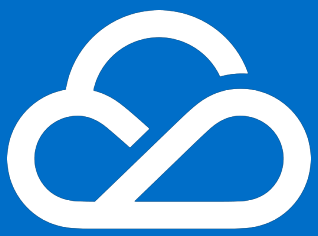
腾讯云对象存储V3

SDK 文档

产品文档



腾讯云

【版权声明】

【商标声明】

【服务声明】

本文档意在向客户介绍腾讯云全部或部分产品、服务的当时的整体概况，部分产品、服务的内容可能有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或模式的承诺或保证。

文档目录

SDK 概览

SDK 概览

除了直接使用 API 接口外，COS 提供了丰富多样的 SDK 供开发者使用。

SDK	接入文档
PHP SDK	PHP SDK 接入说明
Python SDK	Python SDK 接入说明
Node.js SDK	Node.js SDK 接入说明
Java SDK	Java SDK 接入说明
JavaScript SDK	JavaScript SDK 接入说明
C++ SDK	C++ SDK 接入说明
C sharp SDK	C sharp SDK 接入说明
Android SDK	Android SDK 接入说明
iOS SDK	iOS SDK 接入说明

服务端 SDK

PHP SDK

开发准备

SDK 获取

GitHub 项目地址：https://github.com/tencentyun/cos-php-sdk

Composer 项目名：tencentyun/cos-php-sdk

开发环境

1. 依赖环境：PHP 5.3.0 版本及以上
2. 从控制台获取APP ID、SecretID、SecretKey。

SDK 配置

下载 SDK 后，使用 composer 安装，直接执行下述命令：

```
php composer.phar require tencentyun/cos-php-sdk
```

在使用 SDK 时，加载 include.php 即可。

```
require('./include.php');
use Qcloud_cos\Auth;
use Qcloud_cos\Cosapi;
```

若需要支持 HTTPS ，修改 conf.php 中的 API_COSAPI_END_POINT 的值为如下：

```
const API_COSAPI_END_POINT = 'https://web.file.myqcloud.com/files/v1/';
```

生成签名

多次有效签名

方法原型

public static function appSign($expired, $bucketName);

参数说明

参数名	类型	必填	参数描述
expired	long	否	过期时间，Unix时间戳
bucketName	String	是	bucket 名称

返回结果说明

base64编码的字符串

示例

```
$expired = time() + 60;
$bucketName = 'testbucket';
$sign = Auth::appSign($expired, $bucketName);
```

单次有效签名

方法原型

public static function appSign_once($path, $bucketName);

参数说明

参数名	类型	必填	参数描述
path	String	是	文件路径，以斜杠开头，例

参数名	类型	必填	参数描述
			如 /filepath/filename，为文件在此 bucketname 下的全路径
bucketName	String	是	bucket 名称

返回结果说明

base64编码的字符串

示例

```
$bucketName = 'testbucket';
$path = "/myFloder/myFile.rar";
$sign = Auth::appSign_once($path, $bucketName);
```

更多签名相关的详细说明，参考签名算法

目录操作

创建目录

接口说明：用于目录的创建，可通过此接口在指定 bucket 下创建目录。

方法原型

```
public static function createFolder($bucketName, $path, $bizAttr = null);
```

参数说明

参数名	类型	必填	参数描述
bucketName	String	是	bucket 名称
path	String	是	目录全路径
bizAttr	String	否	目录属性信息，业务自行维

参数名	类型	必填	参数描述
			护

返回结果说明

参数名	类型	必带	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息
data	Array	否	返回数据，请参考《Restful API 创建目录》

示例

```
$bizAttr = "attr_folder";
$result = Cosapi::createFolder($bu cketName, $path,$bizAttr)
```

目录更新

接口说明：用于目录业务自定义属性的更新，调用者可以通过此接口更新业务的自定义属性字段。

方法原型

```
public static function updateFolder($bucketName, $path, $bizAttr = null);
```

参数说明

参数名	类型	必填	参数描述
bucketName	String	是	bucket 名称
path	String	是	目录路径
bizAttr	String	否	目录属性信息

返回结果说明

参数名	类型	必带	参数描述

参数名	类型	必带	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息

示例

```
$bizAttr = "folder new attribute";
$result = Cosapi::updateFolder($bucketName, $path, $bizAttr)
```

目录查询

接口说明：用于目录属性的查询，调用者可以通过此接口查询目录的属性。

方法原型

```
public static function statFolder($bucketName, $path);
```

参数说明

参数名	类型	必填	参数描述
bucketName	String	是	bucket 名称
path	String	是	目录路径

返回结果说明

参数名	类型	必带	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息
data	Array	否	目录 属性数据 ，请参考《Restful API 目录查询》

示例

```
$result= Cosapi::statFolder($bucketName, $path);
```

删除目录

接口说明：用于目录的删除，调用者可以通过此接口删除空目录，如果目录中存在有效文件或目录，将不能删除。

方法原型

```
public static function delFolder($bucketName, $path);
```

参数说明

参数名	类型	必填	参数描述
bucketName	String	是	bucket 名称
path	String	是	目录全路径

返回结果说明

参数名	类型	必带	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息

示例

```
$result = Cosapi::delFolder($bucketName, $path);
```

列举目录中的文件和目录

接口说明：用于列举目录下文件和目录，可以通过此接口查询目录下的文件和目录属性。

方法原型

```
public static function listFolder($bucketName, $path, $num = 20, $pattern = 'eListBoth', $order = 0,
```

$context = null);

参数说明

参数名	类型	必填	参数描述
bucketName	String	是	bucket名称
path	String	是	目录的全路径
num	int	否	要查询的目录/文件数量
context	String	否	透传字段，查看第一页，则传空字符串。若需要翻页，需要将前一页返回值中的context透传到参数中。order用于指定翻页顺序。若order填0，则从当前页正序/往下翻页；若order填1，则从当前页倒序/往上翻页
order	int	否	默认正序(=0), 填1为反序
pattern	String	否	eListBoth：列举文件和目录；eListDirOnly：仅列举目录；eListFileOnly：仅列举文件

返回结果说明

参数名	类型	必带	参数描述
code	Int	是	API 错误码，成功时为0
message	String	是	错误信息
data	Array	是	返回数据，请参考《Restful API 目录列表》

示例

$result = Cosapi::listFolder($bucketName, $path, 20, 'eListBoth',0);

列举目录下指定前缀文件&目录

接口说明：用于列举目录下指定前缀的文件和目录，可以通过此接口查询目录下的指定前缀的文件和目录信息。

方法原型

public static function prefixSearch($bucketName, $prefix, $num = 20, $pattern = 'eListBoth', $order = 0, $context = null);

参数说明

参数名	类型	必填	参数描述
bucketName	String	是	bucket名称，bucket创建参见创建Bucket
prefix	String	是	列出含此前缀的所有文件(带全路径)
num	int	否	要查询的目录/文件数量
context	String	否	透传字段，查看第一页，则传空字符串。若需要翻页，需要将前一页返回值中的context透传到参数中。order用于指定翻页顺序。若order填0，则从当前页正序/往下翻页；若order填1，则从当前页倒序/往上翻页
order	int	否	默认正序(=0), 填1为反序
pattern	String	否	eListBoth：列举文件和目录；eListDirOnly：仅列举目录；eListFileOnly：仅列举文件

返回结果说明

参数名	类型	必带	参数描述

参数名	类型	必带	参数描述
code	Int	是	错误码，成功时为0
message	String	是	API 错误信息
data	Array	是	返回数据，请参考《Restful API目录列表》

示例

```
$prefix= "/myFolder/2015-";
$result = Cosapi::prefixSearch($bucketName, $prefix, 20, 'eListBoth',0);
```

文件操作

文件上传

接口说明：文件上传的统一接口，对于大于20M的文件，内部会通过多次分片的方式进行文件上传。

方法原型

```
public static function upload($bucketName, $srcPath, $dstPath,
  $bizAttr = null, $slicesize = null, $insertOnly = null);
```

参数说明

参数名	类型	必填	参数描述
bucketName	String	是	bucket名称，bucket创建参见创建Bucket
srcPath	String	是	本地要上传文件的全路径
dstPath	String	是	文件在COS服务端的全路径，不包括/appid/bucketname
bizAttr	String	否	文件属性，业务端维护
slicesize	int	否	文件分片大小，当文件大于

参数名	类型	必填	参数描述
			20M时，SDK内部会通过多次分片的方式进行上传，默认分片大小为1M，支持的最大分片大小为3M
insertOnly	int	否	同名文件是否进行覆盖。0：覆盖；1：不覆盖

返回结果说明

参数名	类型	必带	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息
data	Array	是	返回数据，请参考《Restful API 创建文件》

示例

```
$dstPath = "/myFolder/test.mp4";
$bizAttr = "";
$insertOnly = 0;
$sliceSize = 3 * 1024 * 1024;
$result = Cosapi::upload($bucketName,$srcPath,$dstPath ,"biz_attr");
```

文件属性更新

接口说明：用于目录业务自定义属性的更新，可以通过此接口更新业务的自定义属性字段。

方法原型

```
public static function update($bucketName, $path,
  $bizAttr = null, $authority=null,$customer_headers_array=null);
```

参数说明

参数名	类型	必填	参数描述
bucketName	String	是	bucket 名称
path	String	是	文件在文件服务端的全路径，不包括/appid/bucketname
bizAttr	String	否	待更新的文件属性信息
authority	String	否	eInvalid(继承Bucket的读写权限)；eWRPrivate(私有读写)；eWPrivateRPublic(公有读私有写)
customer_headers_array	String	否	用户自定义头域。可携带参数名分别为：'Cache-Control'、'Content-Type'、'Content-Disposition'、'Content-Language'、以及以'x-cos-meta-'为前缀的参数名称

返回结果说明

参数名	类型	必带	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息

示例

```
$bizAttr = "";
$authority = "eWPrivateRPublic";
$customer_headers_array = array(
  'Cache-Control' => "no",
  'Content-Language' => "ch",
);
$result = Cosapi::update($bucketName, $dstPath, $bizAttr,$authority, $customer_headers_array);
```

文件查询

接口说明：用于文件的查询，调用者可以通过此接口查询文件的各项属性信息。

方法原型

public static function stat($bucketName, $path);

参数说明

参数名	类型	必填	参数描述
bucketName	String	是	bucket 名称
path	String	是	文件在文件服务端的全路径

返回结果说明

参数名	类型	必带	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息
data	Array	是	文件 属性数据 ，请参考《Restful API 文件查询》

示例

$result = Cosapi::stat($bucketName, $path);

文件删除

接口说明：用于文件的删除，调用者可以通过此接口删除已经上传的文件。

方法原型

public static function delFile($bucketName, $path);

参数说明

参数名	类型	必填	参数描述
bucketName	String	是	bucket 名称
path	String	是	文件的全路径

返回结果说明

参数名	类型	必带	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息

示例

```
$result = Cosapi::delFile($bucketName, $path);
```

Python SDK

Python SDK

开发准备

SDK获取

对象存储服务 Python SDK下载地址：github项目

开发环境

python 2.7

获取python版本的方法:

- Linux Shell

$ python -V
Python 2.7.11

- Windows cmd

D:\>python -V
Python 2.7.11

如果提示不是内部或外部命令，请现在windows环境变量PATH里添加上python的绝对路径

SDK配置

- pip安装

pipinstall qcloud_cos

- 源码安装

github上下载SDK, 解压后如下执行(如果提示permission deny, 需要有管理员权限)

python setup.py install

卸载SDK

pip uninstall qcloud_cos

文件操作

上传文件

方法原型

def upload_file(self, request)

参数说明

参数名	类型	默认值	参数描述
request	UploadFileRequest	无	上传文件类型请求

request成员	类型	默认值	设置方法	描述
bucket_name	unicode	无	构造函数或set方法	bucket名称
cos_path	unicode	无	构造函数或set方法	cos路径, 必须从bucket下的根/开始，文件路径不能以/结尾, 例如 /mytest/demo.txt

request成员	类型	默认值	设置方法	描述
local_path	unicode	无	构造函数或set方法	要上传的本地文件的绝对路径
biz_attr	unicode	空	构造函数或set方法	文件的备注，主要用于对该文件用途的描述
insert_only	int (枚举)	1	构造函数或set方法	是否直插入不覆盖已存在的文件, 1表示只直插入不覆盖, 当文件存在返回错误 0 表示允许覆盖

返回结果说明

返回值类型	返回值描述
code	code为0表示成功
mess	message为SUCCESS或者失败原因
data	data中包含相关的属性

示例

```
request = UploadFileRequest(bucket, u'/sample_file.txt', u'local_file_1.txt')
  upload_file_ret = cos_client.upload_file(request)
```

获取文件属性

方法原型

```
def stat_file(self, request)
```

参数说明

参数名	参数类型	默认值	参数描述
request	StatFileRequest	无	获取文件属性请求

request成员	类型	默认值	设置方法	描述

request成员	类型	默认值	设置方法	描述
bucket_name	unicode	无	构造函数或set方法	bucket名称
cos_path	unicode	无	构造函数或set方法	cos路径, 必须从bucket下的根/开始，文件路径不能以/结尾, 例如 /mytest/demo.txt

返回结果说明

返回值类型	返回值描述
dict	{'code':\$code, 'message':$mess, 'data':\$data}, code为0表示成功, message为SUCCESS或者失败原因, data中包含相关的属性, 详情请参见返回值模块

示例

```
request = StatFileRequest(bucket, u'/sample_file.txt')
  stat_file_ret = cos_client.stat_file(request)
```

更新文件属性

方法原型

```
def update_file(self, request)
```

参数说明

参数名	参数类型	默认值	参数描述
request	UpdateFileRequest	无	更新文件属性请求

request成员	类型	默认值	设置方法	描述
bucket_name	unicode	无	构造函数或set方法	bucket名称
cos_path	unicode	无	构造函数或set方法	cos路径, 必须从bucket下的根/开始，文件路径不能以/结尾,

request成员	类型	默认值	设置方法	描述
				例如 /mytest/demo.txt
biz_attr	unicode	无	set方法	文件的备注，主要用于对改文件用途的描述
authority	unicode (枚举)	无	set方法	文件权限，默认是继承bucket的权限合法取值: eInvalid(继承bucket), eWRPrivate(私有读写), eWPrivateRPublic(私有写, 公有读)
cache_control	unicode	无	set方法	参见HTTP的Cache-Control
content_type	unicode	无	set方法	参见HTTP的Content-Type
content_language	unicode	无	set方法	参见HTTP的Content-Language
content_disposition	unicode	无	set方法	参见HTTP的Content-Disposition
x-cos-meta-	unicode	无	set方法	自定义HTTP 头，参数必须以x-cos-meta-开头，值由用户定义，可设置多个

tips: 更新属性可以选择其中的某几个，对于HTTP头部cache_control，content_type, content_disposition和x-cos-meta-,
如果本次只更新其中的某几个，其他的都会被抹掉，即这4个属性是整体更新。

