供する。_署名者_lite“はフラスコのサービス枠に基づく一時鍵サービスで、python 3で書かれています。自分のマシンや仮想マシン上で動作し、あなたのCOS端末(Android/iOS)SDKに一時鍵サービスを提供することができます。

> このプロジェクトはデモンストレーションにのみ使用されており、生産環境では使用できません。COSサーバSDKの一時鍵APIを生産環境で使用することができます。

配置環境

“cossign”はPython 3のみに対応しており、pipコマンドを使って直接インストールすることができる一時鍵サービスを提供しています。

```
pip3 install cossign
```

サービスを有効にする.

インストールが完了すると、一時鍵サービスを有効にするための以下のコマンドが実行されてもよい。

```
cossign --secret_id your_secret_id --secret_key your_secret_key --port 5000
```

> デフォルトポートは5000。コマンドを実行する時にポートを指定する必要はありません。
> 重要な情報はクラウドAPIキーコンソール.

テストサービス

あなたは訪問することができます `http://your_server_ip:5000/sign` 以下の命令を実行する。

```
curl http://your_server_ip:5000/sign
```

以下の結果に戻ると、サービスは成功している。

```
{
”code”:0,
”message”:””,
”codeDesc”:”Success”,
”data”:{
”credentials”:{
”sessionToken”:”634aa09dccc3274045ba413ec081c1df64007f0a30001”,
”tmpSecretId”:”AKIDwxHZGTUvXAfcbLaOedJUQuwBXWUXG4m3”,
”tmpSecretKey”:”kriDdZsOuuF9zrZPlSAVVG0Sg4RXZu6M”},
”expiredTime”:1530515889}
}
```

故障排除ガイドライン

一時鍵作成中にエラーが発生した場合は、以下のガイドラインを参照して障害排除を行う。

1. **質問：** `comment not found pip3` インストールコマンドを実行する時に報告する `pip3 install flask` .
   分析：Python 3がインストールされていないか、Python 3のインストール経路が正しく設定されていない。
   解決策：あなたのシステムによってPython 3をインストールしたり、経路設定をチェックしたりします。

2. **質問：**サービス開始後、エラー `URLError: urlopen error [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed` 一時鍵取得時に報告する。
   解決策：この問題はMAC OS Xシステムでよく見られる.証明書モジュールをインストールすることができる.より多くの情報については，参照のこと.スタック溢れでよくある問題の解答.

ユーザのアプリケーションクライアントを設定する

配置クライアント

ファイルQCloudCOSXMLDemo/QCloudCOSXMLDemo/TestCommonDefe.hを修正し，appIdと一時鍵を取得するための配置アドレスを入力し，以下の命令を実行する.

```
pod install
```

コマンドを実行後、QCloudCOSXMLDemo.xcワークプレースを開いてデモを見る。

運行実演

バケツのリストを検索し、バケットを作成する

1. すべての利用可能領域は、デモアプリケーションが開いた後に表示される。
2. 1つの領域を選択し、現在のアカウントの下にこの領域内のすべての格納桶を表示する。**たるをつくる**ボタンは右上にあります。

書類をアップロードする

クリックでアップロードテストができます。**写真->アップロード->一時停止する.->回復する.->取り消し**下から順に。

ファイルをダウンロードする

ダウンロードインタフェースに移行すると、現在のストレージバケツにあるすべてのファイルが表示される。**ダウンロードする.**テストをしに行く。

Java SDK
かいそく入門

最終更新日：：2019-12-31 16:42:19

ダウンロードとインストール

関連資源

- COS XML Java SDKのリソースをダウンロードできるここです。.
- デモはダウンロードできますここです。.

環境要求

- SDKはJDKV 1.7とより高いバージョンをサポートする。
- JDKの実装に関する情報は、参照のことJavaのインストールと構成.

> - “事務局”“事務局キーワード”“桶”などの用語の定義については、参照のこと。COS語彙.
> - COS Java SDKでよく使われているクラスを以下のパッケージで見つけることができます：
>
> 1. クライアントの配置に関するクラスは，com.qcloud.cosパケット中にある.
> 2. 権限に関連するクラスは，com.qcloud.cos.authサブパッケージにある.
> 3. 例外に関連するクラスは，com.qcloud.cos.Exceptionサブパッケージにある.
> 4. リクエストに関連するクラスは，com.qcloud.cos.modelのサブセットにある.
> 5. 領域に関するクラスは，com.qcloud.cos.regionサブパッケージにある.
> 6. 上位APIに関連するクラスは，com.qcloud.cos.transferサブパッケージにある.

SDKのインストール

MavenまたはソースコードでSDKをインストールできます：

- Mavenでインストール
  関連する依存項をMavenプロジェクトのPom.xmlファイルに追加すると、以下のようになります。

```
<dependency>
<groupId>com.qcloud</groupId>
<artifactId>cos_api</artifactId>
<version>5.5.5</version>
</dependency>
```

- ソースコードで実装
  ダウンロードソースXML Java SDKMavenで導入しました**書類書類.>輸入>修士.>現在のMavenプロジェクトは.**

SDKのアンインストール

POM依存項やソースコードを削除することでSDKをアンインストールする.

かいそく入門

次節では、クライアントの初期化、バケット作成、クエリーバケツリスト、アップロードオブジェクト、クエリオブジェクトリスト、ダウンロードオブジェクトおよび削除オブジェクトなど、COS Java SDKを用いて基本動作を実行する方法について説明する。

導入類

COS Java SDKのパッケージ名は `com.qcloud.cos.*` あなたは、IDEツール(EclipseやIntelliJなど)を使ってプログラム実行に必要なクラスを導入することができます。

初期化クライアント

任意のCOS要求を実行する前に、COS APIを呼び出すためのCOSClientクラスのオブジェクトを生成する必要があります。COSClientインスタンスを生成すると、スレッドセキュリティを再利用することができます。プログラムやサービスが退出した場合、クライアントを閉じる必要があります。

もし永久鍵を使ってCOSClientを初期化するならばAPI鍵管理CAMコンソール内のページ。永久鍵は多くのアプリケーション方式に使用することができます。以下は一例です：

```
// 1. Initialize the user credentials (secretId, secretKey)
String secretId = "COS_SECRETID";
String secretKey = "COS_SECRETKEY";
COSCredentials cred = new BasicCOSCredentials(secretId, secretKey);
// 2. Set the bucket region. For abbreviations of COS regions, see https://cloud.tencent.com/document/product/436/622
4
// clientConfig contains the set methods to set region, HTTPS (HTTP by default), timeout, proxy, etc. For details, se
e the source code or the FAQs about Java SDK
Region region = new Region("ap-guangzhou");
ClientConfig clientConfig = new ClientConfig(region);
// 3. Generate a COS client.
COSClient cosClient = new COSClient(cred, clientConfig);
```

また、一時鍵を用いてCOSClientを初期化することもできる。一時鍵をどのように生成し、使用するかについての情報は、参照されたい一時キーの生成と使用以下に一例を示す。

```
// 1. Pass in the obtained temporary key (tmpSecretId, tmpSecretKey, sessionToken)
String tmpSecretId = "COS_SECRETID";
String tmpSecretKey = "COS_SECRETKEY";
String sessionToken = "COS_TOKEN";
BasicSessionCredentials cred = new BasicSessionCredentials(tmpSecretId, tmpSecretKey, sessionToken);
// 2. Set the bucket region. For abbreviations of COS regions, see https://cloud.tencent.com/document/product/436/622
4
// clientConfig contains the set methods to set region, HTTPS (HTTP by default), timeout, and proxy, etc. For detail
s, see the source code or the FAQs about Java SDK
Region region = new Region("ap-guangzhou");
ClientConfig clientConfig = new ClientConfig(region);
// 3. Generate a COS client
COSClient cosClient = new COSClient(cred, clientConfig);
```

ClientConfigクラスは，以下の主要メンバを含む構成情報クラスである．

会員名	設定方法	説明する.	タイプ

会員名	設定方法	説明する.	タイプ
地域.	構造関数または集合方法	COSエリアの略については、参照のことドメイン·アクセスドメイン	地域.
HTTPSプロトコル	SET術	リクエストに使用するプロトコル.デフォルトの場合，HTTPはCOSとインタラクションするために使用される.	HttpProtocol
署名が期限切れになる	SET術	請求署名の有効期間は，デフォルトで1時間とした.	INT
接続タイムアウト	SET術	COSサービスとの接続のタイムアウト時間は，デフォルトでは30秒である.	INT
ソケットタイムアウト	SET術	クライアントがデータを読み出すタイムアウト時間は，デフォルトで30秒である.	INT
httpProxyIp	SET術	プロキシIP	串
httpProxyPort	SET術	プロキシサーバポート	INT

たるをつくる

エリア·ストレージの名称を決定した場合は、以下の例を参照して、保存桶を作成する。

```
Region region = new Region(”ap-guangzhou”);
ClientConfig clientConfig = new ClientConfig(region);
COSClient cosClient = new COSClient(cred, clientConfig);
String bucket = ”examplebucket-1250000000”; // Bucket name in the format of BucketName-APPID
CreateBucketRequest createBucketRequest = new CreateBucketRequest(bucket);
// Set the bucket permission to PublicRead (public read and private write). Other options include private read/write and public read/write
createBucketRequest.setCannedAcl(CannedAccessControlList.PublicRead);
try{
Bucket bucketResult = cosClient.createBucket(createBucketRequest);
} catch (CosServiceException serverException) {
serverException.printStackTrace();
} catch (CosClientException clientException) {
clientException.printStackTrace();
}
```

クエリーバケツリスト

記憶桶リストについては、以下の例を参照されたい。

```
try {
List<Bucket> buckets = cosClient.listBuckets();
System.out.println(buckets);
} catch (CosServiceException serverException) {
serverException.printStackTrace();
} catch (CosClientException clientException) {
clientException.printStackTrace();
}
```

アップロード相手

既知の長さのローカルファイルや入力ストリームをCOSにアップロードします。これは20 MB以下の小さな画像ファイルをアップロードするのに適しています。許容される最大ファイルサイズは5 GBです。5 GBを超えるファイルに対しては、マルチロードまたは高度なAPIを使用する必要があります。

- もしあなたの多くのローカルファイルが20 MBを超えるなら、高級APIを使ってそれらをアップロードすることをお勧めします。
- COSに同じ鍵を持つオブジェクトが存在する場合，そのオブジェクトは新たにアップロードされたオブジェクトによって上書きされる.
- カタログオブジェクトを作成する場合は、ご参照くださいSDKを使ってディレクトリを作成する方法.
- オブジェクト鍵(鍵)は，バケット中のオブジェクトのユニークIDである. `examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com/images/picture.jpg` ，オブジェクトキーはimages/picture.jpgである.オブジェクトキー.