返回结果说明

返回值类型	返回值描述
dict	{'code':\$code, 'message':$mess}, code为0表示成功, message为SUCCESS或者失败原因,

返回值类型	返回值描述
	详情请参见返回值模块

示例

```
request = UpdateFileRequest(bucket, u'/sample_file.txt')

  request.set_biz_attr(u'这是个demo文件') # 设置文件biz_attr属性
  request.set_authority(u'eWRPrivate') # 设置文件的权限
  request.set_cache_control(u'cache_xxx') # 设置Cache-Control
  request.set_content_type(u'application/text') # 设置Content-Type
  request.set_content_disposition(u'ccccxxx.txt') # 设置Content-Disposition
  request.set_content_language(u'english') # 设置Content-Language
  request.set_x_cos_meta(u'x-cos-meta-xxx', u'xxx') # 设置自定义的x-cos-meta-属性
  request.set_x_cos_meta(u'x-cos-meta-yyy', u'yyy') # 设置自定义的x-cos-meta-属性

  update_file_ret = cos_client.update_file(request)
```

移动文件(重命名文件)

方法原型

```
def move_file(self, request)
```

参数说明

参数名	参数类型	默认值	参数描述
request	MoveFileRequest	无	移动文件请求

request成员	类型	默认值	设置方法	描述
bucket_name	unicode	无	构造函数或set方法	bucket名称
cos_path	unicode	无	构造函数或set方法	cos路径, 必须从bucket下的根/开始，文件路径不能以/结尾, 例如

request成员	类型	默认值	设置方法	描述
				/mytest/demo.txt
to_over_write	int	0	构造函数或set方法	是否覆盖, 0(默认): 不覆盖, 1: 覆盖

返回结果说明

返回值类型	返回值描述
dict	{'code':\$code, 'message':$mess}, code为0表示成功, message为SUCCESS或者失败原因, 详情请参见返回值模块

示例

```
request = MoveFileRequest(bucket, u'/sample_file.txt', u'/sample_file_move.txt')
  stat_file_ret = cos_client.move_file(request)
```

删除文件

方法原型

```
def del_file(self, request)
```

参数说明

参数名	参数类型	默认值	参数描述
request	DelFileRequest	无	删除文件请求

request成员	类型	默认值	设置方法	描述
bucket_name	unicode	无	构造函数或set方法	bucket名称
cos_path	unicode	无	构造函数或set方法	cos路径, 必须从bucket下的根/开始，文件路径不能以/结尾, 例如 /mytest/demo.txt

返回结果说明

返回值类型	返回值描述
dict	{'code':\$code, 'message':$mess}, code为0表示成功, message为SUCCESS或者失败原因, 详情请参见返回值模块

示例

```
request = DelFileRequest(bucket, u'/sample_file_move.txt')
  del_ret = cos_client.del_file(request)
```

目录操作

创建目录

方法原型

```
def create_folder(self, request)
```

参数说明

参数名	参数类型	默认值	参数描述
request	CreateFolderRequest	无	创建目录请求

request成员	类型	默认值	设置方法	描述
bucket_name	unicode	无	构造函数或set方法	bucket名称
cos_path	unicode	无	构造函数或set方法	cos路径, 必须从bucket下的根/开始，目录路径必须以/结尾, 例如 /mytest/dir/
biz_attr	unicode	空	set方法	目录的备注，主要用于对目录用途的描述

返回结果说明

返回值类型	返回值描述
dict	{'code':\$code, 'message':$mess}, code为0表示成功, message为SUCCESS或者失败原因, 详情请参见返回值模块

示例

```
request = CreateFolderRequest(bucket, u'/sample_folder/')
  create_folder_ret = cos_client.create_folder(request)
```

获取目录属性

方法原型

```
def stat_folder(self, request)
```

参数说明

参数名	参数类型	默认值	参数描述
request	StatFolderRequest	无	获取目录属性请求

request成员	类型	默认值	设置方法	描述
bucket_name	unicode	无	构造函数或set方法	bucket名称
cos_path	unicode	无	构造函数或set方法	cos路径, 必须从bucket下的根/开始，目录路径必须以/结尾, 例如 /mytest/dir/

返回结果说明

返回值类型	返回值描述
dict	{'code':\$code, 'message':$mess, 'data':\$data}, code为0表示成功, message为SUCCESS或者失败原因, data中包含相关的属性, 详情请参见返回值模块

示例

```
request = StatFolderRequest(bucket, u'/sample_folder/')
  stat_folder_ret = cos_client.stat_folder(request)
```

更新目录属性

方法原型

```
def update_folder(self, request)
```

参数说明

参数名	参数类型	默认值	参数描述
request	UpdateFolderRequest	无	更新目录属性请求

request成员	类型	默认值	设置方法	描述
bucket_name	unicode	无	构造函数或set方法	bucket名称
cos_path	unicode	无	构造函数或set方法	cos路径, 必须从bucket下的根/开始，目录路径必须以/结尾, 例如 /mytest/dir/
biz_attr	unicode	空	set方法	目录的备注，主要用于对目录用途的描述

返回结果说明

返回值类型	返回值描述
dict	{'code':\$code, 'message':$mess}, code为0表示成功, message为SUCCESS或者失败原因, 详情请参见返回值模块

示例

```
request = UpdateFolderRequest(bucket, u'/sample_folder/')
  request.set_biz_attr(u'这是一个测试目录')
  update_folder_ret = cos_client.update_folder(request)
```

获取目录列表

方法原型

def list_folder(self, request)

参数说明

参数名	参数类型	默认值	参数描述
request	ListFolderRequest	无	获取目录成员请求

request成员	类型	默认值	设置方法	描述
bucket_name	unicode	无	构造函数或set方法	bucket名称
cos_path	unicode	无	构造函数或set方法	cos路径, 必须从bucket下的根/开始，目录路径必须以/结尾, 例如 /mytest/dir/
num	int	199	构造函数或set方法	获取列表成员的数量，最大为199
pattern	unicode	eListBoth	构造函数或set方法	获取列表成员类型, 合法取值eListBoth(获取文件和目录), eListDirOnly(只获取目录), eListFileOnly(只获取文件)
prefix	unicode	空	构造函数或set方法	搜索成员的前缀, 例如prefix为test表示只搜索以test开头的文件或目录
context	unicode	空	构造函数或set方法	搜索上下文, 由上一次list的结果返回，作为这一次搜索的起点，用于循环获取一个目录下的所有成员
order	int	0	构造函数或set方法	搜索顺序, 0: 正序, 1:

request成员	类型	默认值	设置方法	描述
				逆序

返回结果说明

返回值类型	返回值描述
dict	{'code':\$code, 'message':$mess, 'data':\$data}, code为0表示成功, message为SUCCESS或者失败原因, data中包含成员列表, 详情请参见返回值模块

示例

```
request = ListFolderRequest(bucket, u'/sample_folder/')
  list_folder_ret = cos_client.list_folder(request)
```

删除目录

方法原型

```
def del_folder(self, request)
```

参数说明

参数名	参数类型	默认值	参数描述
request	DelFolderRequest	无	删除目录请求

request成员	类型	默认值	设置方法	描述
bucket_name	unicode	无	构造函数或set方法	bucket名称
cos_path	unicode	无	构造函数或set方法	cos路径, 必须从bucket下的根/开始，目录路径必须以/结尾, 例如 /mytest/dir/

返回结果说明

返回值类型	返回值描述
dict	{'code':\$code, 'message':$mess},

返回值类型	返回值描述
	code为0表示成功, message为SUCCESS或者失败原因, 详情请参见返回值模块

示例

```
request = DelFolderRequest(bucket, u'/sample_folder/')
  delete_folder_ret = cos_client.del_folder(request)
```

签名管理

签名模块提供了生成多次签名、单次签名和下载签名的接口，其中多次签名和单次签名在文件和目录操作的api内部使用，用户不用关心，下载签名用于方便用户生成下载私有bucket的文件签名。

多次签名

方法原型

```
def sign_more(self, bucket, cos_path, expired)
```

使用场景

上传文件, 重命名文件, 创建目录, 获取文件目录属性, 拉取目录列表

参数说明

参数名	参数类型	默认值	参数描述
bucket	unicode	无	bucket名称
cos_path	unicode	无	要签名的cos路径
expired	int	无	签名过期时间, UNIX时间戳

返回结果说明

base64编码的字符串

示例

```
cred = CredInfo(100000, u'xxxxxxx', u'xxxxxxxx') # appid, secret_id, secret_key
auth_obj = Auth(cred)
sign_str = auth_obj.sign_more(u'mybucket', u'/pic/1.jpg', int(time.time()) + 600)
```

单次签名

方法原型

```
def sign_once(self, bucket, cos_path)
```

使用场景

删除和更新文件目录

参数说明

参数名	参数类型	默认值	参数描述
bucket	unicode	无	bucket名称
cos_path	unicode	无	要签名的cos路径

返回结果说明

base64编码的字符串

示例

```
cred = CredInfo(100000, u'xxxxxxx', u'xxxxxxxx') # appid, secret_id, secret_key
auth_obj = Auth(cred)
sign_str = auth_obj.sign_once(u'mybucket', u'/pic/1.jpg')
```

下载签名

方法原型

def sign_download(self, bucket, cos_path, expired)

使用场景

生成文件的下载签名, 用于下载私有bucket的文件

参数说明

参数名	参数类型	默认值	参数描述
bucket	unicode	无	bucket名称
cos_path	unicode	无	要签名的cos路径
expired	int	无	签名过期时间, UNIX时间戳

返回结果说明

base64编码的字符串

示例

```
cred = CredInfo(100000, u'xxxxxxx', u'xxxxxxxx') # appid, secret_id, secret_key
auth_obj = Auth(cred)
sign_str = auth_obj.sign_download(u'mybucket', u'/pic/1.jpg', int(time.time()) + 600)
```

操作返回值说明

code	含义
0	操作成功
-1	输入参数错误, 例如输入的本地文件路径不存在, cos文件路径不符合规范
-2	网络错误, 如404等
-3	连接cos时发生异常，如连接超时
-71	操作频率过快，触发cos的攻击

Java SDK

开发准备

SDK获取

对象存储服务 Java SDK下载地址：github项目

开发环境

JDK 1.7

SDK配置

- maven安装

pom.xml 添加依赖

```
<dependency>
  <groupId>com.qcloud</groupId>
  <artifactId>cos_api</artifactId>
  <version>3.3</version>
</dependency>
```

- 源码安装

从github下载源码

卸载SDK

删除pom依赖或源码

文件操作

上传文件

方法原型

String uploadFile(UploadFileRequest request);

参数说明

参数名	类型	默认值	参数描述
request	UploadFileRequest	无	上传文件类型请求

request成员	类型	默认值	设置方法	描述
bucketName	String	无	构造函数或set方法	bucket名称
cosPath	String	无	构造函数或set方法	cos路径, 必须从bucket下的根/开始，文件路径不能以/结尾, 例如 /mytest/demo.txt
localPath	String	无	构造函数或set方法	要上传的本地文件的绝对路径
bizAttr	String	空	构造函数或set方法	文件的备注，主要用于对该文件用途的描述
insertOnly	InsertOnly (枚举)	NO_OVER_WRITE (不覆盖)	set方法	是否直插入不覆盖已存在的文件, NO_OVER_WRITE表示只直插入不覆盖, 当文件存在返回错误 OVER_WRITE 表示允许覆盖

返回结果说明

返回值类型	返回值描述
String	{'code':\$code, 'message':$mess, 'data':\$data}, code为0表示成功, message为SUCCESS或者失败原因,

返回值类型	返回值描述
	data中包含相关的属性, 详情请参见返回值模块

示例

```
UploadFileRequest uploadFileRequest = new UploadFileRequest(bucketName, "/sample_file.txt",
"local_file_1.txt");
String uploadFileRet = cosClient.uploadFile(uploadFileRequest);
```

获取文件属性

方法原型

```
String statFile(StatFileRequest request);
```

参数说明

参数名	参数类型	默认值	参数描述
request	StatFileRequest	无	获取文件属性请求

request成员	类型	默认值	设置方法	描述
bucketName	String	无	构造函数或set方法	bucket名称
cosPath	String	无	构造函数或set方法	cos路径, 必须从bucket下的根/开始，文件路径不能以/结尾, 例如/mytest/demo.txt

返回结果说明

返回值类型	返回值描述
String	{'code':\$code, 'message':$mess, 'data':\$data}, code为0表示成功, message为SUCCESS或者失败原因, data中包含相关的属性, 详情请参见返回值模块

示例

```
StatFileRequest statFileRequest = new StatFileRequest(bucketName, "/sample_file.txt");
String statFileRet = cosClient.statFile(statFileRequest);
```

更新文件属性

方法原型

```
String updateFile(UpdateFileRequest request);
```

参数说明

参数名	参数类型	默认值	参数描述
request	UpdateFileRequest	无	更新文件属性请求

request成员	类型	默认值	设置方法	描述
bucketName	String	无	构造函数或set方法	bucket名称
cosPath	String	无	构造函数或set方法	cos路径, 必须从bucket下的根/开始，文件路径不能以/结尾, 例如 /mytest/demo.txt
bizAttr	String	无	set方法	文件的备注，主要用于对改文件用途的描述
authority	String (枚举)	无	set方法	文件权限，默认是继承bucket的权限合法取值: eInvalid(继承bucket), eWRPrivate(私有读写), eWPrivateRPublic(私有写, 公有读)
cacheControl	String	无	set方法	参见HTTP的Cache-Control
contentType	String	无	set方法	参见HTTP的Content-

request成员	类型	默认值	设置方法	描述
				Type
contentLanguage	String	无	set方法	参见HTTP的Content-Language
contentDisposition	String	无	set方法	参见HTTP的Content-Disposition
x-cos-meta-	String	无	set方法	自定义HTTP 头，参数必须以x-cos-meta-开头，值由用户定义，可设置多个

tips: 更新属性可以选择其中的某几个，对于HTTP头部cache_control，content_type, content_disposition和x-cos-meta-,
如果本次只更新其中的某几个，其他的都会被抹掉，即这4个属性是整体更新。

返回结果说明

返回值类型	返回值描述
String	{'code':\$code, 'message':$mess}, code为0表示成功, message为SUCCESS或者失败原因, 详情请参见返回值模块

示例

```
UpdateFileRequest updateFileRequest = new UpdateFileRequest(bucketName, "/sample_file.txt");

updateFileRequest.setBizAttr("测试目录");
updateFileRequest.setAuthority(FileAuthority.WPRIVATE);
updateFileRequest.setCacheControl("no cache");
updateFileRequest.setContentDisposition("cos_sample.txt");
updateFileRequest.setContentLanguage("english");
updateFileRequest.setContentType("application/json");
updateFileRequest.setXCosMeta("x-cos-meta-xxx", "xxx");
updateFileRequest.setXCosMeta("x-cos-meta-yyy", "yyy");

String updateFileRet = cosClient.updateFile(updateFileRequest);
```

移动文件(重命名文件)

方法原型

String moveFile(MoveFileRequest request);

参数说明

参数名	参数类型	默认值	参数描述
request	MoveFileRequest	无	移动文件请求

request成员	类型	默认值	设置方法	描述
bucketName	String	无	构造函数或set方法	bucket名称
cosPath	String	无	构造函数或set方法	cos路径, 必须从bucket下的根/开始，文件路径不能以/结尾, 例如/mytest/demo.txt
overWrite	OverWrite	OverWrite.NO_OVER_WRITE	构造函数或set方法	是否覆盖, 0(默认): 不覆盖, 1: 覆盖

返回结果说明

返回值类型	返回值描述
String	{'code':\$code, 'message':$mess}, code为0表示成功, message为SUCCESS或者失败原因, 详情请参见返回值模块

示例

```
MoveFileRequest moveFileRequest = new MoveFileRequest(bucketName, "/sample_file.txt",
"/sample_file_move.txt");
String moveFileRet = cosClient.moveFile(moveFileRequest);
```

删除文件

方法原型

String delFile(DelFileRequest request);

参数说明

参数名	参数类型	默认值	参数描述
request	DelFileRequest	无	删除文件请求

request成员	类型	默认值	设置方法	描述
bucketName	String	无	构造函数或set方法	bucket名称
cosPath	String	无	构造函数或set方法	cos路径, 必须从bucket下的根/开始，文件路径不能以/结尾, 例如 /mytest/demo.txt

返回结果说明

返回值类型	返回值描述
String	{'code':\$code, 'message':$mess}, code为0表示成功, message为SUCCESS或者失败原因, 详情请参见返回值模块

示例

```
DelFileRequest delFileRequest = new DelFileRequest(bucketName, "/sample_file_move.txt");
String delFileRet = cosClient.delFile(delFileRequest);
```

目录操作

创建目录

方法原型

String createFolder(CreateFolderRequest request);

参数说明

参数名	参数类型	默认值	参数描述
request	CreateFolderRequest	无	创建目录请求

request成员	类型	默认值	设置方法	描述
bucketName	String	无	构造函数或set方法	bucket名称
cosPath	String	无	构造函数或set方法	cos路径, 必须从bucket下的根/开始，目录路径必须以/结尾, 例如 /mytest/dir/
bizAttr	String	空	set方法	目录的备注，主要用于对目录用途的描述

返回结果说明

返回值类型	返回值描述
String	{'code':\$code, 'message':$mess}, code为0表示成功, message为SUCCESS或者失败原因, 详情请参见返回值模块

示例

```
CreateFolderRequest createFolderRequest = new CreateFolderRequest(bucketName,
"/sample_folder/");
String createFolderRet = cosClient.createFolder(createFolderRequest);
```

获取目录属性

方法原型

String statFolder(StatFolderRequest request);

参数说明

参数名	参数类型	默认值	参数描述
request	StatFolderRequest	无	获取目录属性请求

request成员	类型	默认值	设置方法	描述
bucketName	String	无	构造函数或set方法	bucket名称
cosPath	String	无	构造函数或set方法	cos路径, 必须从bucket下的根/开始，目录路径必须以/结尾, 例如 /mytest/dir/

返回结果说明

返回值类型	返回值描述
String	{'code':\$code, 'message':$mess, 'data':\$data}, code为0表示成功, message为SUCCESS或者失败原因, data中包含相关的属性, 详情请参见返回值模块

示例

```
StatFolderRequest statFolderRequest = new StatFolderRequest(bucketName, "/sample_folder/");
String statFolderRet = cosClient.statFolder(statFolderRequest);
```

更新目录属性

方法原型

```
String updateFolder(UpdateFolderRequest request);
```

参数说明

参数名	参数类型	默认值	参数描述
request	UpdateFolderRequest	无	更新目录属性请求

request成员	类型	默认值	设置方法	描述
bucketName	String	无	构造函数或set方法	bucket名称

request成员	类型	默认值	设置方法	描述
cosPath	String	无	构造函数或set方法	cos路径, 必须从bucket下的根/开始，目录路径必须以/结尾, 例如 /mytest/dir/
bizAttr	String	空	set方法	目录的备注，主要用于对目录用途的描述

返回结果说明

返回值类型	返回值描述
String	{'code':\$code, 'message':$mess}, code为0表示成功, message为SUCCESS或者失败原因, 详情请参见返回值模块

示例

```
UpdateFolderRequest updateFolderRequest = new UpdateFolderRequest(bucketName,
"/sample_folder/");
updateFolderRequest.setBizAttr("这是一个测试目录");
String updateFolderRet = cosClient.updateFolder(updateFolderRequest);
```

获取目录列表

方法原型

```
String listFolder(ListFolderRequest request);
```

参数说明

参数名	参数类型	默认值	参数描述
request	ListFolderRequest	无	获取目录成员请求

request成员	类型	默认值	设置方法	描述
bucketName	String	无	构造函数或set方法	bucket名称
cosPath	String	无	构造函数或set方法	cos路径, 必须从buck

request成员	类型	默认值	设置方法	描述
				et下的根/开始，目录路径必须以/结尾，例如 /mytest/dir/
num	int	199	构造函数或set方法	获取列表成员的数量，最大为199
pattern	ListPattern (枚举)	BOTH	构造函数或set方法	获取列表成员类型，合法取值 BOTH(获取文件和目录), DIR_ONLY(只获取目录), FILE_ONLY(只获取文件)
prefix	String	空	构造函数或set方法	搜索成员的前缀，例如prefix为test表示只搜索以test开头的文件或目录
context	String	空	构造函数或set方法	搜索上下文，由上一次list的结果返回，作为这一次搜索的起点，用于循环获取一个目录下的所有成员
order	ListOrder (枚举)	POSITIVE (正序)	构造函数或set方法	搜索顺序，POSITIVE: 正序，NEGATIVE: 逆序

返回结果说明

返回值类型	返回值描述
String	{'code':\$code, 'message':$mess, 'data':\$data}, code为0表示成功，message为SUCCESS或者失败原因，data中包含成员列表，详情请参见返回值模块

示例

```
ListFolderRequest listFolderRequest = new ListFolderRequest(bucketName, "/sample_folder/");
String listFolderRet = cosClient.listFolder(listFolderRequest);
```

删除目录

方法原型

String delFolder(DelFolderRequest request);

参数说明

参数名	参数类型	默认值	参数描述
request	DelFolderRequest	无	删除目录请求

request成员	类型	默认值	设置方法	描述
bucketName	String	无	构造函数或set方法	bucket名称
cosPath	String	无	构造函数或set方法	cos路径, 必须从bucket下的根/开始，目录路径必须以/结尾, 例如 /mytest/dir/

返回结果说明

返回值类型	返回值描述
String	{'code':\$code, 'message':$mess}, code为0表示成功, message为SUCCESS或者失败原因, 详情请参见返回值模块

示例

```
DelFolderRequest delFolderRequest = new DelFolderRequest(bucketName, "/sample_folder/");
String delFolderRet = cosClient.delFolder(delFolderRequest);
```

签名管理

签名模块提供了生成多次签名、单次签名和下载签名的接口，其中多次签名和单次签名在文件和目录操作的api内部使用，用户不用关心，下载签名用于方便用户生成下载私有bucket的文件签名。

多次签名

方法原型

String getPeriodEffectiveSign(String bucketName, String cosPath, Credentials cred, long expired)

使用场景

上传文件, 重命名文件, 创建目录, 获取文件目录属性, 拉取目录列表

参数说明

参数名	参数类型	默认值	参数描述
bucket	String	无	bucket名称
cos_path	String	无	要签名的cos路径
cred	Credentials	无	用户身份信息, 包括appid, secretId, secretkey
expired	long	无	签名过期时间, UNIX时间戳

返回结果说明

base64编码的字符串

示例

```
Credentials cred = new Credentials(appId, secretId, secretKey);
long expired = System.currentTimeMillis() / 1000 + 600;
String signStr = Sign.getPeriodEffectiveSign(bucketName, "/pic/test.jpg", cred, expired);
```

单次签名

方法原型

String getOneEffectiveSign(String bucketName, String cosPath, Credentials cred)

使用场景

删除和更新文件目录

参数说明

参数名	参数类型	默认值	参数描述
bucket	unicode	无	bucket名称
cos_path	unicode	无	要签名的cos路径
cred	Credentials	无	用户身份信息, 包括appid, secretId, secretkey

返回结果说明

base64编码的字符串

示例

```
Credentials cred = new Credentials(appId, secretId, secretKey);
String signStr = Sign.getOneEffectiveSign(bucketName, "/pic/test.jpg", cred);
```

下载签名

方法原型

String getDownLoadSign(String bucketName, String cosPath, Credentials cred, long expired)

使用场景

生成文件的下载签名, 用于下载私有bucket的文件

参数说明

参数名	参数类型	默认值	参数描述

参数名	参数类型	默认值	参数描述
bucket	unicode	无	bucket名称
cos_path	unicode	无	要签名的cos路径
cred	Credentials	无	用户身份信息, 包括appid, secretId, secretkey
expired	long	无	签名过期时间, UNIX时间戳

返回结果说明

base64编码的字符串

示例

```
Credentials cred = new Credentials(appId, secretId, secretKey);
long expired = System.currentTimeMillis() / 1000 + 600;
String signStr = Sign.getDownLoadSign(bucketName, "/pic/test.jpg", cred, expired);
```

操作返回值说明

code	含义
0	操作成功
-1	输入参数错误, 例如输入的本地文件路径不存在, cos文件路径不符合规范
-2	网络错误, 如404等
-3	连接cos时发生异常，如连接超时
-71	操作频率过快，触发cos的频控

C++ SDK

开发准备

SDK 获取

对象存储服务的C++ SDK的下载地址： https://github.com/tencentyun/cos-cpp-sdk/tree/3.3.0

开发环境

1. 安装openssl的库和头文件 http://www.openssl.org/source/
2. 安装curl的库和头文件 http://curl.haxx.se/download/curl-7.43.0.tar.gz
3. 安装jsoncpp的库和头文件 https://github.com/open-source-parsers/jsoncpp
4. 安装cmake工具 http://www.cmake.org/download/
5. 从控制台获取APP ID、SecretID、SecretKey

注意：

1. sdk中提供了curl和jsoncpp的库以及头文件，以上库编译好后替换掉sdk中相应的库和头文件即可，如果以上库已经安装到系统里，也可删除sdk中相应的库和头文件。
2. curl默认不支持多线程环境，如果项目使用多线程，在编译curl执行 configure 时需指定 --enable-ares 参数来开启异步DNS解析，依赖 c-ares库，如果系统没有，可到http://c-ares.haxx.se/ 下载安装。
3. jsoncpp的1.y.x版本需要c++11的支持，如果编译器不支持，可以换成 0.y.x版本。

SDK 配置

直接下载github上提供的源代码，集成到您的开发环境。

执行下面的命令 ：

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

cmake ..
make

cos_demo.cpp里面有常见API的例子，需要将cos_demo.cpp里的appid、secretId、secretKey、bucket等信息换成自己的信息。

生成的cos_demo就可以直接运行，试用，生成的静态库，名称为：libcosdk.a。

生成的 libcosdk.a 放到你自己的工程里lib路径下， include 目录下的 auth_utility.h cos_api.h curl json openssl 都放到你自己的工程的include路径下。

例如项目里只有一个 sample.cpp ，项目目录和sdk在同级目录， copy libcosdk.a
到项目所在目录那么编译命令为:

g++ -o sample sample.cpp -I ./include/ -L. -L../cos-cpp-sdk/lib/ -lcosdk -lcurl -lcrypto -lssl -lrt -ljsoncpp

若需要 HTTPS 支持，修改 cos_api_defines 中kApiCosapiEndpoint 的值为：

https://web.file.myqcloud.com/files/v1/

生成签名

多次有效签名

方法原型

```
static string AppSignMuti(const uint64_t appId,
  const string &secretId,
  const string &secretKey,
  const uint64_t expired_time,
  const string &bucketName);
```

参数说明

参数名	类型	是否必填	默认值	参数描述
appId	uint64_t	是	无	项目APP ID
secretId	String	是	无	用户 Secret ID
secretKey	String	是	无	用户 SecretKey
expired_time	uint64_t	否	无	过期时间，Unix时间戳
bucketName	String	否	无	bucket名称，bucket创建参见创建Bucket

返回结果说明

返回值：签名字符串

示例

```
uint64_t expired = time(NULL) + 60;
string sign = AuthUtility::AppSignMuti(10000000, "SecretId",
  "SecretKey", expired ,
  "bucketName");
```

单次有效签名

方法原型

```
static string AppSignOnce(const uint64_t appId,
  const string &secretId,
  const string &secretKey,
  const string &path,
  const string &bucketName);
```