以下は5 GB以下のファイルをアップロードした例です：

```
try {
// Specify the file to be uploaded
File localFile = new File(”exampleobject”);
// Specify the destination bucket
String bucketName = ”examplebucket-1250000000”;
// Specify the key of the object to be uploaded to COS
String key = ”exampleobject”;
PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName, key, localFile);
PutObjectResult putObjectResult = cosClient.putObject(putObjectRequest);
} catch (CosServiceException serverException) {
serverException.printStackTrace();
} catch (CosClientException clientException) {
clientException.printStackTrace();
}
```

照会先リスト

タンク内のオブジェクトリストを調べるには、以下の例を参照されたい。

```
try {
String bucket = ”examplebucket-1250000000”;
ListObjectsRequest listObjectsRequest = new ListObjectsRequest();
// Set the bucket name
listObjectsRequest.setBucketName(bucket);
// Prefix indicates that the object keys to be listed begin with prefix
listObjectsRequest.setPrefix(””);
// Set the maximum number of objects to be traversed, which can be up to 1,000 in one listobject operation
listObjectsRequest.setMaxKeys(1000);
listObjectsRequest.setDelimiter(”/”);
ObjectListing objectListing = cosClient.listObjects(listObjectsRequest);
for (COSObjectSummary cosObjectSummary : objectListing.getObjectSummaries()) {
// Object path key
String key = cosObjectSummary.getKey();
// Object etag
String etag = cosObjectSummary.getETag();
// Object length
long fileSize = cosObjectSummary.getSize();
// Object storage class
String storageClass = cosObjectSummary.getStorageClass();
```

```
System.out.println("key:" + key + "; etag:" + etag + "; fileSize:" + fileSize + "; storageClass:" + storageClass);
}
} catch (CosServiceException serverException) {
serverException.printStackTrace();
} catch (CosClientException clientException) {
clientException.printStackTrace();
}
```

ダウンロード対象

オブジェクトをアップロードすると、同じキーを使ってGetObject APIを呼び出して、そのオブジェクトをダウンロードすることができます。事前に署名されたリンクを生成することもできます。(ダウンロード操作の場合は、メソッドをGETに指定してください。より多くの情報については、こちらをご参照ください。あらかじめ署名したURL)。このリンクはダウンロードを共有することができます。ただし、あなたのファイルがプライベートである場合は、必ず事前に署名されたリンクの有効期限をチェックしてください。
ファイルを指定されたローカルパスにダウンロードするには、以下の例を参照されたい。

```
try{
// Specify the bucket where the object is stored
String bucketName = "examplebucket-1250000000";
// Specify the key of the object in COS
String key = "exampleobject";
// Specify the destination local path
File downFile = new File("D:¥¥exampleobject");
GetObjectRequest getObjectRequest = new GetObjectRequest(bucketName, key);
ObjectMetadata downObjectMeta = cosClient.getObject(getObjectRequest, downFile);
} catch (CosServiceException serverException) {
serverException.printStackTrace();
} catch (CosClientException clientException) {
clientException.printStackTrace();
}
```

対象を削除する

以下のコードを用いてCOSで指定パス中のオブジェクトを削除する：

```
try {
// Specify the bucket where the object is stored
String bucketName = "examplebucket-1250000000";
// Specify the key of the object in COS
String key = "exampleobject";
cosClient.deleteObject(bucketName, key);
} catch (CosServiceException serverException) {
serverException.printStackTrace();
} catch (CosClientException clientException) {
clientException.printStackTrace();
}
```

クライアントを閉じる

以下のコードを用いてcosClientを閉じ，HTTPに接続されたバックエンド管理スレッドを解放する：

```
// Close the client (close the backend thread)
cosClient.shutdown();
```

JavaScript SDK
かいそく入門

最終更新日：：2019-12-31 17:22:32

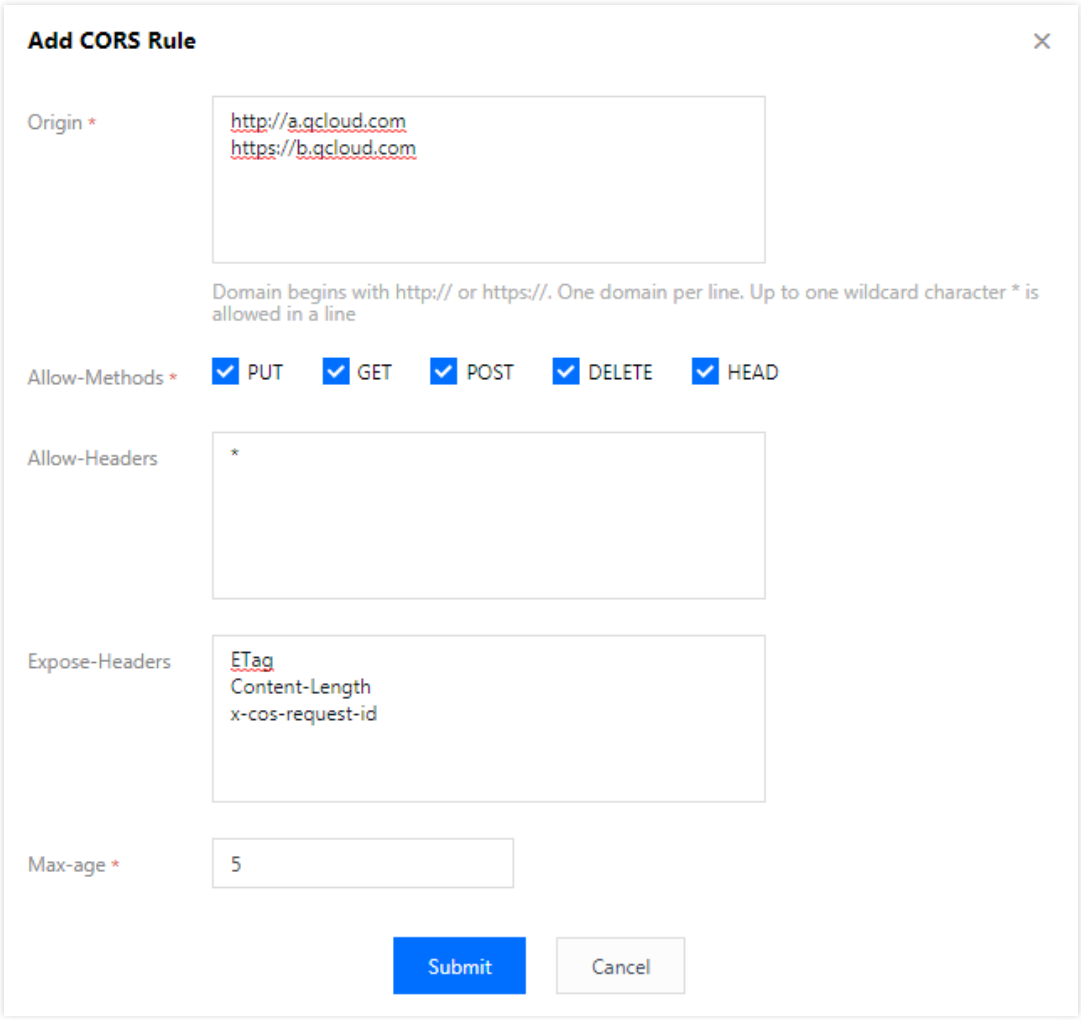
ダウンロードとインストール

関連資源

- COS XML JS SDKソースコード：ダウンロードはこちらから.
- デモ：ダウンロードはこちらから.

環境準備

1. JavaScriptSDKは、AjaxファイルアップロードとMD 5チェックサム計算のために、基本的なHTML 5特性(IE 10およびそれ以上のバージョン)をブラウザにサポートするよう求めている。
2. 登録するCOSコンソールたるをつくるそしてバケツの名前と地域名.
3. 登録するカムコンソールそしてあなたのプロジェクトの秘書と秘書のキーワードを手に入れて。
4. CORSルールを配置する。入力 `*` Expose Headersに対しては、JSが読み込む必要があるETag、Content-Length、その他のタイトルフィールドを以下のように入れている。ソース資源の共有を設置する.



事務局ID、事務局キーワード、樽その他の用語の定義については、参照のことCOS語彙.

SDKのインストール

SDKは以下のようにインストールできます：

スクリプトで導入

```
<script src="./cos-auth.min.js"></script>
```

scriptタグがSDKを参照する場合、SDKはグローバル変数名COSを占有し、その構造関数はSDKインスタンスを作成することができる。

webpackで導入

以下のコマンドを実行することでSDK依存項を実装する. `npm i cos-js-sdk-v5 --save` webpackパッケージプランをサポートします。以下のコードを使ってモジュールとしてnpmを導入することができます：

```
var COS = require('cos-js-sdk-v5');
```

かいそく入門

一時鍵を取得する

鍵をフロントに置くと、secIdとsecreKeyが公開されますので、永続鍵をバックエンドに置き、フロントエンドがajaxを通ってバックエンドから仮鍵を取得します。実際の展開では、バックエンドにあなたのウェブサイトの付加的な認証手順を追加してください。より多くの情報については、こちらを参照してください。一時キーの生成と使用.