参数说明

参数名	类型	必须	默认值	参数描述
appId	uint64_t	是	无	项目 APP ID
secretId	String	是	无	项目 SecretID
secretKey	String	是	无	项目 SecretKey
bucketName	String	否	无	bucket名称，bucket创建参见创建Bucket
path	String	是	无	文件路径，以斜杠开头，例如/filepath/filename，为文件在此bucketname下的全路径

返回结果说明

返回值：签名字符串

示例

```
string path= "/myFloder/myFile.rar";
sign = AuthUtility::AppSignOnce(10000000, "SecretId", "SecretKey", path, bucketName);
```

更多签名相关详细说明，请参考签名算法 。

初始化操作

初始化

接口说明：在使用CosApi进行操作之前，需要首先初始化使用的库和资源。

方法原型

```
int COS_Init();
```

返回结果说明

返回0代表成功，否则代表失败

目录操作

创建目录

接口说明：用于目录的创建，调用者可以通过此接口在指定bucket下创建目录。

方法原型

```
string CreateFolder(const string &bucketName,
  const string &path,
  const CustomOptions& options = CustomOptions());
```

参数说明

参数名	类型	是否必填	默认值	参数描述
bucketName	String	是	无	bucket名称，bucket创建参见创建Bucket
path	String	是	无	需要创建目录的全路径，以"/"开头,以"/"结尾，api会补齐
options	CustomOptions	否	空	用户自定义选项，为一组key-value对

返回结果说明

通过函数返回值返回请求结果的json字符串

参数名	类型	参数描述
code	Int	错误码，成功时为0
message	String	错误信息

参数名	类型	参数描述
data	Array	返回数据
data.ctime	String	目录的创建时间，unix时间戳
data.resource_path	String	目录的资源路径

示例

```
CosApiClientOption client_option("your appid", "your secretId", "your secretKey");
CosApi cos_client(client_option);
string bucketName = "myBucket";
string path = "/myFolder/";
CustomOptions options;
options.AddStringOption("biz_attr", "attr_folder");
string result = cos_client.CreateFolder(bucketName, path, options);
```

目录属性更新

接口说明：用于目录业务自定义属性的更新，可以通过此接口更新业务的自定义属性字段。

方法原型

```
string UpdateFolder(const string &bucketName,
  const string &path,
  const CustomOptions& options);
```

参数说明

参数名	类型	是否必填	默认值	参数描述
bucketName	String	是	无	bucket名称，bucket创建参见创建Bucket
path	String	是	无	需要创建目录的全路径，以"/"开头,以"/"结尾，api会补齐
options	CustomOptions	否	空	用户自定义选项，为

参数名	类型	是否必填	默认值	参数描述
				一组key-value对

Options说明

参数名	类型	是否必填	默认值	参数描述
biz_attr	String	否	""	目录/文件属性，业务端维护

返回结果说明

通过函数返回值返回请求结果的json字符串

参数名	类型	参数描述
code	Int	错误码，成功时为0
message	String	错误信息

示例

```
CosApiClientOption client_option("your appid", "your secretId", "your secretKey");
CosApi cos_client(client_option);
string bucketName = "myBucket";
string path = "/myFolder/";
CustomOptions options;
options.AddStringOption("biz_attr", "attr_folder");
string result = cos_client.UpdateFolder(bucketName, path, options);
```

目录查询

接口说明：用于目录属性的查询，可以通过此接口查询目录的属性。

方法原型

```
string StatFolder(const string &bucketName, const string &path);
```

参数说明

参数名	类型	是否必填	默认值	参数描述
bucketName	String	是	无	bucket名称，bucket创建参见创建Bucket
path	String	是	无	目录的全路径，以"/"开头,以"/"结尾，api会补齐

返回结果说明

通过函数返回值返回请求结果的json字符串

参数名	类型	参数描述
code	Int	错误码，成功时为0
message	String	错误信息
data	Array	目录属性数据
data.biz_attr	String	目录绑定的属性信息，业务自行维护
data.ctime	String	目录的创建时间，unix时间戳
data.mtime	String	目录的修改时间，unix时间戳
data.name	String	目录的名称

示例

```
CosApiClientOption client_option("your appid", "your secretId", "your secretKey");
CosApi cos_client(client_option);
string bucketName = "myBucket";
string path = "/myFolder/";
string result = cos_client.StatFolder(bucketName, path, options);
```

目录删除

接口说明：用于目录的删除，可以通过此接口删除空目录，如果目录中存在有效文件或目录，将不能删除。

方法原型

string DelFolder(const string &bucketName, const string &path);

参数说明

参数名	类型	是否必填	默认值	参数描述
bucketName	String	是	无	bucket名称，bucket创建参见创建Bucket
path	String	是	无	目录的全路径，以"/"开头,以"/"结尾，api会补齐

返回结果说明

通过函数返回值返回请求结果的json字符串

参数名	类型	参数描述
code	Int	错误码，成功时为0
message	String	错误信息

示例

```
CosApiClientOption client_option("your appid", "your secretId", "your secretKey");
CosApi cos_client(client_option);
string bucketName = "myBucket";
string path = "/myFolder/";
string result = cos_client.DelFolder(bucketName, path, options);
```

列举目录下文件&目录

接口说明：用于列举目录下文件和目录，可以通过此接口查询目录下的文件和目录属性。

方法原型

string ListFolder(const string &bucketName,

```
const string &path,
const CustomOptions& options = DefaultListOptions());
```

参数说明

参数名	类型	是否必填	默认值	参数描述
bucketName	String	是	无	bucket名称，bucket创建参见创建Bucket
path	String	是	无	目录的全路径，以"/"开头,以"/"结尾，api会补齐
options	CustomOptions	否	空	用户自定义选项，为一组key-value对

Options说明

参数名	类型	是否必填	默认值	参数描述
num	int	否	20	拉去总数
order	int	否	0	0是正序，1是反序
pattern	String	否	eListBoth	eListBoth,eListDirOnly,eListFileOnly
context	String	否	空字符串	透传字段，查看第一页，则传空字符串。若需要翻页，需要将前一页返回值中的context透传到参数中

返回结果说明

通过函数返回值返回请求结果的json字符串

参数名	类型	是否必然返回	参数描述
code	Int	是	API 错误码，成功时为0
message	String	是	错误信息
data	Array	是	返回数据

参数名	类型	是否必然返回	参数描述
data.has_more	Bool	是	是否有内容可以继续往前/往后翻页
data.context	String	是	透传字段，查看第一页，则传空字符串。若需要翻页，需要将前一页返回值中的context透传到参数中。order用于指定翻页顺序。若order填0，则从当前页正序/往下翻页；若order填1，则从当前页倒序/往上翻页
data.dircount	String	是	子目录数量(总)
data.filecount	String	是	子文件数量(总)
data.infos	Array	是	文件、目录集合，可以为空
data.infos.name	String	是	文件或目录名
data.infos.biz_attr	String	是	目录或文件属性，业务端维护
data.infos.ctime	String	是	目录或文件的创建时间，unix时间戳
data.infos.mtime	String	是	目录或文件的修改时间，unix时间戳
data.infos.filesize	Int	否(当类型为文件时返回)	文件大小
data.infos.filelen	Int	否(当类型为文件时返回)	文件已传输大小(通过与filesize对比可知文件传输进度)
data.infos.sha	String	否(当类型为文件时返回)	文件sha
data.infos.access_url	String	否(当类型为文件时返回)	生成的下载url

示例

```
CosApiClientOption client_option("your appid", "your secretId", "your secretKey");
CosApi cos_client(client_option);
string bucketName = "myBucket";
string path = "/myFolder/";
CustomOptions options;
options.AddUIntOption("num", 20);
```

```
options.AddUIntOption("order", 0);
options.AddStringOption("pattern", "eListBoth");
options.AddStringOption("context", "");
string result = cos_client.ListFolder(bucketName, path, options);
```

列举目录下指定前缀文件&目录

接口说明：用于列举目录下指定前缀的文件和目录，可以通过此接口查询目录下的指定前缀的文件和目录信息。

方法原型

```
string PrefixSearch(const string &bucketName,
  const string &path,
  const CustomOptions& options = DefaultListOptions());
```

参数说明

参数名	类型	是否必填	默认值	参数描述
bucketName	String	是	无	bucket名称，bucket创建参见创建Bucket
prefix	String	是	无	列出含此前缀的所有文件(带全路径)
options	CustomOptions	否	空	用户自定义选项，为一组key-value对

Options说明

参数名	类型	是否必填	默认值	参数描述
num	int	是	20	要查询的目录/文件数量
context	String	是	无	透传字段，查看第一页，则传空字符串。若需要翻页，需要将

参数名	类型	是否必填	默认值	参数描述
				前一页返回值中的context透传到参数中。order用于指定翻页顺序。若order填0，则从当前页正序/往下翻页；若order填1，则从当前页倒序/往上翻页
order	int	是	无	默认正序(=0), 填1为反序
pattern	String	是	无	eListBoth,eListDirOnly,eListFileOnly 默认eListBoth

返回结果说明

通过函数返回值返回请求结果的json字符串

参数名	类型	是否必然返回	参数描述
code	Int	是	错误码，成功时为0
message	String	是	API 错误信息
data	Array	是	返回数据
data.has_more	Bool	是	是否有内容可以继续往前/往后翻页
data.context	String	是	透传字段，查看第一页，则传空字符串。若需要翻页，需要将前一页返回值中的context透传到参数中。order用于指定翻页顺序。若order填0，则从当前页正序/往下翻页；若order填1，则从当前页倒序/往上翻页
data.dircount	String	是	子目录数量(总)
data.filecount	String	是	子文件数量(总)

参数名	类型	是否必然返回	参数描述
data.infos	Array	是	文件、目录集合，可以为空
data.infos.name	String	是	文件或目录名
data.infos.biz_attr	String	是	目录或文件属性，业务端维护
data.infos.ctime	String	是	目录或文件的创建时间，unix时间戳
data.infos.mtime	String	是	目录或文件的修改时间，unix时间戳
data.infos.filesize	Int	否(当类型为文件时返回)	文件大小
data.infos.filelen	Int	否(当类型为文件时返回)	文件已传输大小(通过与filesize对比可知文件传输进度)
data.infos.sha	String	否(当类型为文件时返回)	文件sha
data.infos.access_url	String	否(当类型为文件时返回)	生成的下载url

示例

```
CosApiClientOption client_option("your appid", "your secretId", "your secretKey");
CosApi cos_client(client_option);
string bucketName = "myBucket";
string path = "/myFolder/";
string result = cos_client.PrefixSearch(bucketName, path);
```

文件操作

文件上传

接口说明：用于较小文件(一般小于8MB)的上传，可以通过此接口上传较小的文件并获得文件的url，较大的文件请使用分片上传接口。

方法原型

```
string Upload(const string &srcPath,
  const string &bucketName,
```

```
const string &dstPath,
const CustomOptions& options = DefaultUploadOptions());
```

参数说明

参数名	类型	是否必填	默认值	参数描述
srcPath	String	是	无	本地要上传文件的全路径
bucketName	String	是	无	bucket名称，bucket创建参见创建Bucket
dstPath	String	是	无	文件在COS服务端的全路径，不包括/appid/bucketname
options	CustomOptions	否	空	用户自定义选项，为一组key-value对

Options说明

参数名	类型	是否必填	默认值	**参数描述
biz_attr	String	否	""	目录/文件属性，业务端维护

返回结果说明

通过函数返回值返回请求结果的json字符串

参数名	类型	是否必然返回	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息
data	Array	是	返回数据
data.access_url	Bool	是	生成的文件下载url
data.url	String	是	操作文件的url
data.resource_path	String	是	资源路径.格式:/appid/bucket/xxx

示例

```
CosApiClientOption client_option("your appid", "your secretId", "your secretKey");
CosApi cos_client(client_option);
string bucketName = "myBucket";
string srcPath= "/data/test.mp4";
string dstPath = "/myFolder/test.mp4";
CustomOptions upload_options;
upload_options.AddStringOption("biz_attr", "test");
string result = cos_client.Upload(srcPath, bucketName, dstPath, upload_options);
```

文件分片上传

接口说明：用于较大文件(一般大于8MB)的上传，可以通过此接口上传较大文件并获得文件的url和唯一标识resource_path（用于调用其他api）。

方法原型

```
string UploadSlice(const string &srcPath,
  const string &bucketName,
  const string &dstPath,
  const CustomOptions& options = DefaultUploadSliceOptions());
```

参数说明

参数名	类型	是否必填	默认值	参数描述
srcPath	String	是	无	本地要上传文件的全路径
bucketName	String	是	无	bucket名称，bucket创建参见创建Bucket
dstPath	String	是	无	上传到cos的路径
options	CustomOptions	否	空	用户自定义选项，为一组key-value对

Options说明

参数名	类型	是否必填	默认值	参数描述
slice_size	Int	是	512*1024字节	分片大小，用户可以根据网络状况自行设置
biz_attr	String	否	空	文件属性，业务端维护
insertOnly	String	否	1	是否覆盖文件，0-允许，1-不允许

返回结果说明

通过函数返回值返回请求结果的json字符串：

参数名	类型	必然返回	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息
data	Array	是	返回数据
data.access_url	Bool	是	生成的文件下载url
data.url	String	是	操作文件的url
data.resource_path	String	是	资源路径. 格式:/appid/bucket/xxx

示例

```
CosApiClientOption client_option("your appid", "your secretId", "your secretKey");
CosApi cos_client(client_option);
string bucketName = "myBucket";
string srcPath= "/data/test.mp4";
string dstPath = "/myFolder/test.mp4";
CustomOptions upload_options;
upload_options.AddStringOption("biz_attr", "test");
string result = cos_client.UploadSlice(srcPath, bucketName, dstPath, upload_options);
```

文件属性更新

接口说明：用于目录业务自定义属性的更新，可以通过此接口更新业务的自定义属性字段。

方法原型

```
int UpdateFile(const string &bucketName,
  const string &path,
  const CustomOptions& options = CustomOptions());
```

参数说明

参数名	类型	是否必填	默认值	参数描述
bucketName	String	是	无	bucket名称，bucket创建参见创建Bucket
path	String	是	无	文件在对象存储服务端的全路径，不包括/appid/bucketname
options	CustomOptions	否	空	用户自定义选项，为一组key-value对