初期化

1. web.htmlファイルを作成し、以下のコードを入力し、web.html内のストレージバレル名と領域を修正します。
2. 一時鍵サービスをバックエンドに展開し、getAuthalizationで鍵サービスアドレスを修正する。
3. web.htmlファイルをWebサーバ上に置き、ブラウザでページにアクセスし、ファイルのアップロードをテストします。web.htmlのサンプルコードは以下の通りです：

```
<input id="file-selector" type="file">
<script src="dist/cos-js-sdk-v5.min.js"></script>
<script>
var Bucket = 'examplebucket-1250000000';
var Region = 'ap-beijing';

// Initialize an instance
var cos = new COS({
getAuthorization: function (options, callback) {
// Get a temporary key asynchronously
$.get('http://example.com/server/sts.php', {
bucket: options.Bucket,
region: options.Region,
}, function (data) {
var credentials = data.credentials;
callback({
TmpSecretId: credentials.tmpSecretId,
TmpSecretKey: credentials.tmpSecretKey,
XCosSecurityToken: credentials.sessionToken,
ExpiredTime: data.expiredTime
});
});
}
});

// The COS instance can then be used to call a COS request
// TODO

</script>
```

わりつけ項

法律で例を示す

以下の方法の1つを用いてCOS SDKインスタンスを作成する：

- 方法1(推奨)：バックエンドは1つの一時鍵を取得し，それをフロントエンドに送信して署名計算を行う.

```
var COS = require('cos-js-sdk-v5');
var cos = new COS({
// Required parameters
getAuthorization: function (options, callback) {
// Server-side JS and PHP sample: https://github.com/tencentyun/cos-js-sdk-v5/blob/master/server/
// For examples of other server-side programming languages, see COS STS SDK: https://github.com/tencentyun/qcloud-cos-sts-sdk
// For the STS documentation, visit https://cloud.tencent.com/document/product/436/14048
$.get('http://example.com/server/sts.php', {
// The required parameters can be obtained from options
}, function (data) {
callback({
TmpSecretId: data.TmpSecretId,
TmpSecretKey: data.TmpSecretKey,
XCosSecurityToken: data.XCosSecurityToken,
ExpiredTime: data.ExpiredTime, // The SDK will not call getAuthorization again before the ExpiredTime time
});
});
}
});
```

- 方法2(推奨)：権限は細粒度で制御される.バックエンドは1つの一時鍵を取得してフロントエンドに送信する.フロントエンドは同一要求に対してのみその鍵を再利用し，バックエンドはスコープを通して権限を細粒度で管理することができる.

```
var COS = require('cos-js-sdk-v5');
var cos = new COS({
// Required parameters
getAuthorization: function (options, callback) {
// Server-side sample: https://github.com/tencentyun/qcloud-cos-sts-sdk/edit/master/scope.md
$.ajax({
method: 'POST',
url: ' http://example.com/server/sts-scope.php',
data: JSON.stringify(options.Scope),
beforeSend: function () {
xhr.setRequestHeader('Content-Type', 'application/json');
},
dataType: 'json',
success: function (data) {
var credentials = data.credentials;
callback({
TmpSecretId: credentials.tmpSecretId,
TmpSecretKey: credentials.tmpSecretKey,
XCosSecurityToken: credentials.sessionToken, // sessionToken needs to be passed to XCosSecurityToken
ExpiredTime: data.expiredTime,
ScopeLimit: true, // You need to set the fine-grained permission control to true to make sure that the key will only be reused for the same request
});
}
});
}
});
```

- 方法3(推奨しない)：要求ごとに先頭はgetAuthizationで署名を取得する必要があり，バックエンドは永久または仮鍵を用いて署名を計算し，先頭に戻す方法であり，複数のアップロードの権限を制御することが困難となるため，この方法を用いることは推奨されな

い.

```
var cos = new COS({
// Required parameters
getAuthorization: function (options, callback) {
// The server obtains a signature. For more information, see the COS SDK for the corresponding programming language: https://cloud.tencent.com/document/product/436/6474
// Note: There is a security risk. The backend needs to strictly control the permissions through method and pathname, such as prohibiting put /
$.get('http://example.com/server/auth.php', {
method: options.Method,
pathname: '/' + options.Key,
}, function (data) {
callback({
Authorization: data.authorization,
// XCosSecurityToken: data.sessionToken, // If a temporary key is used, sessionToken needs to be passed to XCosSecurityToken
});
});
},
// Optional parameters
FileParallelLimit: 3, // Limit the number of concurrent file uploads
ChunkParallelLimit: 3, // Limit the number of concurrent part uploads when using multipart upload for a single file
ProgressInterval: 1000, // Limit the interval of upload onProgress callbacks
});
```

- 方法4(推奨しない)：フロントエンドは固定鍵を用いて署名を計算する.この方法はフロントエンドデバッグに利用できるが，鍵の漏洩を必ず避ける.

```
var cos = new COS({
SecretId: 'COS_SECRETID',
SecretKey: 'COS_SECRETKEY',
});
```

構造関数のパラメータ

パラメータ名	説明する.	タイプ	ひっくるめ書き
秘書ID	ユーザー秘書ID	串	いえ。
分泌キー	ユーザの事務局鍵は，フロントエンドのデバッグのみに用いることを提案し，公開すべきではない.	串	いえ。
ファイル並列制限	同一例における合併ファイルの数；デフォルト値：3	番号	いえ。
春風並限法	個々のファイルの合併部品のアップロード回数；デフォルト値：3	番号	いえ。
春日時報	複数のアップロードに失敗した場合の再試行回数；デフォルト値：3(要求は、初期要求を含む合計4回発行される)	番号	いえ。
春日	多部分アップロードの各部分の大きさ(バイト単位)；デフォルト値：1,048,576(1 MB)	番号	いえ。

パラメータ名	説明する.	タイプ	ひっくるめ書き
SliceSize	アップロードファイルを用いてファイルを一括アップロードする場合、ファイルサイズがそのパラメータの値を超えると、マルチロード(SliceUploadFile)が使用され、そうでなければ、単純アップロード(PutObject)が使用される	番号	いえ。
CopyChun並列制限	多部分複製の合併部品コピーアップロード；デフォルト値：20	番号	いえ。
COPYCHUNKSize	多部分複製の各部分の大きさ(バイト単位)(SliceCopyFile)；デフォルト値：10，485，760(10 MB)	番号	いえ。
CopySliceSize	ファイルコピーについては，ファイルサイズがこのパラメータの値を超えていれば，マルチコピー(SliceCopyFile)を使用し，そうでなければ，単純コピー(PutObjectCopy)を用いる.	番号	いえ。
進歩性	アップロード進捗コールバックメソッドon Progressのコールバック周波数(ミリ秒単位)；デフォルト値：1，000	番号	いえ。
協議する.	要求を出すためのプロトコル；実効値： `https:``http:` .デフォルトでは， `http:` 現在のページで使用される `http:` そうでなければ `https:` 使われることになります	串	いえ。
サービスドメイン	GetServiceメソッドを呼び出す際のリクエストドメイン名、例えば `cos.ap-beijing.myqcloud.com`	串	いえ。
領域	Bucket呼び出しのリクエストドメイン名。テンプレートを使用することができます、例えば `”{Bucket}.cos.{Region}.myqcloud.com”`	串	いえ。
UploadQueueSize	アップロード列の最大サイズ.タスクの状態が「待ち」，「チェック」または「アップロード」でなければ，最大長を超えるタスクは除去される.デフォルト値：10，000	番号	いえ。
ForcePathStyle	送信要求には接尾辞を強制使用する.接尾辞バケットはドメインの後ろの経路名に，バケットは署名経路名に追加して計算する.デフォルト値：false	ブル	いえ。
UploadCheckContentMd5	アップロード時にContent—MD 5を検証し，デフォルトでfalseとする.このオプションを有効にすれば，アップロードされたファイルのMD 5値が計算され，大きなファイルには時間がかかる可能性がある.	ブル	いえ。
授権を得る	署名を取得するためのコールバック方法。事務局IDまたは事務局鍵がない場合には、このパラメータが必要となる	機能.	いえ。

承認コールバック関数の取得(使用方法1)

```
getAuthorization: function(options, callback) { ... }
```

許可されたコールバックパラメータを取得する

パラメータ名	説明する.	タイプ
選択肢	一時鍵取得に必要なパラメータオブジェクト	機能.
-桶だ	BucketName-appId形式のストレージバレル名	串

パラメータ名	説明する.	タイプ
-エリアだ	タンクのあるエリアです。列挙数については、こちらを参照されたいと思いますバレル領域情報	串
回撥	一時鍵を取得した後のコールバック方法。	機能.

一時鍵を取得すると，オブジェクトを返す.返すオブジェクトの属性は以下のとおりである.

属性名	パラメータ記述	タイプ	ひっくるめ書き
TmpsecId	取得した一時鍵のtmpsecId	串	はい。
プロンプトキー	取得した一時鍵のtmpuneKey	串	いえ。
XCosSecurityToken	取得した一時鍵のsessionTokenは、ヘッダ内のx-cos-security-Tokenフィールドに対応する。	串	いえ。
えんたい時間	一時鍵の期限切れ時間，すなわちタイムアウト時間が取得されている.	串	いえ。

承認コールバック関数の取得(使用方法2)

```
getAuthorization: function(options, callback) { ... }
```

許可されたコールバックパラメータを取得する

パラメータ名	説明する.	タイプ	ひっくるめ書き
選択肢	署名取得に必要なパラメータ対象	相手.	いえ。
-方法だ	当面の要求の方法	相手.	いえ。
-パス名だ	署名計算のための要求経路	串	いえ。
-キー	オブジェクトキー(オブジェクト名)は，バケット中のオブジェクトのユニークIDである.オブジェクトキー記述	串	いえ。
-了解	現在要求されている問合せパラメータオブジェクトは，フォームが{key：‘val’}である.	相手.	いえ。
-タイトルだ	現在要求されているHeaderパラメータオブジェクトは，フォーマットが{key：‘val’}である.	相手.	いえ。
回撥	一時鍵を取得した後のコールバック	機能.	いえ。

getAuthization計算が完了した後、1つの署名文字列または1つのオブジェクトを返す：
署名文字列が返される場合、この文字列は、認証ヘッダにおいて、要求によって使用されるべき証拠フィールド“許可”である。
オブジェクトを返すと，そのオブジェクトの属性は以下のようになる.

属性名	パラメータ記述	タイプ	ひっくるめ書き
授権する.	永久または一時鍵計算の授権を使用する。	串	はい。
XCosSecurityToken	取得した一時鍵のsessionTokenは、ヘッダ内のx-cos-security-Tokenフィールドに対応する。	串	いえ。

身元証明の証拠を得る

3つの方法は実例のために身分証明証拠を得ることができます：

1. 事例化の過程で，事務局IDと事務局鍵が渡され，署名が必要となるたびに，インスタンスは内部で計算される.
2. インスタンス化では，getAuthizationコールバックを渡す.署名が必要となるたびに計算され，そのコールバックによってインスタンスに返される.
3. インスタンス化では，etAuthizationコールバックが入ってくる.コールバックが呼び出された場合には，1つのインスタンス鍵を返す.鍵が切れた後，再びコールバックを呼び出す.

以下は一般的なAPIの例である。デモンストレーション.