Options说明

参数名	类型	是否必填	默认值	参数描述
biz_attr	String	否	无	待更新的文件属性信息

返回结果说明

通过函数返回值返回请求结果的json字符串：

参数名	类型	是否必然返回	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息

示例

```
CosApiClientOption client_option("your appid", "your secretId", "your secretKey");
CosApi cos_client(client_option);
string bucketName = "myBucket";
string path = "/data/test.mp4";
string result = cos_client.UpdateFile(bucketName， path);
```

文件查询

接口说明：用于文件的查询，可以通过此接口查询文件的各项属性信息。

方法原型

```
int Stat(const string &bucketName, const string &path);
```

参数说明

参数名	类型	是否必填	默认值	参数描述
bucketName	String	是	无	bucket名称，bucket创建参见创建Bucket
path	String	是	无	文件在COS服务端的全路径，不包括/appid/bucketname

返回结果说明

通过函数返回值返回请求结果的json字符串

参数名	类型	必然返回	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息
data	Array	是	文件属性数据
data.name	String	是	文件或目录名
data.biz_attr	String	是	文件属性，业务端维护

参数名	类型	必然返回	参数描述
data.ctime	String	是	文件的创建时间，unix时间戳
data.mtime	String	是	文件的修改时间，unix时间戳
data.filesize	Int	是	文件大小
data.filelen	Int	是	文件已传输大小(通过与filesize对比可知文件传输进度)
data.sha	String	是	文件sha
data.access_url	String	是	生成的文件下载url

示例

```
CosApiClientOption client_option("your appid", "your secretId", "your secretKey");
CosApi cos_client(client_option);
string bucketName = "myBucket";
string path = "/data/test.mp4";
string result = cos_client.StatFile(bucketName, path);
```

文件删除

接口说明：用于文件的删除，可以通过此接口删除已经上传的文件。

方法原型

```
int DelFile(const string &bucketName, const string &path);
```

参数说明

参数名	类型	是否必填	默认值	参数描述
bucketName	String	是	无	bucket名称，bucket创建参见创建Bucket
path	String	是	无	文件在对象存储服务端的全路径，不包括/

参数名	类型	是否必填	默认值	参数描述
				appid/bucketname

返回结果说明

通过函数返回值返回请求结果的json字符串

参数名	类型	是否必然返回	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息

示例

```
CosApiClientOption client_option("your appid", "your secretId", "your secretKey");
CosApi cos_client(client_option);
string bucketName = "myBucket";
string path = "/data/test.mp4";
string result = cos_client.DelFile(bucketName, path);
```

文件移动

接口说明：用于文件移动，可以通过此接口移动已经上传的文件。

方法原型

```
std::string MoveFile(
  const std::string& bucketName,
  const std::string& srcPath,
  const std::string& dstPath,
  const CustomOptions& options = DefaultRenameFileOptions()
);
```

参数说明

参数名	类型	是否必填	默认值	参数描述
bucketName	String	是	无	bucket名称，bucket创建参见创建Bucket
srcPath	String	是	无	要移动文件的源地址
dstPath	String	是	无	要移动到的目的地址
options	CustomOptions	否	空	用户自定义选项，为一组key-value对

返回结果说明

通过函数返回值返回请求结果的json字符串

参数名	类型	是否必然返回	**参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息

示例

```
CosApiClientOption client_option("your appid", "your secretId", "your secretKey");
CosApi cos_client(client_option);
string bucketName = "myBucket";
string srcPath = "/data/test.mp4";
string dstPath = "/data/test.mp4.rename";
string result = cos_client.MoveFile(bucketName, srcPath, dstPath);
```

C sharp SDK

开发准备

SDK 获取

对象存储服务的C sharp SDK的下载地址

开发准备

1. sdk依赖C sharp 4.0版本及以上， 推荐使用相同的版本。
2. 从控制台获取APP ID、SecretID、SecretKey，详情参考权限控制。

SDK 配置

直接下载github上提供的源代码，集成到开发环境。

若需 HTTPS 支持，则将 cos_dotnet_sdk/Api/CosCloud.cs 文件中变量 COSAPI_CGI_URL 中的 http 修改为 https 即可。

生成签名

多次有效签名

方法原型

``` c sharp
public static string Signature(int appId, string secretId, string secretKey, long expired, string bucketName)
```

参数说明

参数名	**类型**	**必填**	**参数描述**

appId	int	是	AppId
secretId	string	是	Secret Id
secretKey	string	是	Secret Key，以上三项从[控制台](/document/product/430/5904)获取。
expired	long	是	过期时间，Unix时间戳
bucketName	string	是	bucket名称，bucket创建参见[创建Bucket](/document/product/430/5887)

示例

```c sharp
var sign = Sign.Signature(appId, secretId, secretKey, expired, bucketName);
```

单次有效签名

方法原型

```c sharp
public static string SignatureOnce(int appId, string secretId, string secretKey, string remotePath, string bucketName)
```

参数说明

参数名	**类型**	**必填**	**参数描述**
appId	int	是	AppId
secretId	string	是	Secret Id
secretKey	string	是	Secret Key，以上三项从[控制台](/document/product/430/5904)获取。
bucketName	string	是	bucket名称，bucket创建参见[创建Bucket](/document/product/430/5887)
remotePath	string	是	文件唯一的标识，格式/appid/bucketname/filepath/filename，其中/filepath/filename为文件在此bucketname下的全路径，

示例

```c sharp
```

var sign = Sign.SignatureOnce(appId, secretId, secretKey,remotePath, bucketName);

更多签名详细说明，请参考权限控制 。

目录操作

创建目录

接口说明：用于目录的创建，可以通过此接口在指定bucket下创建目录。

方法原型

``` c sharp
public string CreateFolder(string bucketName, string remotePath, Dictionary parameterDic = null)
```

参数说明

参数名	**类型**	**必填**	**参数描述**
bucketName	string	是	bucket名称
remotePath	string	是	需要创建目录的全路径
parameterDic	string	否	属性字典

属性字典说明

参数名	类型	必填	参数描述
biz_attr	string	否	目录属性

返回结果说明

参数名	**类型**	必带	**参数描述**

code	Int	是	错误码，成功时为0
message	String	是	错误信息
data	Array	否	返回数据，请参考《Restful API 创建目录》

示例

```c sharp
var createFolderParasDic = new Dictionary<string, string>();
  createFolderParasDic.Add(CosParameters.PARA_BIZ_ATTR,"new attribute");
var result = cos.CreateFolder(bucketName, folder, createFolderParasDic);
```

目录更新

接口说明：用于目录业务自定义属性的更新，可以通过此接口更新业务的自定义属性字段。

方法原型

```c sharp
public string UpdateFolder(string bucketName, string remotePath, Dictionary parameterDic = null)
```

参数说明

参数名	**类型**	**必填**	**参数描述**
bucketName	string	是	bucket名称
remotePath	string	是	需要创建目录的全路径
parameterDic	string	是	目录更新参数字典

目录更新参数字典

参数名	类型	必填	参数描述
biz_attr	string	否	目录属性

返回结果说明

参数名	**类型**	必选	**参数描述**
code	Int	是	错误码，成功时为0
message	String	是	错误信息

示例

```c sharp
var updateParasDic = new Dictionary<string, string>();
  updateParasDic.Add(CosParameters.PARA_BIZ_ATTR,"new attribute");
var result = cos.UpdateFolder(bucketName, folder, updateParasDic);
```

目录查询

接口说明：用于目录属性的查询，可以通过此接口查询目录的属性。

方法原型

```c sharp
public string GetFolderStat(string bucketName, string remotePath)
```

参数说明

参数名	**类型**	**必填**	**参数描述**
bucketName	string	是	bucket名称
remotePath	string	是	目录的全路径

返回结果说明

参数名	**类型**	必选	**参数描述**
code	Int	是	错误码，成功时为0
message	String	是	错误信息
data	Array	否	目录属性数据，请参考《Restful API 目录查询》

示例

```c sharp
var result = cos.GetFolderStat(bucketName, folder);
```

目录删除

接口说明：用于目录的删除，可以通过此接口删除空目录，如果目录中存在有效文件或目录，将不能删除。

方法原型

```c sharp
public string DeleteFolder(string bucketName, string remotePath)
```

参数说明

参数名	**类型**	**必填**	**参数描述**
bucketName	string	是	bucket名称
remotePath	string	是	目录的全路径

返回结果说明

参数名	**类型**	必选	**参数描述**
code	Int	是	错误码，成功时为0
message	String	是	错误信息

示例

```c sharp
var result = cos.DeleteFolder(bucketName, folder);
```

列举目录下文件&目录

接口说明：用于列举目录下文件和目录，可以通过此接口查询目录下的文件和目录属性。

方法原型

```c sharp
public string GetFolderList(string bucketName, string remotePath, Dictionary parameterDic = null)
```

参数说明

参数名	**类型**	**必填**	**参数描述**
bucketName	string	是	bucket名称
remotePath	string	是	目录的全路径
parameterDic	Dictionary<string, string>	是	目录列表参数字典

目录列表参数字典

参数名	**类型**	**必填**	参数描述
num	string	是	要查询的目录/文件数量，最大1000
listFlag	string	否	listFlag:大于0返回全部数据，否则返回部分数据
context	String	否	透传字段,用于翻页,前端不需理解,需要往后翻页则透传回来
prefix	string	否	搜索前缀

返回结果说明

参数名	**类型**	**必带**	**参数描述**
code	Int	是	API 错误码，成功时为0
message	String	是	错误信息
data	Array	是	返回数据，请参考《Restful API 目录列表》

示例

```c sharp
var folderlistParasDic = new Dictionary<string, string>();
  folderlistParasDic.Add(CosParameters.PARA_NUM,"100");
var result = cos.GetFolderList(bucketName, folder, folderlistParasDic);
```

文件操作

文件上传

接口说明：文件上传的统一接口，对于大于8m的文件，内部会通过多次分片的方式进行文件上传。没有断点续传功能，需要自己用分片上传接口实现。

方法原型

```c sharp
public string UploadFile(string bucketName, string remotePath, string localPath, Dictionary
parameterDic = null)
```

参数说明

参数名	**类型**	**必填**	**参数描述**
bucketName	string	是	bucket名称，bucket创建参见[创建Bucket](/document/product/430/5887)
remotePath	string	是	文件在服务端的全路径

| localPath | string | 是 | 文件本地路径 |
| parameterDic | Dictionary<string, string> | 否 | 文件上传参数字典 |

文件上传参数字典

参数名	类型	必填	参数描述
biz_attr	string	否	文件属性
insertOnly	string	否	同名文件是否覆盖；0：覆盖，1：不覆盖（默认）
slice_size	string	否	可选取值为:64*1024 512*1024，1*1024*1024（默认），2*1024*1024，3*1024*1024

返回结果说明

参数名	**类型**	**必带**	**参数描述**
code	Int	是	错误码，成功时为0
message	String	是	错误信息
data	Array	是	返回数据，请参考《Restful API 创建文件》

示例

```c sharp
var uploadParasDic = new Dictionary<string, string>();
uploadParasDic.Add(CosParameters.PARA_BIZ_ATTR,"file attribute");
uploadParasDic.Add(CosParameters.PARA_INSERT_ONLY,"0");
uploadParasDic.Add(CosParameters.PARA_SLICE_SIZE,SLICE_SIZE.SLIZE_SIZE_3M.ToString());
result = cos.UploadFile(bucketName, remotePath, localPath, uploadParasDic);
```

文件属性更新

接口说明：用于目录业务自定义属性的更新，可以通过此接口更新业务的自定义属性字段。

方法原型

``` c sharp
public string UpdateFile(string bucketName, string remotePath, Dictionary parameterDic = null)
```

参数说明

参数名	**类型**	**必填**	**参数描述**
bucketName	string	是	bucket名称
remotePath	string	是	文件在Cos的全路径
parameterDic	string	否	文件更新参数字典

文件更新参数字典

参数名	类型	必填	参数描述
biz_attr	string	否	待更新的文件属性信息
authority	string	否	eInvalid(继承Bucket的读写权限)；eWRPrivate(私有读写)；eWPrivateRPublic(公有读私有写)
Cache-Control	string	否	指定请求和响应遵循的缓存机制，如：no-cache；max-age=200
Content-Type	string	否	返回内容的MIME类型，如：text/html
Content-Disposition	string	否	控制用户请求所得的内容存为一个文件的时候提供一个默认的文件名，如：attachment; filename="fname.ext"
Content-Language	string	否	使用的语言，如：zh-CN
x-cos-meta-自定义内容	string	否	表示以“x-cos-meta-”名字开头的参数，用户按照自身业务场景，设置需要在 Header 中传输什么参数

返回结果说明

参数名	**类型**	**必带**	**参数描述**
code	Int	是	错误码，成功时为0

| message | String | 是 | 错误信息 |

示例

```c sharp
var optionParasDic = new Dictionary<string, string>();
  optionParasDic.Add(CosParameters.PARA_BIZ_ATTR,"new attribute");
  optionParasDic.Add(CosParameters.PARA_AUTHORITY,AUTHORITY.AUTHORITY_PRIVATEPUBLIC);
  optionParasDic.Add(CosParameters.PARA_CACHE_CONTROL,"no");
  optionParasDic.Add(CosParameters.PARA_CONTENT_TYPE,"application/text");
  optionParasDic.Add(CosParameters.PARA_CONTENT_DISPOSITION,"filename=\"test.pdf\"");
  optionParasDic.Add(CosParameters.PARA_CONTENT_LANGUAGE,"en");
  optionParasDic.Add("x-cos-meta-test","test");
result = cos.UpdateFile(bucketName, remotePath, optionParasDic);
```

文件查询

接口说明：用于文件的查询，可以通过此接口查询文件的各项属性信息。

方法原型

```c sharp
public string GetFileStat(string bucketName, string remotePath)
```

参数说明

参数名	**类型**	**必填**	**参数描述**
bucketName	string	是	bucket名称
remotePath	string	是	文件在Cos的全路径

返回结果说明

参数名	**类型**	**必带**	**参数描述**
code	Int	是	错误码，成功时为0
message	String	是	错误信息
data	Array	是	文件属性数据，请参考《Restful API 文件查询》

示例

```c sharp
var result = cos.GetFileStat(bucketName, remotePath);
```

文件删除

接口说明：用于文件的删除，可以通过此接口删除已经上传的文件。

方法原型

```c sharp
public string DeleteFile(string bucketName, string remotePath)
```

参数说明

参数名	**类型**	**必填**	**参数描述**
bucketName	string	是	bucket名称
remotePath	string	是	文件在服务端的全路径

返回结果说明

参数名	**类型**	**必带**	**参数描述**
code	Int	是	错误码，成功时为0
message	String	是	错误信息

示例

```c sharp
var result = cos.DeleteFile(bucketName, remotePath);
```

分片上传list

接口说明：查询分片上传的情况

方法原型

```c sharp
public string UploadSliceList(string bucketName, string remotePath)
```

参数说明

参数名	**类型**	**必填**	**参数描述**
bucketName	string	是	bucket名称
remotePath	string	是	文件在服务端的全路径

返回结果说明

参数名	**类型**	**必带**	**参数描述**
code	Int	是	错误码，成功时为0
message	String	是	错误信息
data	Array	是	分片上传信息

上传完成data的数据说明

参数名	**类型**	**必带**	**参数描述**

url	String	是	操作文件的url
access_url	String	是	生成的文件下载url
source_url	String	是	源地址
resource_path	String	是	资源路径. 格式:/appid/bucket/xxx
preview_url	String	否	预览地址

上传未完成data的数据说明

参数名	**类型**	**必带**	**参数描述**
session	String	是	init返回的标识
filesize	Int	是	文件大小
slice_size	Int	是	分片大小（64K-3M）大于1M 必须为1M 整数倍
sha	String	否	文件的全文sha值，init时带了则返回
listparts	Json Array	是	已上传完成的分片，形如：[{“offset”:0, “datalen”:1024}, {}, {}].