アップロード相手

簡単なアップロードAPIは、小さなファイルをアップロードすることができます。大きなファイルについては、複数のアップロードAPIを使用してください。対象操作.

```
document.getElementById('file-selector').onchange = function () {
var file = this.files[0];
if (!file) return;
cos.putObject({
Bucket: 'examplebucket-1250000000',
Region: 'ap-beijing',
Key: 'destination path/' + file.name,
}, function (err, data) {
console.log(err || data);
});
});
```

照会先リスト

```
cos.getBucket({
Bucket: 'examplebucket-1250000000',
Region: 'ap-beijing',
Prefix: 'exampledir/', // Pass in the prefix of files to be listed here
}, function (err, data) {
console.log(err || data.Contents);
});
```

ダウンロード対象

> このAPIは，オブジェクトの内容を読み取るためのものである.ブラウザを起動してファイルをダウンロードする必要があれば，cos.getObjectUrlでURLを取得し，ブラウザでダウンロードを開始することができる.あらかじめ署名したURL.

```
cos.getObject({
Bucket: 'examplebucket-1250000000',
Region: 'ap-beijing',
Key: 'exampleobject.txt',
}, function (err, data) {
```

```
console.log(err || data.Body);
});
```

対象を削除する

```
cos.deleteObject({
Bucket: 'examplebucket-1250000000',
Region: 'ap-beijing',
Key: 'picture.jpg',
}, function (err, data) {
console.log(err || data);
});
```

Node.js SDK
かいそく入門

最終更新日：：2019-12-31 17:22:33

ダウンロードとインストール

関連資源

- JSリソースをダウンロードするCOS XML SDKここです。.
- デモをダウンロードする.ここです。.

環境要求

1. SDKを使うには、あなたの操作環境にNode.jsとNPMが必要です。
2. 登録するCOSコンソールストレージ·バケットを作成し、ストレージ·バケット名を取得し、地域名.
3. 登録するカムコンソールそしてあなたのプロジェクト秘書と秘書の鍵を手に入れて。

> パラメータの定義については，uneId，uneKey，Bucketなどを参照されたい.COS語彙.

SDKのインストール

NPMによるSDKインストール；ダウンロードNPM.

```
npm i cos-nodejs-sdk-v5 --save
```

かいそく入門

初期化

永久鍵を使って初期化する

以下の例示的なコードを参照してください。事務局ID、事務局鍵、バケツ、領域を、あなたの開発環境における実際の値で置き換えて、テストファイルをアップロードしてください。

```
var COS = require('cos-nodejs-sdk-v5');
var cos = new COS({
SecretId: 'COS_SECRETID',
SecretKey: 'COS_SECRETKEY'
});
```

一時鍵を使って初期化

jsのSDKforNode.jsは、一時鍵を用いた初期化をサポートしている。一時鍵の生成と使用に関するより多くの情報については、参照されたい一時キーの生成と使用.例コードを以下に示す.

```
var request = require('request');
var cos = new COS({
getAuthorization: function (options, callback) {
// Get a temporary key asynchronously
request({
url: '../server/sts.php',
data: {
// The required parameters can be obtained from options
}
}, function (err, response, body) {
try {
var data = JSON.parse(body);
var credentials = data.credentials;
} catch(e) {}
callback({
TmpSecretId: credentials.tmpSecretId,
TmpSecretKey: credentials.tmpSecretKey,
XCosSecurityToken: credentials.sessionToken,
ExpiredTime: data.expiredTime,
});
});
}
});
```

以下は、一般的なAPIの例である。より詳細な初期化方法については、参照されたいデモンストレーション.

わりつけ項

構造関数パラメータ記述

パラメータ名	説明する.	タイプ	ひっくるめ書き
秘書ID	ユーザー秘書ID	串	いえ。
分泌キー	フロントエンドのデバッグにはuserkeyのみを用いて鍵の漏洩を回避することを提案している.	串	いえ。
ファイル並列制限	同一例における合併ファイルの数。デフォルト値：3	番号	いえ。
春風並限法	個々のファイルの合併部品のアップロード回数；デフォルト値：3	番号	いえ。
春日時報	複数のアップロードに失敗した場合のリトライ回数.デフォルト値：3(リクエストは初期要求を含む合計4回発行される)	番号	いえ。
春日	多部分アップロードにおける部品サイズ；単位：バイト.デフォルト値：1，048，576(1 MB)	番号	いえ。
SliceSize	使用する `uploadFiles` ，ファイルサイズがこのパラメータの値を超えていれば，多部分アップロード(`sliceUploadFile` (それは)使用されるであろう。そうでなければ，安易な上乗せを用いるであろう。 `putObject`)が使用される	番号	いえ。
CopyChun並列制限	個々のファイルの合併部品のアップロード回数；デフォルト値：3	番号	いえ。
COPYCHUNKSize	複数のアップロードに失敗した場合のリトライ回数.デフォルト値：3(リクエストは初期要求を含む合計4回発行される)	番号	いえ。

パラメータ名	説明する.	タイプ	ひっくるめ書き
CopySliceSize	多部分アップロードにおける部品サイズ；単位：バイト.デフォルト値：1，048，576(1 MB)	番号	いえ。
進歩性	アップロード進行度コールバック方法のコールバック周波数 `onProgress` 単位：ミリ秒デフォルト値：1，000	番号	いえ。
協議する.	要求を出すためのプロトコル；実効値： `https:``http:` .デフォルトでは， `http:` 現在のページで使用される `http:` そうでなければ `https:` 使われることになります	串	いえ。
サービスドメイン	GetServiceメソッドが要求するドメイン名. 例： `cos.ap-beijing.myqcloud.com`	串	いえ。
領域	ストレージバレルを呼び出すためのドメイン名は、テンプレートであってもよい。 例： `{Bucket}.cos.{Region}.myqcloud.com`	串	いえ。
UploadQueueSize	キューの最大サイズ。最大値を超えると、失敗、完了、キャンセル済みのタスクをクリアします。デフォルト値：1，000	番号	いえ。
ForcePathStyle	送信要求には接尾辞を強制的に使用する。接尾辞格納バレルはドメイン名の後ろの経路名に格納され、ストレージバケットは署名経路名に追加されて計算される。デフォルト値： `false`	ブル	いえ。
UploadCheckContentMd5	ファイルのアップロードを行うためにContent—MD 5を強制的に検証し、これはファイル要求本文のMD 5チェックサムを計算し、タイトルのContent—MD 5フィールドに渡す。デフォルト値： `false`	ブル	いえ。
授権を得る	署名を取得するためのコールバック方法.uneIdやSysteKeyがなければ，このパラメータが必要となる.	機能.	いえ。

取得承認コールバック関数記述(使用方法1)

```
getAuthorization: function(options, callback) { ... }
```

許可のコールバックパラメータを取得して説明する：

パラメータ名	説明する.	タイプ
選択肢	一時鍵取得に必要なパラメータオブジェクト	機能.
-桶だ	フォーマットのストレージバレル名 `BucketName-APPID`	串
-エリアだ	枚挙値については、ご参照くださいドメイン·アクセスドメイン	串
回撥	一時鍵を取得した後のコールバック方法。	機能.

一時鍵を取得すると，オブジェクトを返す.返すオブジェクトの属性は以下のとおりである.

属性名	パラメータ記述	タイプ	ひっくるめ書き
TmpsecId	取得した一時鍵のtmpsecId	串	はい。

属性名	パラメータ記述	タイプ	ひっくるめ書き
プロンプトキー	取得した一時鍵のtmpuneKey	串	いえ。
XCosSecurityToken	取得された一時鍵のセッションフラグは、この一時鍵に対応する `x-cos-security-token` タイトル中のフィールド	串	いえ。
えんたい時間	一時鍵の期限切れ時間，すなわちタイムアウト時間が取得されている.	串	いえ。

getAuthizationコールバック関数記述(使用方法2)

```
getAuthorization: function(options, callback) { ... }
```

承認関数コール・パラメータ記述を取得する：

パラメータ名	説明する.	タイプ	ひっくるめ書き
選択肢	署名取得に必要なパラメータ対象	相手.	いえ。
-方法だ	当面の要求の方法	相手.	いえ。
-パス名だ	署名計算のための要求経路	串	いえ。
-キー	オブジェクトキー(オブジェクト名)は，バケット中のオブジェクトのユニークIDである.より多くの情報については，参照されたい.対象概要>オブジェクトキー	串	いえ。
-了解	現在要求されている問合せパラメータオブジェクトは，フォームが{key：‘val’}である.	相手.	いえ。
-タイトルだ	現在要求されているHeaderパラメータオブジェクトは，フォーマットが{key：‘val’}である.	相手.	いえ。
回撥	一時鍵を取得した後のコールバック	機能.	いえ。

getAuthization計算が完了した後、1つの署名文字列または1つのオブジェクトを返す：
署名文字列が返される場合、この文字列は、認証ヘッダにおいて、要求によって使用されるべき証拠フィールド“許可”である。
オブジェクトを返すと，そのオブジェクトの属性は以下のようになる.

属性名	パラメータ記述	タイプ	ひっくるめ書き
授権する.	署名文字列を計算する	串	はい。
XCosSecurityToken	取得された一時鍵のセッションフラグは、この一時鍵に対応する `x-cos-security-token` タイトル中のフィールド	串	いえ。

身元証明の証拠を得る

インスタンス化の過程で異なるパラメータが入力されることにより、3つの方法でインスタンスの認証証拠を得ることができる：

1. 事例化では，事務局IDと事務局鍵が渡され，署名が必要となるたびに，インスタンスは内部で計算される.
2. インスタンス化では，着信getAuthizationコールバックは，署名が必要になるたびに署名を計算し，そのコールバックでインスタンスに返す.
3. インスタンス化の間，着信getSTSコールバックは，一時鍵が必要となるたびに，そのコールバックによってインスタンスに返され，要求のたびにインスタンス内で署名計算が行われる.