示例
```c sharp
var fileSha = SHA1.GetFileSHA1(localPath);
var result = SliceUploadInit(bucketName, remotePath, localPath, fileSha, bizAttribute, sliceSize,
insertOnly);
```

分片上传init

接口说明：分片上传的第一次握手,如果上一次分片上传未完成，会返回{"code":-4019,"message":"_ERROR_FILE_NOT_FINISH_UPLOAD"}init
接口说明：分片上传的第一次握手

方法原型

```c sharp
public string SliceUploadInit(string bucketName, string remotePath, string localPath, string
fileSha,string bizAttribute = "", int sliceSize = SLICE_SIZE.SLIZE_SIZE_1M, int insertOnly = 1)
```

参数说明

参数名	**类型**	**必填**	**参数描述**
bucketName	string	是	bucket名称
remotePath	string	是	文件在服务端的全路径
localPath	string	是	本地文件路径
fileSha	stirng	是	文件的Sha值

返回结果说明

参数名	**类型**	**必带**	**参数描述**
code	Int	是	错误码，成功时为0
message	String	是	错误信息
data	Array	是	分片上传信息

data的数据说明

参数名	**类型**	**必带**	**参数描述**
session	string	否	(非秒传的大部分情况会有) 唯一标识此文件传输过程的id
slice_size	Int	否	(非秒传大部分情况下会有) 分片大小
serial_upload	int	否	(非秒传大部分情况下会有) 1：只支持串行分片上传其它：支持并行分片

示例

```c sharp
var fileSha = SHA1.GetFileSHA1(localPath);
var result = SliceUploadInit(bucketName, remotePath, localPath, fileSha, bizAttribute, sliceSize,
insertOnly);
```

分片上传

接口说明：分片上传数据

方法原型

``` c sharp
public string SliceUploadData(string bucketName, string remotePath, string localPath, string fileSha, string session, long offset,int sliceSise,string sign)
```

参数说明

参数名	**类型**	**必填**	**参数描述**
bucketName	string	是	bucket名称
remotePath	string	是	文件在服务端的全路径
localPath	string	是	本地文件路径
fileSha	stirng	是	文件的Sha值
session	string	是	唯一标识此文件传输过程的id
offset	int	是	分片的偏移量
sliceSize	int	是	分片的大小
sign	string	是	签名（多次）

返回结果说明

参数名	**类型**	**必带**	**参数描述**
code	Int	是	错误码，成功时为0
message	String	是	错误信息
data	Array	是	分片上传信息

data的数据说明

参数名	**类型**	**必带**	**参数描述**
session	string	是否	(非秒传的大部分情况会有) 唯一标识此文件传输过程的id
offset	Int	是	当前分片的offset
datalen	int	是	分片文件长度slice_size
serial_upload	int	否	(非秒传大部分情况下会有) 1：只支持串行分片上传其它：支持并行分片

示例

```c sharp
var fileSha = SHA1.GetFileSHA1(localPath);
var result = SliceUploadDataInit(bucketName, remotePath, localPath, fileSha, session, offset,
sliceSize,sign);
```

分片上传 finish

接口说明：告诉COS所有分片上传成功

方法原型

```c sharp
public string SliceUploadFinish(string bucketName, string remotePath, string localPath, string fileSha,
string session)
```

参数说明

参数名	**类型**	**必填**	**参数描述**
bucketName	string	是	bucket名称
remotePath	string	是	文件在服务端的全路径
localPath	string	是	本地文件路径
fileSha	stirng	是	文件的Sha值
session	string	是	唯一标识此文件传输过程的id

返回结果说明

参数名	**类型**	**必带**	**参数描述**
code	Int	是	错误码，成功时为0
message	String	是	错误信息
data	Array	是	分片上传信息

data的数据说明

参数名	**类型**	**必带**	**参数描述**
session	string	是	(非秒传的大部分情况会有) 唯一标识此文件传输过程的id
offset	Int	是	当前分片的offset
datalen	int	是	分片文件长度

示例

```c sharp
result = SliceUploadFinish(bucketName,remotePath,localPath,fileSha, sessionbizAttribute, sliceSize, insertOnly);
```

JavaScript SDK

开发准备

SDK 获取

对象存储服务的Javascript SDK的下载地址： https://github.com/tencentyun/cos-js-sdk

开发环境

1. SDK需要浏览器支持HTML 5；
2. SDK需要浏览器支持Flash；
3. 从控制台获取APP ID、SecretID、SecretKey，详情参考权限控制 ；

SDK 配置

直接下载github上提供的源代码，将SDK中sdk文件夹包含到您的项目中，使用SDK之前，加载相关支持的Javascript文件即可。

```
<script type="text/javascript" src="sdk/jquery1x.min.js"></script>
<script type="text/javascript" src="sdk/qcloud_sdk.js"></script>
<script type="text/javascript" src="sdk/swfobject.js"></script>
```

本SDK需要jQuery的支持，如果您的项目已经引入jQuery，那么可以省略jquery1x.min.js的引入。

生成签名

多次有效签名

方法原型

public static function appSign($expired, $bucketName)

参数说明

参数名	类型	是否必填	默认值	参数描述
expired	long	否	无	过期时间，Unix时间戳
bucketName	String	否	无	bucket名称，bucket创建参见创建Bucket

返回结果说明

返回值：签名字符串

单次有效签名

方法原型

public static function appSign_once($path, $bucketName)

参数说明

参数名	类型	是否必填	默认值	参数描述
bucketName	String	否	无	bucket名称，bucket创建参见创建Bucket
path	String	是	无	文件路径，以斜杠开头，例如/filepath/filename，为文件在此bucketname下的全路径

返回结果说明

返回值：签名字符串

目录操作

在进行接口调用操作前，需要生成 CosCloud 对象，对象原型入下：

```
var cos = new CosCloud(appid, signUrl);
```

- appid：为项目的 APPID；
- signUrl：为服务端签名地址，默认为当前目录下Sign.php。

SDK的每个方法都需要三个通用参数：

successCallBack
：当方法能从服务端正确的取回数据时，回调的方法。结果以字符串形式返回，方法定义如下：

```
var successCallBack = function(result){
  $("#result").val(result);
};
```

errorCallBack：当请求服务端出现网络错误时，调用该方法。方法定义如下：

```
var errorCallBack = function(result){
  $("#result").val(result.responseText);
}
```

bucketName：bucket名称，bucket创建参见[创建Bucket

创建目录

接口说明：可以通过此接口在指定bucket下创建目录。

方法原型

```
CosCloud.prototype.createFolder = function(success, error, bucketName, remotePath)
```

参数说明

参数名	类型	是否必填	默认值	参数描述
success	Function	是	无	参见通用参数successCallBack
error	Function	是	无	参见通用参数errorCallBack
bucketName	String	是	无	参见通用参数bucketName
remotePath	String	是	无	需要创建目录的全路径

返回结果说明

参数名	类型	参数描述
code	Int	错误码，成功时为0
message	String	错误信息
data	Array	返回数据
data.ctime	String	目录的创建时间，unix时间戳
data.resource_path	String	目录的资源路径

示例

```
cos.createFolder(successCallBack, errorCallBack, bucketName, "/newfolder/");
```

目录属性更新

接口说明：用于目录业务自定义属性的更新，可以通过此接口更新业务的自定义属性字段。

方法原型

```
CosCloud.prototype.updateFolder = function(success, error, bucketName, remotePath, bizAttribute)
```

参数说明

参数名	类型	是否必填	默认值	参数描述
success	Function	是	无	参见通用参数successCallBack
error	Function	是	无	参见通用参数errorCallBack
bucketName	String	是	无	参见通用参数bucketName
remotePath	String	是	无	需要创建目录的全路径
bizAttribute	String	是	无	新的目录绑定的属性信息

返回结果说明

参数名	类型	参数描述
code	Int	错误码，成功时为0
message	String	错误信息

示例

```
cos.updateFolder(successCallBack, errorCallBack, bucketName, "/newfolder/", "This is sdk test folder");
```

目录查询

接口说明：用于目录属性的查询，可以通过此接口查询目录的属性。

方法原型

```
CosCloud.prototype.getFolderStat = function(success, error, bucketName, remotePath)
```

参数说明：

参数名	类型	是否必填	默认值	参数描述
success	Function	是	无	参见通用参数successCallBack
error	Function	是	无	参见通用参数errorCallBack
bucketName	String	是	无	参见通用参数bucketName
remotePath	String	是	无	目录的全路径

返回结果说明

参数名	类型	参数描述
code	Int	错误码，成功时为0
message	String	错误信息
data	Array	目录属性数据
data.biz_attr	String	目录绑定的属性信息，业务自行维护
data.ctime	String	目录的创建时间，unix时间戳
data.mtime	String	目录的修改时间，unix时间戳
data.name	String	目录的名称

示例

```
cos.getFolderStat(successCallBack, errorCallBack, bucketName, "/newfolder/");
```

目录删除

接口说明：用于目录的删除，可以通过此接口删除空目录，如果目录中存在有效文件或目录，将不能删除。

方法原型

```
CosCloud.prototype.deleteFolder = function(success, error, bucketName, remotePath)
```

参数说明

参数名	类型	是否必填	默认值	参数描述

参数名	类型	是否必填	默认值	参数描述
success	Function	是	无	参见通用参数success CallBack
error	Function	是	无	参见通用参数errorCallBack
bucketName	String	是	无	参见通用参数bucket Name
remotePath	String	是	无	目录的全路径

返回结果说明

参数名	类型	参数描述
code	Int	错误码，成功时为0
message	String	错误信息

示例

```
cos.deleteFolder(successCallBack, errorCallBack, bucketName, "/newfolder/");
```

列举目录下文件&目录

接口说明：用于列举目录下文件和目录，可以通过此接口查询目录下的文件和目录属性。

方法原型

```
CosCloud.prototype.getFolderList = function(success, error, bucketName, remotePath, num, context, order, pattern, prefix)
```

参数说明

参数名	类型	是否必填	默认值	参数描述
success	Function	是	无	参见通用参数success CallBack
error	Function	是	无	参见通用参数errorCa

参数名	类型	是否必填	默认值	参数描述
				llBack
bucketName	String	是	无	参见通用参数bucketName
remotePath	String	是	无	目录的全路径
num	int	是	无	要查询的目录/文件数量
context	String	是	无	透传字段，查看第一页，则传空字符串。若需要翻页，需要将前一页返回值中的context透传到参数中。order用于指定翻页顺序。若order填0，则从当前页正序/往下翻页；若order填1，则从当前页倒序/往上翻页
order	int	是	无	默认正序(=0), 填1为反序
pattern	String	否	eListBoth	eListBoth,eListDirOnly,eListFileOnly 默认both
prefix	String	否	无	列出含此前缀的所有文件

返回结果说明

参数名	类型	是否必然返回	参数描述
code	Int	是	API 错误码，成功时为0
message	String	是	错误信息
data	Array	是	返回数据
data.has_more	Bool	是	是否有内容可以继续往前/往后翻页
data.context	String	是	透传字段，查看第一页，则

参数名	类型	是否必然返回	参数描述
			传空字符串。若需要翻页，需要将前一页返回值中的context透传到参数中。order用于指定翻页顺序。若order填0，则从当前页正序/往下翻页；若order填1，则从当前页倒序/往上翻页
data.dircount	String	是	子目录数量(总)
data.filecount	String	是	子文件数量(总)
data.infos	Array	是	文件、目录集合，可以为空
data.infos.name	String	是	文件或目录名
data.infos.biz_attr	String	是	目录或文件属性，业务端维护
data.infos.ctime	String	是	目录或文件的创建时间，unix时间戳
data.infos.mtime	String	是	目录或文件的修改时间，unix时间戳
data.infos.filesize	Int	否(当类型为文件时返回)	文件大小
data.infos.filelen	Int	否(当类型为文件时返回)	文件已传输大小(通过与filesize对比可知文件传输进度)
data.infos.sha	String	否(当类型为文件时返回)	文件sha
data.infos.access_url	String	否(当类型为文件时返回)	生成的文件下载url
data.infos.authority	String	否	eInvalid,eWRPrivate,eWPrivateRPublic,文件可以与bucket拥有不同的权限类型，已经设置过权限的文件如果想要撤销，直接赋值为eInvalid，则会采用bucket的权限

示例

```
cos.getFolderList(successCallBack, errorCallBack, bucketName, "/", 20, "", 0);
```

文件操作

文件上传

接口说明：用于较小文件(一般小于10MB)的上传，可以通过此接口上传较小的文件并获得文件的url，较大的文件请使用分片上传接口。

方法原型

CosCloud.prototype.uploadFile = function(success, error, bucketName, remotePath, file)