たるをつくる

```
cos.putBucket({
Bucket: 'examplebucket-1250000000',
Region: 'ap-beijing',
}, function (err, data) {
console.log(err || data);
});
```

クエリーバケツリスト

```
cos.getService(function (err, data) {
console.log(data && data.Buckets);
});
```

アップロード相手

このAPIは、アップロード小ファイルに適用されます。大ファイルの場合は、複数のアップロードAPIをご利用ください。より多くの情報については、ご参照ください対象操作.

```
cos.putObject({
Bucket: 'examplebucket-1250000000',
Region: 'ap-beijing',
Key: 'destination path/picture.jpg',
Body: fs.createReadStream('./picture.jpg'), // Pass in the prefix here
}, function (err, data) {
console.log(err || data);
});
```

照会先リスト

```
cos.getBucket({
Bucket: 'examplebucket-1250000000',
Region: 'ap-beijing',
Prefix: 'exampledir/', // Pass in the prefix of the files to be listed
}, function (err, data) {
console.log(err || data.Contents);
});
```

ダウンロード対象

```
cos.getObject({
Bucket: 'examplebucket-1250000000',
Region: 'ap-beijing',
Key: 'exampleobject.txt',
Output: fs.createWriteFile(__dirname + '/exampleobject.txt'),
}, function (err, data) {
console.log(err || data.Body);
});
```

対象を削除する

```
cos.deleteObject({
Bucket: 'examplebucket-1250000000',
Region: 'ap-beijing',
Key: 'picture.jpg',
}, function (err, data) {
console.log(err || data);
});
```

PHP SDK
かいそく入門

最終更新日：：2019-12-31 17:22:35

ダウンロードとインストール

関連資源

- COS XML PHP SDKソースコード：ダウンロードはこちらから.
- デモ：ダウンロードはこちらから.

環境要求

- PHP 5.3+
  あなたは運行することができます `php -v` コマンドは現在のPHPバージョンを見る。
- 縮れ縮れ
  あなたは運行することができます `php -m` 命令は，cURL拡張が正しく実装されているかどうかを検査する.
- UbuntuでAPT-GETパケットマネージャを使ってPHPのcurl拡張をインストールして次のように命令できます：

```
sudo apt-get install php-curl
```

- CentOSではyumパッケージマネージャを使ってPHPのcurl拡張をインストールできます。

```
sudo yum install php-curl
```

SDKのインストール

3つの方法でSDKをインストールすることができますライタ方法Phar法そしてソースコード方法.

ライタ方法

Composerを使ってcos-php-sdk-v 5をインストールすることを提案し、ComposerはPHP依存項マネージャであり、プロジェクトに必要な依存項を宣言し、自動的にそれらをプロジェクトにインストールすることを可能にします。

> Composerをどのようにインストールするか、自動ロードをどのように構成するか、依存項を定義する他のベストプラクティスなど、より多くの情報を以下の位置で見つけることができる。作曲家公式サイト.

設置手順

1. 端末を起動する。
2. Composerをダウンロードして以下のコマンドを実行する.

```
curl -sS https://getcomposer.org/installer | php
```

3. 作成名は `composer.json` 内容は以下の通り。

```
{
"require": {
"qcloud/cos-sdk-v5": ">=1.0"
}
}
```

4. Composer付きSDKの実装は以下のコマンドを実行する.

```
php composer.phar install
```

この命令は1つの `vendor` フォルダにはSDKの依存関係ライブラリと `autoload.php` プロジェクト中の未来呼び出しのためのスクリプトファイル。
5.autoloaderスクリプトを用いてcos—php—sdk—v 5を呼び出す.

```
require '/path/to/sdk/vendor/autoload.php';
```

現在、プロジェクトでCOS XML PHP SDKをご利用いただけます。

Phar法

Phar手法を用いてSDKを実装すると，以下のようになる.

1. Pharファイルをダウンロードここです。.
2. Pharファイルをコードにインポート：

```
require '/path/to/cos-sdk-v5.phar';
```

ソースコード方法

ソースコードを用いてSDKを実装すると，以下のようになる.

1. ダウンロードする. `cos-sdk-v5.tar.gz` 包装する.ここです。（注： `Source code` パケットはGitHubによって圧縮されたデフォルトのパケットであり、含まれていない `vendor` 目次）
2. ストレス解消後 `autoload.php` 脚本：

```
require '/path/to/sdk/vendor/autoload.php';
```

かいそく入門

次節では、クライアントの初期化、バケット作成、クエリーバケツリスト、アップロードオブジェクト、クエリオブジェクトリスト、ダウンロードオブジェクトおよび削除オブジェクトなど、COS PHP SDKを用いて基本動作を実行する方法について説明する。

初期化

```
$secretId = "COS_SECRETID"; //"TencentCloud API key's SecretId";
$secretKey = "COS_SECRETKEY"; //"TencentCloud API key's SecretKey";
$region = "ap-beijing"; // Set a default bucket region
$cosClient = new Qcloud¥Cos¥Client(
array(
```

```
'region' => $region,
'schema' => 'https', // Protocol header; http by default
'credentials'=> array(
'secretId' => $secretId ,
'secretKey' => $secretKey)));
```

もしあなたが使うなら仮鍵以下のように例を作成する.

```
$tmpSecretId = "COS_SECRETID"; //"Temporary key's SecretId";
$tmpSecretKey = "COS_SECRETKEY"; //"Temporary key's SecretKey";
$tmpToken = "COS_TOKEN"; //"Temporary key's token";
$region = "ap-beijing"; // Set a default bucket region
$cosClient = new Qcloud¥Cos¥Client(
array(
'region' => $region,
'schema' => 'https', // Protocol header; http by default
'credentials'=> array(
'secretId' => $tmpSecretId,
'secretKey' => $tmpSecretKey,
'token' => $tmpToken)));
```

たるをつくる

```
try {
$bucket = "examplebucket-1250000000"; // Bucket name in the format of BucketName-APPID
$result = $cosClient->createBucket(array('Bucket' => $bucket));
// Request succeeded
print_r($result);
} catch (¥Exception $e) {
// Request failed
echo($e);
}
```

クエリーバケツリスト

```
try {
// Request succeeded
$result = $cosClient->listBuckets();
print_r($result);
} catch (¥Exception $e) {
// Request failed
echo($e);
}
```

アップロード相手

- putObject APIアップロードファイル(最大5 GB)を使用します。
- Upload APIを使って部品にファイルをアップロードする。

```
# Upload a file
## putObject (upload API which can upload files of up to 5 GB)
### Upload strings in memory
try {
```

```
$bucket = "examplebucket-1250000000"; // Bucket name in the format of BucketName-APPID
$key = "exampleobject";
$result = $cosClient->putObject(array(
'Bucket' => $bucket,
'Key' => $key,
'Body' => 'Hello World!'));
print_r($result);
} catch (¥Exception $e) {
echo "$e¥n";
}

### Upload a File Stream
try {
$bucket = "examplebucket-1250000000"; // Bucket name in the format of BucketName-APPID
$key = "exampleobject";
$srcPath = "F:/exampleobject";// Absolute path to the local file
$result = $cosClient->putObject(array(
'Bucket' => $bucket,
'Key' => $key,
'Body' => fopen($srcPath, 'rb')));
print_r($result);
} catch (¥Exception $e) {
echo "$e¥n";
}

### Set header and meta
try {
$bucket = "examplebucket-1250000000"; // Bucket name in the format of BucketName-APPID
$key = "exampleobject";
$srcPath = "F:/exampleobject";// Absolute path to the local file
$result = $cosClient->putObject(array(
'Bucket' => $bucket,
'Key' => $key,
'Body' => fopen($srcPath, 'rb'),
'ACL' => 'string',
'CacheControl' => 'string',
'ContentDisposition' => 'string',
'ContentEncoding' => 'string',
'ContentLanguage' => 'string',
'ContentLength' => integer,
'ContentType' => 'string',
'Expires' => 'mixed type: string (date format)|int (unix timestamp)|¥DateTime',
'GrantFullControl' => 'string',
'GrantRead' => 'string',
'GrantWrite' => 'string',
'Metadata' => array(
'string' => 'string',
),
'StorageClass' => 'string'));
print_r($result);
} catch (¥Exception $e) {
echo "$e¥n";
}

## Upload (advanced upload API, which uses multipart upload by default and can upload files of up to 50 TB)
### Upload strings in memory
try {
```

```
$bucket = "examplebucket-1250000000"; // Bucket name in the format of BucketName-APPID
$key = "exampleobject";
$result = $cosClient->Upload(
$bucket = $bucket,
$key = $key,
$body = 'Hello World!');
print_r($result);
} catch (¥Exception $e) {
echo "$e¥n";
}

### Upload a File Stream
try {
$bucket = "examplebucket-1250000000"; // Bucket name in the format of BucketName-APPID
$key = "exampleobject";
$srcPath = "F:/exampleobject";// Absolute path to the local file
$result = $cosClient->Upload(
$bucket = $bucket,
$key = $key,
$body = fopen($srcPath, 'rb'));
print_r($result);
} catch (¥Exception $e) {
echo "$e¥n";
}

### Set header and meta
try {
$bucket = "examplebucket-1250000000"; // Bucket name in the format of BucketName-APPID
$key = "exampleobject";
$srcPath = "F:/exampleobject";// Absolute path to the local file
$result = $cosClient->Upload(
$bucket= $bucket,
$key = $key,
$body = fopen($srcPath, 'rb'),
$options = array(
'ACL' => 'string',
'CacheControl' => 'string',
'ContentDisposition' => 'string',
'ContentEncoding' => 'string',
'ContentLanguage' => 'string',
'ContentLength' => integer,
'ContentType' => 'string',
'Expires' => 'mixed type: string (date format)|int (unix timestamp)|¥DateTime',
'GrantFullControl' => 'string',
'GrantRead' => 'string',
'GrantWrite' => 'string',
'Metadata' => array(
'string' => 'string',
),
'StorageClass' => 'string'));
print_r($result);
} catch (¥Exception $e) {
echo "$e¥n";
}
```

照会先リスト

```
try {
$bucket = "examplebucket-1250000000"; // Bucket name in the format of BucketName-APPID
$result = $cosClient->listObjects(array(
'Bucket' => $bucket
));
// Request succeeded
foreach ($result['Contents'] as $rt) {
print_r($rt);
}
} catch (¥Exception $e) {
// Request failed
echo($e);
}
```

そうだ `listObjects` APIは最大1，000個のオブジェクトを問い合わせることができ，すべてのオブジェクトを調べるためにはそれを繰り返し呼び出す必要がある.

```
try {
$bucket = "examplebucket-1250000000"; // Bucket name in the format of BucketName-APPID
$prefix = ''; // Prefix of the objects to be listed
$marker = ''; // End marker in the last list
while (true) {
$result = $cosClient->listObjects(array(
'Bucket' => $bucket,
'Marker' => $marker,
'MaxKeys' => 1000 // Set the maximum number of entries to be listed in one single query, which is up to 1,000
));
foreach ($result['Contents'] as $rt) {
// Print key
echo($rt['Key'] . "¥n");
}
$marker = $result['NextMarker']; // Set a new end marker
if (!$result['IsTruncated']) {
break; // Determine whether the query is completed
}
}
} catch (¥Exception $e) {
echo($e);
}
```

ダウンロード対象

- getObject APIを使ってファイルをダウンロードする。
- getObjectUrl APIを使ってファイルダウンロードURLを取得する。

```
# Download a file
## getObject (for file download)
### Download to memory
try {
$bucket = "examplebucket-1250000000"; // Bucket in the format of BucketName-APPID
$key = "exampleobject"; // Location of the object in the bucket, i.e., the object key
$result = $cosClient->getObject(array(
'Bucket' => $bucket,
'Key' => $key));
```

```
// Request succeeded
echo($result['Body']);
} catch (¥Exception $e) {
// Request failed
echo "$e¥n";
}

### Download to the local file system
try {
$bucket = "examplebucket-1250000000"; // Bucket in the format of BucketName-APPID
$key = "exampleobject"; // Location of the object in the bucket, i.e., the object key
$localPath = @"F:/exampleobject";// Download to the specified local path
$result = $cosClient->getObject(array(
'Bucket' => $bucket,
'Key' => $key,
'SaveAs' => $localPath));
} catch (¥Exception $e) {
// Request failed
echo "$e¥n";
}

### Specify the download range
/*
* Range field is in the format of 'bytes=a-b'
*/
try {
$bucket = "examplebucket-1250000000"; // Bucket in the format of BucketName-APPID
$key = "exampleobject"; // Location of the object in the bucket, i.e., the object key
$localPath = @"F:/exampleobject";// Download to the specified local path
$result = $cosClient->getObject(array(
'Bucket' => $bucket,
'Key' => $key,
'Range' => 'bytes=0-10',
'SaveAs' => $localPath));
} catch (¥Exception $e) {
// Request failed
echo "$e¥n";
}

### Set the return header
try {
$bucket = "examplebucket-1250000000"; // Bucket in the format of BucketName-APPID
$key = "exampleobject"; // Location of the object in the bucket, i.e., the object key
$localPath = @"F:/exampleobject";// Download to the specified local path
$result = $cosClient->getObject(array(
'Bucket' => $bucket,
'Key' => $key,
'ResponseCacheControl' => 'string',
'ResponseContentDisposition' => 'string',
'ResponseContentEncoding' => 'string',
'ResponseContentLanguage' => 'string',
'ResponseContentType' => 'string',
'ResponseExpires' => 'mixed type: string (date format)|int (unix timestamp)|¥DateTime',
'SaveAs' => $localPath));
} catch (¥Exception $e) {
// Request failed
echo "$e¥n";
```

```
}

## getObjectUrl (for getting file URL)
try {
$bucket = "examplebucket-1250000000"; // Bucket in the format of BucketName-APPID
$key = "exampleobject"; // Location of the object in the bucket, i.e., the object key
$signedUrl = $cosClient->getObjectUrl($bucket, $key, '+10 minutes');
// Request succeeded
echo $signedUrl;
} catch (¥Exception $e) {
// Request failed
print_r($e);
}
```

対象を削除する

```
# Delete an object
## deleteObject
try {
$bucket = "examplebucket-1250000000"; // Bucket in the format of BucketName-APPID
$key = "exampleobject"; // Location of the object in the bucket, i.e., the object key
$result = $cosClient->deleteObject(array(
'Bucket' => $bucket,
'Key' => $key,
'VersionId' => 'string'
));
// Request succeeded
print_r($result);
} catch (¥Exception $e) {
// Request failed
echo($e);
}
# Delete multiple objects
## deleteObjects
try {
$bucket = "examplebucket-1250000000"; // Bucket in the format of BucketName-APPID
$key1 = "exampleobject1"; // Location of the object in the bucket, i.e., the object key
$key2 = "exampleobject2"; // Location of the object in the bucket, i.e., the object key
$result = $cosClient->deleteObjects(array(
'Bucket' => $bucket,
'Objects' => array(
array(
'Key' => $key1,
),
array(
'Key' => $key2,
),
//...
),
));
// Request succeeded
print_r($result);
} catch (¥Exception $e) {
// Request failed
echo($e);
}
```

あらかじめ署名したURL

最終更新日：：2019-12-31 17:22:35

概述

SDK for PHPは、リクエストの事前署名URLを取得するためのAPIを提供する。

永久鍵を用いて要求を事前署名する例

アップロード要求例

```
$secretId = "COS_SECRETID"; // Replace with your permanent key's SecretId
$secretKey = "COS_SECRETKEY"; // Replace with your permanent key's SecretKey
$region = "ap-beijing"; // Set a default bucket region
$cosClient = new Qcloud¥Cos¥Client(
array(
'region' => $region,
'schema' => 'https', // Protocol header; http by default
'credentials'=> array(
'secretId' => $secretId ,
'secretKey' => $secretKey)));
### Pre-signed Simple Upload
try {
$command = $cosClient->getCommand('putObject', array(
'Bucket' => "examplebucket-1250000000", // Bucket in the format of BucketName-APPID
'Key' => "exampleobject", // Position of the object in the bucket, i.e., the object key
'Body' => '', //
));
$signedUrl = $command->createPresignedUrl('+10 minutes');
// Request succeeded
echo ($signedUrl);
} catch (¥Exception $e) {
// Request failed
echo($e);
}

### Pre-signed Multipart Upload
try {
$command = $cosClient->getCommand('uploadPart', array(
'Bucket' => "examplebucket-1250000000", // Bucket in the format of BucketName-APPID
'Key' => "exampleobject", // Position of the object in the bucket, i.e., the object key
'UploadId' => '',
'PartNumber' => '1',
'Body' => '',
));
$signedUrl = $command->createPresignedUrl('+10 minutes');
// Request succeeded
echo ($signedUrl);
} catch (¥Exception $e) {
// Request failed
```

```
echo($e);
}
```

ダウンロード要求例

```
$secretId = "COS_SECRETID"; // Replace with your permanent key's SecretId
$secretKey = "COS_SECRETKEY"; // Replace with your permanent key's SecretKey
$region = "ap-beijing"; // Set a default bucket region
$cosClient = new Qcloud¥Cos¥Client(
array(
'region' => $region,
'schema' => 'https', // Protocol header; http by default
'credentials'=> array(
'secretId' => $secretId ,
'secretKey' => $secretKey)));
### Pre-signed Simple Download
try {
$command = $cosClient->getCommand('getObject', array(
'Bucket' => "examplebucket-1250000000", // Bucket in the format of BucketName-APPID
'Key' => "exampleobject" // Position of the object in the bucket, i.e., the object key
));
$signedUrl = $command->createPresignedUrl('+10 minutes');
// Request succeeded
echo ($signedUrl);
} catch (¥Exception $e) {
// Request failed
echo($e);
}

### Getting Download Signature Using the Encapsulated getObjectUrl
try {
$bucket = "examplebucket-1250000000"; // Bucket in the format of BucketName-APPID
$key = "exampleobject"; // Position of the object in the bucket, i.e., the object key
$signedUrl = $cosClient->getObjectUrl($bucket, $key, '+10 minutes');
// Request succeeded
echo $signedUrl;
} catch (¥Exception $e) {
// Request failed
print_r($e);
}
```

一時鍵を用いて要求を事前署名する例

アップロード要求例

```
$tmpSecretId = "COS_SECRETID"; // Replace with your temporary key's SecretId
$tmpSecretKey = "COS_SECRETKEY"; // Replace with your temporary key's SecretKey
$tmpToken = "COS_TOKEN"; // Replace with your temporary key's token
$region = "ap-beijing"; // Set a default bucket region
$cosClient = new Qcloud¥Cos¥Client(
array(
'region' => $region,
'schema' => 'https', // Protocol header; http by default
```

```
'credentials'=> array(
'secretId' => $tmpSecretId,
'secretKey' => $tmpSecretKey,
'token' => $tmpToken)));
### Pre-signed Simple Upload
try {
$command = $cosClient->getCommand('putObject', array(
'Bucket' => "examplebucket-1250000000", // Bucket in the format of BucketName-APPID
'Key' => "exampleobject", // Position of the object in the bucket, i.e., the object key
'Body' => '',
));
$signedUrl = $command->createPresignedUrl('+10 minutes');
// Request succeeded
echo ($signedUrl);
} catch (¥Exception $e) {
// Request failed
echo($e);
}

### Pre-signed Multipart Upload
try {
$command = $cosClient->getCommand('uploadPart', array(
'Bucket' => "examplebucket-1250000000", // Bucket in the format of BucketName-APPID
'Key' => "exampleobject", // Position of the object in the bucket, i.e., the object key
'UploadId' => '',
'PartNumber' => '1',
'Body' => '',
));
$signedUrl = $command->createPresignedUrl('+10 minutes');
// Request succeeded
echo ($signedUrl);
} catch (¥Exception $e) {
// Request failed
echo($e);
}
```

ダウンロード要求例

```
$tmpSecretId = "COS_SECRETID"; // Replace with your temporary key's SecretId
$tmpSecretKey = "COS_SECRETKEY"; // Replace with your temporary key's SecretKey
$tmpToken = "COS_TOKEN"; // Replace with your temporary key's token
$region = "ap-beijing"; // Set a default bucket region
$cosClient = new Qcloud¥Cos¥Client(
array(
'region' => $region,
'schema' => 'https', // Protocol header; http by default
'credentials'=> array(
'secretId' => $tmpSecretId,
'secretKey' => $tmpSecretKey,
'token' => $tmpToken)));
### Pre-signed Simple Download
try {
$command = $cosClient->getCommand('getObject', array(
'Bucket' => "examplebucket-1250000000", // Bucket in the format of BucketName-APPID
'Key' => "exampleobject" // Position of the object in the bucket, i.e., the object key
));
```

```
$signedUrl = $command->createPresignedUrl('+10 minutes');
// Request succeeded
echo ($signedUrl);
} catch (¥Exception $e) {
// Request failed
echo($e);
}

### Getting Download Signature Using the Encapsulated getObjectUrl
try {
$bucket = "examplebucket-1250000000"; // Bucket in the format of BucketName-APPID
$key = "exampleobject"; // Position of the object in the bucket, i.e., the object key
$signedUrl = $cosClient->getObjectUrl($bucket, $key, '+10 minutes');
// Request succeeded
echo $signedUrl;
} catch (¥Exception $e) {
// Request failed
print_r($e);
}
```

Python SDK
かいそく入門

最終更新日：：2019-12-31 17:22:36

ダウンロードとインストール

関連資源

COS XML Python SDKのリソースをダウンロードできるここです。.
デモをダウンロードできますここです。.