参数说明

参数名	类型	是否必填	默认值	参数描述
success	Function	是	无	参见通用参数success CallBack
error	Function	是	无	参见通用参数errorCallBack
bucketName	String	是	无	参见通用参数bucket Name
remotePath	String	是	无	文件在COS服务端的全路径，不包括/appid/bucketname
file	File	是	无	本地要上传文件的文件对象
insertOnly	Int	否	无	insertOnly==0 表示允许覆盖文件 1表示不允许 其他值忽略

返回结果说明

参数名	类型	是否必然返回	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息

参数名	类型	是否必然返回	参数描述
data	Array	是	返回数据
data.access_url	Bool	是	生成的文件下载url
data.url	String	是	操作文件的url
data.resource_path	String	是	资源路径. 格式:/appid/bucket/xxx

示例

```
cos.uploadFile(successCallBack, errorCallBack, bucketName, "/tel.txt", files[0]);
```

文件分片上传

接口说明：用于较大文件(一般大于10MB)的上传，可以通过此接口上传较大文件并获得文件的url和唯一标识resource_path（用于调用其他api）。

方法原型

```
CosCloud.prototype.sliceUploadFile = function(success, error, bucketName, remotePath, file)
```

参数说明

参数名	类型	是否必填	默认值	参数描述
success	Function	是	无	参见通用参数successCallBack
error	Function	是	无	参见通用参数errorCallBack
bucketName	String	是	无	参见通用参数bucketName
remotePath	String	是	无	文件在COS服务端的全路径，不包括/appid/bucketname
file	File	是	无	本地要上传文件的文件对象

参数名	类型	是否必填	默认值	参数描述
insertOnly	Int	否	无	insertOnly==0 表示允许覆盖文件 1表示不允许 其他值忽略
sliceSize	Int	否	无	分片大小，正整数类型，目前支持512K,1M,2M,3M，512K==512 1024,1M==11024*1024

返回结果说明

参数名	类型	必然返回	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息
data	Array	是	返回数据
data.access_url	Bool	是	生成的文件下载url
data.url	String	是	操作文件的url
data.resource_path	String	是	资源路径. 格式:/appid/bucket/xxx

示例

```
cos.sliceUploadFile(successCallBack, errorCallBack, bucketName, "/movie/" + files[0].name, files[0])
```

文件属性更新

接口说明：用于目录业务自定义属性的更新，可以通过此接口更新业务的自定义属性字段。

方法原型

```
CosCloud.prototype.getFileStat = function(success, error, bucketName, remotePath)
```

参数说明：

参数名	类型	是否必填	默认值	参数描述
success	Function	是	无	参见通用参数success CallBack
error	Function	是	无	参见通用参数errorCallBack
bucketName	String	是	无	参见通用参数bucket Name
remotePath	String	是	无	文件在COS服务端的全路径，不包括/appid/bucketname
file	File	是	无	本地要上传文件的文件对象
authority	String	否	无	eInvalid,eWRPrivate,eWPrivateRPublic,文件可以与bucket拥有不同的权限类型，已经设置过权限的文件如果想要撤销，直接赋值为eInvalid，则会采用bucket的权限
custom_headers	String	否	无	自定义header对象

自定义header对象说明

参数名	类型	是否必然返回	参数描述
Cache-Control	String	否	文件的缓存机制
Content-Type	String	否	文件的MIME信息
Content-Disposition	String	否	MIME协议的扩展
Content-Language	String	否	文件的语言
x-cos-meta-自定义内容	String	否	自定义信息

返回结果说明

参数名	类型	是否必然返回	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息

示例

```
cos.sliceUploadFile(successCallBack, errorCallBack, bucketName, "/movie/" + files[0].name, files[0])
```

文件查询

接口说明：用于文件的查询，可以通过此接口查询文件的各项属性信息。

方法原型

```
CosCloud.prototype.getFileStat = function(success, error, bucketName, remotePath)
```

参数说明

参数名	类型	是否必填	默认值	参数描述
success	Function	是	无	参见通用参数successCallBack
error	Function	是	无	参见通用参数errorCallBack
bucketName	String	是	无	参见通用参数bucketName
remotePath	String	是	无	文件在COS服务端的全路径，不包括/appid/bucketname

自定义header对象说明

参数名	类型	是否必然返回	参数描述
Cache-Control	String	否	文件的缓存机制
Content-Type	String	否	文件的MIME信息
Content-Disposition	String	否	MIME协议的扩展

参数名	类型	是否必然返回	参数描述
Content-Language	String	否	文件的语言
x-cos-meta-自定义内容	String	否	自定义信息

返回结果说明

参数名	类型	是否必然返回	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息
data	Array	是	文件属性数据
data.name	String	是	文件或目录名
data.biz_attr	String	是	文件属性，业务端维护
data.ctime	String	是	文件的创建时间，unix时间戳
data.mtime	String	是	文件的修改时间，unix时间戳
data.filesize	Int	是	文件大小
data.filelen	Int	是	文件已传输大小(通过与filesize对比可知文件传输进度)
data.sha	String	是	文件文件sha
data.access_url	String	是	生成的文件下载url
data.authority	String	否	无
data.custom_headers	String	否	无

示例

```
cos.getFileStat(successCallBack, errorCallBack, bucketName, "/tel.txt");
```

文件删除

接口说明：用于文件的删除，可以通过此接口删除已经上传的文件。

方法原型

```
CosCloud.prototype.deleteFile = function(success, error, bucketName, remotePath)
```

参数说明

参数名	类型	是否必填	默认值	参数描述
success	Function	是	无	参见通用参数successCallBack
error	Function	是	无	参见通用参数errorCallBack
bucketName	String	是	无	参见通用参数bucketName
remotePath	String	是	无	文件在COS服务端的全路径，不包括/appid/bucketname

返回结果说明

参数名	类型	必然返回	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息

示例

```
cos.deleteFile(successCallBack, errorCallBack, bucketName, "/tel.txt");
```

文件移动

接口说明：用于文件的移动和重命名。

方法原型

```
CosCloud.prototype.moveFile = function(success, error, bucketName, remotePath, destPath, overWrite)
```

参数说明

参数名	类型	是否必填	默认值	参数描述
success	Function	是	无	参见通用参数successCallBack
error	Function	是	无	参见通用参数errorCallBack
bucketName	String	是	无	参见通用参数bucketName
remotePath	String	是	无	文件在COS服务端的全路径，不包括/appid/bucketname
destPath	String	是	无	文件在COS服务端的全路径，不包括/appid/bucketname
overWrite	Int	否	无	是否覆盖同名文件, 0表示不覆盖 1表示覆盖 可选参数

返回结果说明

参数名	类型	必然返回	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息

示例

```
var destPath = "/222/tel.txt";
var overWrite = 1;
cos.moveFile(successCallBack, errorCallBack, bucketName, "/tel.txt", destPath, overWrite);
```

Node.js SDK

开发准备

SDK 获取

COS服务的Node.js sdk的下载地址： https://github.com/tencentyun/cos-nodejs-sdk.git

开发环境

1. sdk采用Node.js v0.10.29版本开发， 推荐使用相同的版本。
2. 从控制台获取APP ID、SecretID、SecretKey，详情参考权限控制。

SDK 配置

下载 Node.js SDK 后，可通过以下两种方式安装：

- 执行 npm install qcloud_cos 直接安装。
- 执行git clone https://github.com/tencentyun/cos-nodejs-sdk.git
  或者直接在github网站下载zip包

注意：sdk依赖formstream包，使用方法二需要自行安装此包。

在IDE中导入qcloud_cos包

```
var qcloud = require('qcloud_cos');
```

若需要支持 HTTPS 协议上传，则将 qcloud_cos/lib/conf.js 文件中变量 API_COS_END_POINT 中的http修改为 https 即可。

生成签名

多次有效签名

方法原型

function signMore(bucket, expired);

参数说明

参数名	类型	是否必填	默认值	参数描述
expired	long	是	无	过期时间，Unix时间戳
bucket	String	是	无	bucket名称，bucket创建参见创建Bucket

返回结果说明

签名字符串。

示例

```
var qcloud = require('qcloud_cos');

//如果conf.js中已经设置好APPID和SECRET_KEY则不需要此步骤。
qcloud.conf.setAppInfo('10000','xxxxxx','xxxxxx');

var expired = parseInt(Date.now() / 1000) + conf.EXPIRED_SECONDS;
var sign = qcloud.auth.signMore(bucket, expired);
```

单次有效签名

方法原型

function signOnce(bucket, fileid);

参数说明

参数名	类型	是否必填	默认值	参数描述
bucket	String	是	无	bucket名称，bucket创建参见创建Bucket
fileid	String	是	无	文件唯一的标识，格式/appid/bucketname/filepath/filename，其中/filepath/filename为文件在此bucketname下的全路径

返回结果说明

签名字符串

示例

```
var qcloud = require('qcloud_cos');

//如果conf.js中已经设置好APPID和SECRET_KEY则不需要此步骤。
qcloud.conf.setAppInfo('10000','xxxxxx','xxxxxx');

var sign = qcloud.auth.signOnce(bucket, '/'+conf.APPID+'/'+bucketname+'/'+remoteFilepath);
```

更多签名相关详细说明，请参考权限控制 。

目录操作

创建目录

接口说明：用于目录的创建，可以通过此接口在指定bucket下创建目录。

方法原型

function createFolder(bucket, path, bizattr, callback);

参数说明

参数名	类型	是否必填	默认值	参数描述
bucket	String	是	无	bucket名称，bucket创建参见创建Bucket
path	String	是	无	需要创建目录的全路径，以"/"开头,以"/"结尾，如果忘记api会补齐
bizattr	String	否	空串	目录绑定的属性信息，业务自行维护
callback	function	否	输出返回结果	结构为function(ret){}的函数，ret为json结构，默认直接输出。

返回结果说明

参数名	类型	参数描述
code	Int	错误码，成功时为0
message	String	错误信息
data	Array	返回数据
data.ctime	String	目录的创建时间，unix时间戳
data.resource_path	String	目录的资源路径

示例

```
var qcloud = require('qcloud_cos');

//如果conf.js中已经设置好APPID和SECRET_KEY则不需要此步骤。
qcloud.conf.setAppInfo('10000','xxxxxx','xxxxxx');
```

```
qcloud.cos.createFolder('bucketname', '/myFolder/', function(ret) {
  //deal with ret
});
```

目录属性更新

接口说明：用于目录业务自定义属性的更新，可以通过此接口更新业务的自定义属性字段。

方法原型

```
function updateFolder(bucket, path, bizattr, callback);
```

参数说明

参数名	类型	是否必填	默认值	参数描述
bucket	String	是	无	bucket名称，bucket创建参见创建Bucket
path	String	是	无	需要创建目录的全路径，以"/"开头,以"/"结尾，如果忘记api会补齐
bizattr	String	是	无	新的目录绑定的属性信息
callback	function	否	输出返回结果	结构为function(ret){}的函数，ret为json结构，默认直接输出。

返回结果说明

参数名	类型	参数描述
code	Int	错误码，成功时为0
message	String	错误信息

示例

```
var qcloud = require('qcloud_cos');

//如果conf.js中已经设置好APPID和SECRET_KEY则不需要此步骤。
qcloud.conf.setAppInfo('10000','xxxxxx','xxxxxx');

qcloud.cos.updateFolder('bucketname', '/myFolder/', 'bizAttribute', function(ret) {//deal with ret});
```

目录查询

接口说明：用于目录属性的查询，可以通过此接口查询目录的属性。

方法原型

```
function statFolder(bucket, path, callback);
```

参数说明

参数名	类型	是否必须	默认值	参数描述
bucket	String	是	无	bucket名称，bucket创建参见创建Bucket
path	String	是	无	目录的全路径，以"/"开头,以"/"结尾，如果忘记api会补齐
callback	function	否	输出返回结果	结构为function(ret){}的函数，ret为json结构，默认直接输出。

返回结果说明

参数名	类型	参数描述
code	Int	错误码，成功时为0
message	String	错误信息
data	Array	目录属性数据

参数名	类型	参数描述
data.biz_attr	String	目录绑定的属性信息，业务自行维护
data.ctime	String	目录的创建时间，unix时间戳
data.mtime	String	目录的修改时间，unix时间戳
data.name	String	目录的名称

示例

```
var qcloud = require('qcloud_cos');
//如果conf.js中已经设置好APPID和SECRET_KEY则不需要此步骤。
qcloud.conf.setAppInfo('10000','xxxxxx','xxxxxx');

qcloud.cos.statFolder('bucketname', '/myFolder/', function(ret) {
  //deal with ret
});
```

目录删除

接口说明：用于目录的删除，可以通过此接口删除空目录，如果目录中存在有效文件或目录，将不能删除。

方法原型

```
function deleteFolder(bucket, path, callback);
```

参数说明

参数名	类型	是否必填	默认值	参数描述
bucket	String	是	无	bucket名称，bucket创建参见创建Bucket
path	String	是	无	目录的全路径，以"/"开头,以"/"结尾，如果忘记api会补齐
callback	function	否	输出返回结果	结构为function(ret){}

参数名	类型	是否必填	默认值	参数描述
				的函数，ret为json结构，默认直接输出。

返回结果说明

参数名	类型	参数描述
code	Int	错误码，成功时为0
message	String	错误信息

示例

```
var qcloud = require('qcloud_cos');

//如果conf.js中已经设置好APPID和SECRET_KEY则不需要此步骤。
qcloud.conf.setAppInfo('10000','xxxxxx','xxxxxx');

qcloud.cos.deleteFolder('bucketname', '/myFolder/', function(ret) {
  //deal with ret
});
```

列举目录下文件&目录

接口说明：用于列举目录下文件和目录，可以通过此接口查询目录下的文件和目录属性。

方法原型

```
function list(bucket, path, num, pattern, order, context, callback);
```

参数说明

参数名	类型	是否必填	默认值	参数描述
bucket	String	是	无	bucket名称，bucket创建参见创建Bucket

参数名	类型	是否必填	默认值	参数描述
path	String	是	无	目录的全路径，以"/"开头,以"/"结尾，api会补齐
num	int	否	20	要查询的目录/文件数量
context	String	否	空串	透传字段，查看第一页，则传空字符串。若需要翻页，需要将前一页返回值中的context透传到参数中。order用于指定翻页顺序。若order填0，则从当前页正序/往下翻页；若order填1，则从当前页倒序/往上翻页
order	int	否	0	默认正序(=0), 填1为反序
pattern	String	否	eListBoth	pattern eListFileOnly=>仅列举文件，ListDirOnly=>仅列举目录，eListBoth=>列举目录&文件
callback	function	否	输出返回结果	结构为function(ret){}的函数，ret为json结构，默认直接输出。