環境要求

COS XML Python SDKは現在、Python 2.6、2.7、3.xをサポートしている。

> パラメータの定義については，uneId，uneKey，Bucketなどを参照されたい.COS語彙.

SDKのインストール

3つの方法でSDKをインストールすることができ、pipインストール、手動インストール、オフラインでインストールすることができます。

パイプで取り付ける(推奨)

```
pip install -U cos-python-sdk-v5
```

手動取り付け

ダウンロードソースここです。そして，以下のコマンドを実行することにより，インストールプログラムによりSDKを手動でインストールする.

```
python setup.py install
```

オフライン実装

```
# Run the following commands on a computer connected to the Internet
mkdir cos-python-sdk-packages
pip download cos-python-sdk-v5 -d cos-python-sdk-packages
tar -czvf cos-python-sdk-packages.tar.gz cos-python-sdk-packages

# Copy the installation package to a computer that is not connected to the Internet and run the following commands
# Please make sure that the Python versions on the two computers are the same; otherwise, the installation will fail
tar -xzvf cos-python-sdk-packages.tar.gz
pip install cos-python-sdk-v5 --no-index -f cos-python-sdk-packages
```

かいそく入門

次節では、クライアントの初期化、バケット作成、クエリーバケツリスト、アップロードオブジェクト、クエリオブジェクトリスト、ダウンロードオブジェクトおよび削除オブジェクトなど、COS Python SDKを用いて基本動作を実行する方法について説明する。

初期化

以下の例を参照。

```
# appid has been removed from the configuration. Please add the appid to the Bucket parameter which is in the format
of BucketName-APPID
# 1. Set user profile, including secretId, secretKey, and Region
# -*- coding=utf-8
from qcloud_cos import CosConfig
from qcloud_cos import CosS3Client
import sys
import logging

logging.basicConfig(level=logging.INFO, stream=sys.stdout)

secret_id = 'COS_SECRETID' # Replace with your secretId
secret_key = 'COS_SECRETKEY' # Replace with your secretKey
region = 'ap-beijing' # Replace with your region
token = None # If a temporary key is used, Token needs to be passed in. This is optional and is null by default
scheme = 'https' # Specify whether to use HTTP or HTTPS protocol to access COS. This is optional and is https by defa
ult
config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=token, Scheme=scheme)
# 2. Get the client object
client = CosS3Client(config)
# Refer to the description below or the demo. For more information, see https://github.com/tencentyun/cos-python-sdk-
v5/blob/master/qcloud_cos/demo.py
```

ランタイムキーの生成と利用に関する情報は、参照のこと。一時キーの生成と使用.

たるをつくる

```
response = client.create_bucket(
Bucket='examplebucket-1250000000'
)
```

クエリーバケツリスト

```
response = client.list_buckets(
)
```

アップロード相手

```
#### Simple Upload of a File Stream
file_name = 'test.txt'
# It is strongly recommended to open the file in binary mode; otherwise, an error may occur
with open('test.txt', 'rb') as fp:
response = client.put_object(
Bucket='examplebucket-1250000000',
Body=fp,
Key=file_name,
StorageClass='STANDARD',
```

```
EnableMD5=False
)
print(response['ETag'])

#### Simple Upload of a Byte Stream
response = client.put_object(
Bucket='examplebucket-1250000000',
Body=b'bytes',
Key=file_name,
EnableMD5=False
)
print(response['ETag'])


#### Simple Chunk Upload
import requests
stream = requests.get('https://intl.cloud.tencent.com/document/product/436/7778')

# Online streams will be transferred to COS in the Transfer-Encoding:chunked manner
response = client.put_object(
Bucket='examplebucket-1250000000',
Body=stream,
Key=file_name
)
print(response['ETag'])

#### Advanced Upload API (Recommended)
It can automatically select simple upload or multipart upload based on the file size. Multipart upload supports uploa
d resumption.
response = client.upload_file(
Bucket='examplebucket-1250000000',
LocalFilePath='local.txt',
Key=file_name,
PartSize=1,
MAXThread=10,
EnableMD5=False
)
print(response['ETag'])
```

照会先リスト

```
response = client.list_objects(
Bucket='examplebucket-1250000000',
Prefix='folder1'
)
```

そうだ `list_objects` APIは最大1，000個のオブジェクトを問い合わせることができ，すべてのオブジェクトを調べるためにはそれを繰り返し呼び出す必要がある.

```
marker = ""
while True:
response = client.list_objects(
Bucket='examplebucket-1250000000',
Prefix='folder1',
Marker=marker
```

```
)
print(response['Contents'])
if response['IsTruncated'] == 'false':
break
marker = response['NextMarker']
```

ダウンロード対象

```
#### Download to the local file system
response = client.get_object(
Bucket='examplebucket-1250000000',
Key=file_name,
)
response['Body'].get_stream_to_file('output.txt')

#### Download a file stream
response = client.get_object(
Bucket='examplebucket-1250000000',
Key=file_name,
)
fp = response['Body'].get_raw_stream()
print(fp.read(2))

#### Set the response HTTP header
response = client.get_object(
Bucket='examplebucket-1250000000',
Key=file_name,
ResponseContentType='text/html; charset=utf-8'
)
print response['Content-Type']
fp = response['Body'].get_raw_stream()
print(fp.read(2))

#### Specify the download range
response = client.get_object(
Bucket='examplebucket-1250000000',
Key=file_name,
Range='bytes=0-10'
)
fp = response['Body'].get_raw_stream()
print(fp.read())
```

対象を削除する

```
# Delete an object
## deleteObject
response = client.delete_object(
Bucket='examplebucket-1250000000',
Key='exampleobject'
)

# Delete multiple objects
## deleteObjects
response = client.delete_objects(
Bucket='examplebucket-1250000000',
```

```
Delete={
'Object': [
{
'Key': 'exampleobject1',
},
{
'Key': 'exampleobject2',
},
],
'Quiet': 'true'|'false'
}
)
```

Mini Program SDK
かいそく入門

最終更新日：：2019-12-31 17:33:05

ダウンロードとインストール

関連資源

- COS XML SDK for WeChat Miniプログラムのソースコードをダウンロードここです。.
- デモをダウンロードする.ここです。.

環境要求

1. このSDKはWeChatアプレットにのみ適用される。
2. 登録するCOSコンソールストレージ·バケットを作成し、ストレージ·バケット名を取得し、地域情報.
3. 登録するカムコンソールそしてあなたのプロジェクト秘書と秘書の鍵を手に入れて。

> パラメータの定義については，uneId，uneKey，Bucketなどを参照されたい.COS語彙.

SDKのインストール

WeChatミニプログラムSDKのインストールには，手動での実装とNPMによる実装の2つの方式があるが，以下のようになる.

手動取り付け

複製するCOS-WX-SDK-v 5.jsファイルのソースコードディレクトリはあなたのWeChatミニプログラムのコードディレクトリに。コードは1つの相対経路で引用することができます：

```
var COS = require('./cos-wx-sdk-v5.js')
```

NPMによる実装

WeChatミニプログラムコードがwebpackを使って梱包されている場合は、npmで依存項をインストールしてください：

```
npm install cos-wx-sdk-v5
```

WeChatミニプログラムコード `var COS = require('cos-wx-sdk-v5');` .

かいそく入門

WeChatミニプログラムのホワイトリストを配置します

WeChatミニプログラムでCOSを要求するには、登録する必要がありますWeChat公式アカウントプラットフォームドメイン名のホワイトリストを配置します**発展する.>開発設定**.SDKは以下の2つのAPIを使用する：

1. cos.postObjectにはwx.ploadFileAPIをご利用ください。
2. 他の方法についてはwx.request APIをご利用ください。

対応するホワイトリストにCOSドメインを配置する必要があります。2つの形式のホワイトリストドメインがあります。

1. 標準的な要求については、例えば、ストレージバレルドメイン名をホワイトリストドメイン名として構成することができる：`examplebucket-1250000000.cos.ap-guangzhou.myqcloud.com`.
2. WeChatミニプログラムに複数のバレルが使用されている場合は、着信によって接尾辞COS要求を選択することができます `ForcePathStyle: true` SDKをインスタンス化する場合、この例では、ドメインドメイン名をホワイトリストドメイン名、例えばホワイトリストドメインとして構成する必要があります `cos.ap-guangzhou.myqcloud.com`.

初期化

```
var cos = new COS({
// ForcePathStyle: true, // If multiple buckets are used, you can use suffixed requests to reduce the number of white listed domain names to be configured; and the region domain name will be used for requests
getAuthorization: function (options, callback) {
// Get a temporary key asynchronously
wx.request({
url: 'https://example.com/server/sts.php',
data: {
bucket: options.Bucket,
region: options.Region,
},
dataType: 'text',
success: function (result) {
var data = result.data;
var credentials = data.credentials;
callback({
TmpSecretId: credentials.tmpSecretId,
TmpSecretKey: credentials.tmpSecretKey,
XCosSecurityToken: credentials.sessionToken,
ExpiredTime: data.expiredTime,
});
}
});
}
});

// Next you can use a COS instance to call a COS request
// TODO
```

わりつけ項

前例を用いる

以下の方法を用いてCOS SDKインスタンスを作成する：

- 方法1(推奨)：バックエンドは1つの一時鍵を取得し，それをフロントエンドに送信して署名計算を行う.

```
var COS = require('cos-nodejs-sdk-v5');
var cos = new COS({
// Required parameters
getAuthorization: function (options, callback) {
// Server-side examples for JS and PHP: https://github.com/tencentyun/cos-js-sdk-v5/blob/master/server/
// For server-side examples for other programming languages, see the COS SDK for STS: https://github.com/tencentyun/qcloud-cos-sts-sdk
```

```
// For the STS documentation, visit https://cloud.tencent.com/document/product/436/14048
wx.request({
url: 'https://example.com/server/sts.php',
data: {
// The required parameters can be obtained from options
},
success: function (result) {
var data = result.data;
var credentials = data.credentials;
callback({
TmpSecretId: credentials.tmpSecretId,
TmpSecretKey: credentials.tmpSecretKey,
XCosSecurityToken: credentials.sessionToken,
ExpiredTime: data.expiredTime,
});
}
});
}
});
```

- 方法2(推奨)：権限は細粒度で制御される.バックエンドは1つの一時鍵を取得してフロントエンドに送信する.フロントエンドは同一要求に対してのみその鍵を再利用し，バックエンドはスコープを通して権限を細粒度で管理することができる.

```
var cos = new COS({
// Required parameters
getAuthorization: function (options, callback) {
// Server-side example: https://github.com/tencentyun/qcloud-cos-sts-sdk/edit/master/scope.md
wx.request({
url: 'https://example.com/server/sts-scope.php',
data: JSON.stringify(options.Scope),
success: function (result) {
var data = result.data;
var credentials = data.credentials;
callback({
TmpSecretId: credentials.tmpSecretId,
TmpSecretKey: credentials.tmpSecretKey,
XCosSecurityToken: credentials.sessionToken,
ExpiredTime: data.expiredTime,
ScopeLimit: true, // You need to set the fine-grained permission control to true to make sure that the key will only
be reused for the same request
});
}
});
}
});
```

- 方法3(推奨しない)：要求ごとに先頭はgetAuthizationで署名を取得する必要があり，バックエンドは固定または仮鍵を用いて署名を計算し，先頭に戻す必要がある.この方法では，複数のアップロードの権限を制御することが困難となるため，この方法を用いることは推奨されない.

```
var cos = new COS({
// Required parameters
getAuthorization: function (options, callback) {
// The server obtains a signature. For more information, see the COS SDK for the corresponding programming language:
```

```
https://cloud.tencent.com/document/product/436/6474
// Note: This method involves security risks. The backend needs to strictly control the permissions through method an
d pathname, such as prohibiting `PUT` /
wx.request({
url: 'https://example.com/server/auth.php',
data: JSON.stringify(options.Scope),
success: function (result) {
var data = result.data;
callback({
Authorization: data.authorization,
// XCosSecurityToken: data.sessionToken, // If a temporary key is used, sessionToken needs to be passed to XCosSecuri
tyToken
});
}
});
},
});
```

- 方法4(推奨しない)：フロントエンドは固定鍵を用いて署名を計算する.この方法をフロントエンドのデバッグに用いることができる.この方法を用いれば，鍵の漏洩を避けるようにしてください.

```
var cos = new COS({
SecretId: 'COS_SECRETID',
SecretKey: 'COS_SECRETKEY',
});
```

構造関数パラメータ記述

パラメータ名	説明する.	タイプ	ひっくるめ書き
秘書ID	ユーザー秘書ID	串	いえ。
分泌キー	フロントエンドのデバッグにはuserkeyのみを用いて鍵の漏洩を回避することを提案している.	串	いえ。
CopyChun並列制限	個々のファイルの合併部品のアップロード回数；デフォルト値：3	番号	いえ。
COPYCHUNKSize	複数のアップロードに失敗した場合のリトライ回数.デフォルト値：3(リクエストは初期要求を含む合計4回発行される)	番号	いえ。
CopySliceSize	多部分アップロードにおける部品サイズ；単位：バイト.デフォルト値：1，048，576(1 MB)	番号	いえ。
協議する.	要求を出すためのプロトコル；実効値： `https:``http:` .デフォルトでは， `http:` 現在のページで使用される `http:` そうでなければ `https:` 使われることになります	串	いえ。
サービスドメイン	GetServiceメソッドが要求するドメイン名. 例： `cos.ap-beijing.myqcloud.com`	串	いえ。
領域	ストレージバレルを呼び出すためのドメイン名は、テンプレートであってもよい。 例： `"{Bucket}.cos.{Region}.myqcloud.com`	串	いえ。

パラメータ名	説明する.	タイプ	ひっくるめ書き
ForcePathStyle	送信要求には接尾辞を強制的に使用する。接尾辞格納バレルはドメイン名の後ろの経路名に格納され、ストレージバケットは署名経路名に追加されて計算される。デフォルト値： `false`	ブル	いえ。
UploadCheckContentMd5	ファイルのアップロードを行うためにContent—MD 5を強制的に検証し、これはファイル要求本文のMD 5チェックサムを計算し、タイトルのContent—MD 5フィールドに渡す。デフォルト値： `false`	ブル	いえ。
授権を得る	署名を取得するためのコールバック方法.uneIdやSysteKeyがなければ，このパラメータが必要となる.	機能.	いえ。

取得承認コールバック関数記述(使用方法1)

```
getAuthorization: function(options, callback) { ... }
```

許可のコールバックパラメータを取得して説明する：

パラメータ名	説明する.	タイプ
選択肢	一時鍵取得に必要なパラメータオブジェクト	機能.
-桶だ	フォーマットのストレージバレル名 `BucketName-APPID`	串
-エリアだ	枚挙値については、ご参照くださいドメイン·アクセスドメイン	串
回撥	一時鍵を取得した後のコールバック方法。	機能.

一時鍵を取得すると，オブジェクトを返す.返すオブジェクトの属性は以下のとおりである.

属性名	パラメータ記述	タイプ	ひっくるめ書き
TmpsecId	取得した一時鍵のtmpsecId	串	はい。
プロンプトキー	取得した一時鍵のtmpuneKey	串	いえ。
XCosSecurityToken	取得された一時鍵のセッションフラグは、この一時鍵に対応する `x-cos-security-token` タイトル中のフィールド	串	いえ。
えんたい時間	一時鍵の期限切れ時間，すなわちタイムアウト時間が取得されている.	串	いえ。

getAuthizationコールバック関数記述(使用方法2)

```
getAuthorization: function(options, callback) { ... }
```

承認関数コール・パラメータ記述を取得する：

パラメータ名	説明する.	タイプ	ひっくるめ書き
選択肢	署名取得に必要なパラメータ対象	相手.	いえ。
-方法だ	当面の要求の方法	相手.	いえ。

パラメータ名	説明する.	タイプ	ひっくるめ書き
-パス名だ	署名計算のための要求経路	串	いえ。
-キー	オブジェクトキー(オブジェクト名)は，バケット中のオブジェクトのユニークIDである.より多くの情報については，参照されたい.対象概要>オブジェクトキー	串	いえ。
-了解	現在要求されている問合せパラメータオブジェクトは，フォームが{key：‘val’}である.	相手.	いえ。
-タイトルだ	現在要求されているHeaderパラメータオブジェクトは，フォーマットが{key：‘val’}である.	相手.	いえ。
回撥	一時鍵を取得した後のコールバック	機能.	いえ。

getAuthization計算が完了した後、1つの署名文字列または1つのオブジェクトを返す：
署名文字列が返される場合、この文字列は、認証ヘッダにおいて、要求によって使用されるべき証拠フィールド“許可”である。
オブジェクトを返すと，そのオブジェクトの属性は以下のようになる.

属性名	パラメータ記述	タイプ	ひっくるめ書き
授権する.	永久または一時鍵計算の授権を使用する。	串	はい。
XCosSecurityToken	取得された一時鍵のセッションフラグは、この一時鍵に対応する `x-cos-security-token` タイトル中のフィールド	串	いえ。

身元証明の証拠を得る

インスタンス化の過程で異なるパラメータが入力されることにより、3つの方法でインスタンスの認証証拠を得ることができる：

1. 事例化では，事務局IDと事務局鍵が渡され，署名が必要となるたびに，インスタンスは内部で計算される.
2. インスタンス化では，着信getAuthizationコールバックは，署名が必要になるたびに署名を計算し，そのコールバックでインスタンスに返す.
3. インスタンス化の間，着信getSTSコールバックは，一時鍵が必要となるたびに，そのコールバックによってインスタンスに返され，要求のたびにインスタンス内で署名計算が行われる.

以下は一般的なAPIの例である。デモンストレーション.

たるをつくる

```
cos.putBucket({
Bucket: 'examplebucket-1250000000',
Region: 'ap-beijing',
Key: filename,
}, function (err, data) {
console.log(err || data);
});
```

WeChatミニプログラムでバレルを作成する必要がありますが、バレル名が不明であれば、バケット名を白列ドメイン名に配置することはできません。逆に、接尾辞コールを使用することができます。詳細については、ご参照くださいありふれた問題.

クエリーバケツリスト

```
cos.getService(function (err, data) {
console.log(data && data.Buckets);
});
```

アップロード相手

WeChatミニプログラムアップロードAPI `wx.uploadFile` ただ支持する. `POST` お願いします。SDKアップロードファイルをご利用の場合は、ご利用ください `postObject` WeChatミニプログラムでファイルアップロードAPIを使うだけであれば、SDKを引用しないことをお勧めします。デモンストレーション.

```
// First, select the file to get the temporary path
wx.chooseImage({
count: 1, // Default value: 9
sizeType: ['original'], // You can specify whether to use the original or compressed image. The original is used by d
efault
sourceType: ['album', 'camera'], // You can specify whether the source is album or camera. Both can be the source by
default
success: function (res) {
var filePath = res.tempFiles[0].path;
var filename = filePath.substr(filePath.lastIndexOf('/') + 1);
cos.postObject({
Bucket: 'examplebucket-1250000000',
Region: 'ap-beijing',
Key: 'destination path/' + filename,
FilePath: filePath,
onProgress: function (info) {
console.log(JSON.stringify(info));
}
}, function (err, data) {
console.log(err || data);
});
}
});
```

照会先リスト

```
cos.getBucket({
Bucket: 'examplebucket-1250000000',
Region: 'ap-beijing',
Prefix: 'exampledir/', // Pass in the prefix of the files to be listed
}, function (err, data) {
console.log(err || data.Contents);
});
```

ダウンロード対象

このAPIは，オブジェクトの内容を読み取るためのものである.ブラウザを起動してファイルをダウンロードする必要があれば，cos.getObjectUrlでURLを取得し，ブラウザでダウンロードを開始することができる.あらかじめ署名したURL.

```
cos.getObject({
Bucket: 'examplebucket-1250000000',
Region: 'ap-beijing',
Key: 'exampleobject.txt',
}, function (err, data) {
console.log(err || data.Body);
});
```

対象を削除する

```
cos.deleteObject({
Bucket: 'examplebucket-1250000000',
Region: 'ap-beijing',
Key: 'picture.jpg',
}, function (err, data) {
console.log(err || data);
});
```