返回结果说明

参数名	类型	是否必然返回	参数描述
code	Int	是	API 错误码，成功时为0
message	String	是	错误信息
data	Array	是	返回数据
data.has_more	Bool	是	是否有内容可以继续往前/

参数名	类型	是否必然返回	参数描述
			往后翻页
data.context	String	是	透传字段，查看第一页，则传空字符串。若需要翻页，需要将前一页返回值中的context透传到参数中。order用于指定翻页顺序。若order填0，则从当前页正序/往下翻页；若order填1，则从当前页倒序/往上翻页
data.dircount	String	是	子目录数量(总)
data.filecount	String	是	子文件数量(总)
data.infos	Array	是	文件、目录集合，可以为空
data.infos.name	String	是	文件或目录名
data.infos.biz_attr	String	是	目录或文件属性，业务端维护
data.infos.ctime	String	是	目录或文件的创建时间，unix时间戳
data.infos.mtime	String	是	目录或文件的修改时间，unix时间戳
data.infos.filesize	Int	否(当类型为文件时返回)	文件大小
data.infos.filelen	Int	否(当类型为文件时返回)	文件已传输大小(通过与filesize对比可知文件传输进度)
data.infos.sha	String	否(当类型为文件时返回)	文件sha
data.infos.access_url	String	否(当类型为文件时返回)	生成的文件下载url
data.infos.authority	String	否	eInvalid,eWRPrivate,eWPrivateRPublic,文件可以与bucket拥有不同的权限类型，已经设置过权限的文件如果想要撤销，直接赋值为eInvalid，则会采用bucket的权限

示例

```
varqcloud = require('qcloud_cos');

//如果conf.js中已经设置好APPID和SECRET_KEY则不需要此步骤。
qcloud.conf.setAppInfo('10000','xxxxxx','xxxxxx');

qcloud.cos.list('bucketname', '/myFolder/', 20, 'eListBoth', 0, '', function(ret) {//deal with ret});
```

列举目录下指定前缀文件&目录

接口说明：用于列举目录下指定前缀的文件和目录，可以通过此接口查询目录下的指定前缀的文件和目录信息。

方法原型

```
function prefixSearch(bucket, path, prefix, num, pattern, order, context, callback);
```

参数说明

参数名	类型	是否必填	默认值	参数描述
bucket	String	是	无	bucket名称，bucket创建参见创建Bucket
path	String	是	无	需要创建目录的全路径，以"/"开头,以"/"结尾，api会补齐
prefix	String	否	空串	读取文件/目录前缀
num	int	否	20	要查询的目录/文件数量
context	String	否	空串	透传字段，查看第一页，则传空字符串。若需要翻页，需要将前一页返回值中的context透传到参数中。order用于指定翻页顺

参数名	类型	是否必填	默认值	参数描述
				序。若order填0，则从当前页正序/往下翻页；若order填1，则从当前页倒序/往上翻页
order	int	否	0	默认正序(=0), 填1为反序
pattern	String	否	eListBoth	pattern eListFileOnly=>仅列举文件；ListDirOnly=>仅列举目录；eListBoth=>列举文件&目录
callback	function	否	输出返回结果	结构为function(ret){}的函数，ret为json结构，默认直接输出。

返回结果说明

参数名	类型	是否必然返回	参数描述
code	Int	是	错误码，成功时为0
message	String	是	API 错误信息
data	Array	是	返回数据
data.has_more	Bool	是	是否有内容可以继续往前/往后翻页
data.context	String	是	透传字段，查看第一页，则传空字符串。若需要翻页，需要将前一页返回值中的context透传到参数中。order用于指定翻页顺序。若order填0，则从当前页正序/往下翻页；若order填1，则从当前页倒序/往上翻页
data.dircount	String	是	子目录数量(总)
data.filecount	String	是	子文件数量(总)

参数名	类型	是否必然返回	参数描述
data.infos	Array	是	文件、目录集合，可以为空
data.infos.name	String	是	文件或目录名
data.infos.biz_attr	String	是	目录或文件属性，业务端维护
data.infos.ctime	String	是	目录或文件的创建时间，unix时间戳
data.infos.mtime	String	是	目录或文件的修改时间，unix时间戳
data.infos.filesize	Int	否(当类型为文件时返回)	文件大小
data.infos.filelen	Int	否(当类型为文件时返回)	文件已传输大小(通过与filesize对比可知文件传输进度)
data.infos.sha	String	否(当类型为文件时返回)	文件sha
data.infos.access_url	String	否(当类型为文件时返回)	生成的文件下载url

示例

```
var qcloud = require('qcloud_cos');

//如果conf.js中已经设置好APPID和SECRET_KEY则不需要此步骤。
qcloud.conf.setAppInfo('10000','xxxxxx','xxxxxx');

qcloud.cos.prefixSearch('bucketname', '/myFolder/', '20150606_', 20, 'eListBoth', 0, '', function(ret) {
  //deal with ret
});
```

文件操作

文件上传

接口说明：用于较小文件(一般小于8MB)的上传，可以通过此接口上传较小的文件并获得文件的url，较大的文件请使用分片上传接口。

方法原型

function upload(filePath, bucket, dstpath, bizattr, insertOnly, callback);

参数说明

参数名	类型	是否必填	默认值	参数描述
bucket	String	是	无	bucket名称，bucket创建参见创建Bucket
dstpath	String	是	无	文件在COS服务端的全路径，不包括/appid/bucketname
filepath	String	是	无	本地要上传文件的全路径
bizattr	String	否	空串	文件属性，业务端维护
insertOnly	Int	否	无	insertOnly==0表示允许覆盖文件 1表示不允许 其他值忽略
callback	function	否	输出返回结果	结构为function(ret){}的函数，ret为json结构，默认直接输出。

返回结果说明

参数名	类型	是否必然返回	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息
data	Array	是	返回数据
data.access_url	Bool	是	生成的文件下载url
data.url	String	是	操作文件的url
data.resource_path	String	是	资源路径. 格式:/appid/bucket/xxx

示例

```
var qcloud = require('qcloud_cos');

//如果conf.js中已经设置好APPID和SECRET_KEY则不需要此步骤。
qcloud.conf.setAppInfo('10000','xxxxxx','xxxxxx');

qcloud.cos.upload('test.mp4', 'bucketName', '/myFolder/mycos.txt', 1, function(ret) {
  //deal with ret
});
```

文件分片上传

接口说明：用于较大文件(一般大于8MB)的上传，可以通过此接口上传较大文件并获得文件的url和唯一标识resource_path（用于调用其他api）。

方法原型

```
function upload_slice(filePath, bucket, dstpath, bizattr, slice_size, session, insertOnly, callback);
```

参数说明

参数名	类型	是否必填	默认值	参数描述
bucket	String	是	无	bucket名称，bucket创建参见创建Bucket
dstpath	String	是	无	文件在COS服务端的全路径，不包括/appid/bucketname
filepath	String	是	无	本地要上传文件的全路径
bizattr	String	否	空串	文件属性，业务端维护
sliceSize	Int	否	512*1024字节	分片大小，用户可以

参数名	类型	是否必填	默认值	参数描述
				根据网络状况自行设置
session	String	否	空串	续传时透传的session，一般不设置。
insertOnly	Int	否	无	insertOnly==0 表示允许覆盖文件 1表示不允许 其他值忽略
callback	function	否	输出返回结果	结构为function(ret){}的函数，ret为json结构，默认直接输出。

返回结果说明

参数名	类型	是否必然返回	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息
data	Array	是	返回数据
data.access_url	Bool	是	生成的文件下载url
data.url	String	是	操作文件的url
data.resource_path	String	是	资源路径. 格式:/appid/bucket/xxx

示例

```
var qcloud = require('qcloud_cos');

//如果conf.js中已经设置好APPID和SECRET_KEY则不需要此步骤。
qcloud.conf.setAppInfo('10000','xxxxxx','xxxxxx');

qcloud.cos.upload_slice('test.mp4', 'bucketName', '/myFolder/mycos.txt', 'bizattr', 2*1024*1024, null,
1, function(ret) {
  //deal with ret
});
```

文件属性更新

接口说明：用于目录业务自定义属性的更新，可以通过此接口更新业务的自定义属性字段。

方法原型

function updateFile(bucket, path, bizattr, authority, custom_headers, callback);

参数说明

参数名	类型	是否必填	默认值	参数描述
bucket	String	是	无	bucket名称，bucket创建参见创建Bucket
path	String	是	无	文件在COS服务端的全路径，不包括/appid/bucketname
bizattr	String	是	无	待更新的文件属性信息
authority	String	否	无	eInvalid,eWRPrivate,eWPrivateRPublic,文件可以与bucket拥有不同的权限类型，已经设置过权限的文件如果想要撤销，直接赋值为eInvalid，则会采用bucket的权限
custom_headers	String	否	无	自定义header对象
callback	function	否	输出返回结果	结构为function(ret){}的函数，ret为json结构，默认直接输出。

自定义header对象说明

参数名	类型	是否必然返回	参数描述
Cache-Control	String	否	文件的缓存机制
Content-Type	String	否	文件的MIME信息
Content-Disposition	String	否	MIME协议的扩展
Content-Language	String	否	文件的语言
x-cos-meta-自定义内容	String	否	自定义信息

返回结果说明

参数名	类型	是否必然返回	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息

示例

```
var qcloud = require('qcloud_cos');

//如果conf.js中已经设置好APPID和SECRET_KEY则不需要此步骤。
qcloud.conf.setAppInfo('10000','xxxxxx','xxxxxx');

var headers = {
  "Cache-Control": "no-cache",
  "Content-Type" : "application/json"
};

qcloud.cos.updateFile('bucketName', '/myFolder/test.txt', 'bizattr', 'eWRPrivate', headers,
function(ret) {
  //deal with ret
});
```

文件查询

接口说明：用于文件的查询，可以通过此接口查询文件的各项属性信息。

方法原型

function statFile(bucket, path, callback);

参数说明

参数名	类型	是否必填	默认值	参数描述
bucket	String	是	无	bucket名称，bucket创建参见创建Bucket
path	String	是	无	文件在COS服务端的全路径，不包括/appid/bucketname
callback	function	否	输出返回结果	结构为function(ret){}的函数，ret为json结构，默认直接输出。

自定义header对象说明

参数名	类型	是否必然返回	参数描述
Cache-Control	String	否	文件的缓存机制
Content-Type	String	否	文件的MIME信息
Content-Disposition	String	否	MIME协议的扩展
Content-Language	String	否	文件的语言
x-cos-meta-自定义内容	String	否	自定义信息

返回结果说明

参数名	类型	是否必然返回	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息
data	Array	是	文件属性数据
data.name	String	是	文件或目录名
data.biz_attr	String	是	文件属性，业务端维护
data.ctime	String	是	文件的创建时间，unix时间戳

参数名	类型	是否必然返回	参数描述
data.mtime	String	是	文件的修改时间，unix时间戳
data.filesize	Int	是	文件大小
data.filelen	Int	是	文件已传输大小(通过与filesize对比可知文件传输进度)
data.sha	String	是	文件sha
data.access_url	String	是	生成的文件下载url
data.authority	String	否	无
data.custom_headers	String	否	无

示例

```
var qcloud = require('qcloud_cos');

//如果conf.js中已经设置好APPID和SECRET_KEY则不需要此步骤。
qcloud.conf.setAppInfo('10000','xxxxxx','xxxxxx');

qcloud.cos.statFile('bucketName', '/myFolder/test.txt', function(ret) {
  //deal with ret
});
```

文件删除

接口说明：用于文件的删除，可以通过此接口删除已经上传的文件。

方法原型

```
function deleteFile(bucket, path, callback);
```

参数说明

参数名	类型	是否必填	默认值	参数描述

参数名	类型	是否必填	默认值	参数描述
bucket	String	是	无	bucket名称，bucket创建参见创建Bucket
path	String	是	无	文件在COS服务端的全路径，不包括/appid/bucketname
callback	function	否	输出返回结果	结构为function(ret){}的函数，ret为json结构，默认直接输出。

返回结果说明

参数名	类型	是否必然返回	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息

示例

```
var qcloud = require('qcloud_cos');

//如果conf.js中已经设置好APPID和SECRET_KEY则不需要此步骤。
qcloud.conf.setAppInfo('10000','xxxxxx','xxxxxx');

qcloud.cos.deleteFile('bucketName', '/myFolder/test.txt', function(ret) {
  //deal with ret
});
```

文件移动

接口说明：用于文件的移动和重命名。

方法原型

```
function moveFile(bucket, path, destPath, overWrite, callback);
```

参数说明

参数名	类型	是否必填	默认值	参数描述
success	Function	是	无	参见通用参数success CallBack
error	Function	是	无	参见通用参数errorCa llBack
bucketName	String	是	无	参见通用参数bucket Name
path	String	是	无	文件在COS服务端的全路径，不包括/appid/bucketname
destPath	String	是	无	文件在COS服务端的全路径，不包括/appid/bucketname
overWrite	Int	否	无	是否覆盖同名文件, 0表示不覆盖 1表示覆盖 可选参数

返回结果说明

参数名	类型	必然返回	参数描述
code	Int	是	错误码，成功时为0
message	String	是	错误信息

示例

```
var qcloud = require('qcloud_cos');

//如果conf.js中已经设置好APPID和SECRET_KEY则不需要此步骤。
qcloud.conf.setAppInfo('10000','xxxxxx','xxxxxx');

qcloud_cos.cos.moveFile('0001', '123/test02.js', '/345/test01.js', 0, function(ret) {
  console.log('moveFile done');
  console.log(ret);
  console.log('############');
```

```
});
```

移动端 SDK

iOS SDK

开发准备

SDK 获取

对象存储服务的 iOS SDK 的下载地址：iOS SDK

更多示例可参考Demo：iOS Demo

开发准备

- iOS 5.0+；
- 手机必须要有网络（GPRS、3G或Wifi网络等）；
- 从控制台获取APP ID、SecretID、SecretKey，详情参考权限控制。

SDK 配置

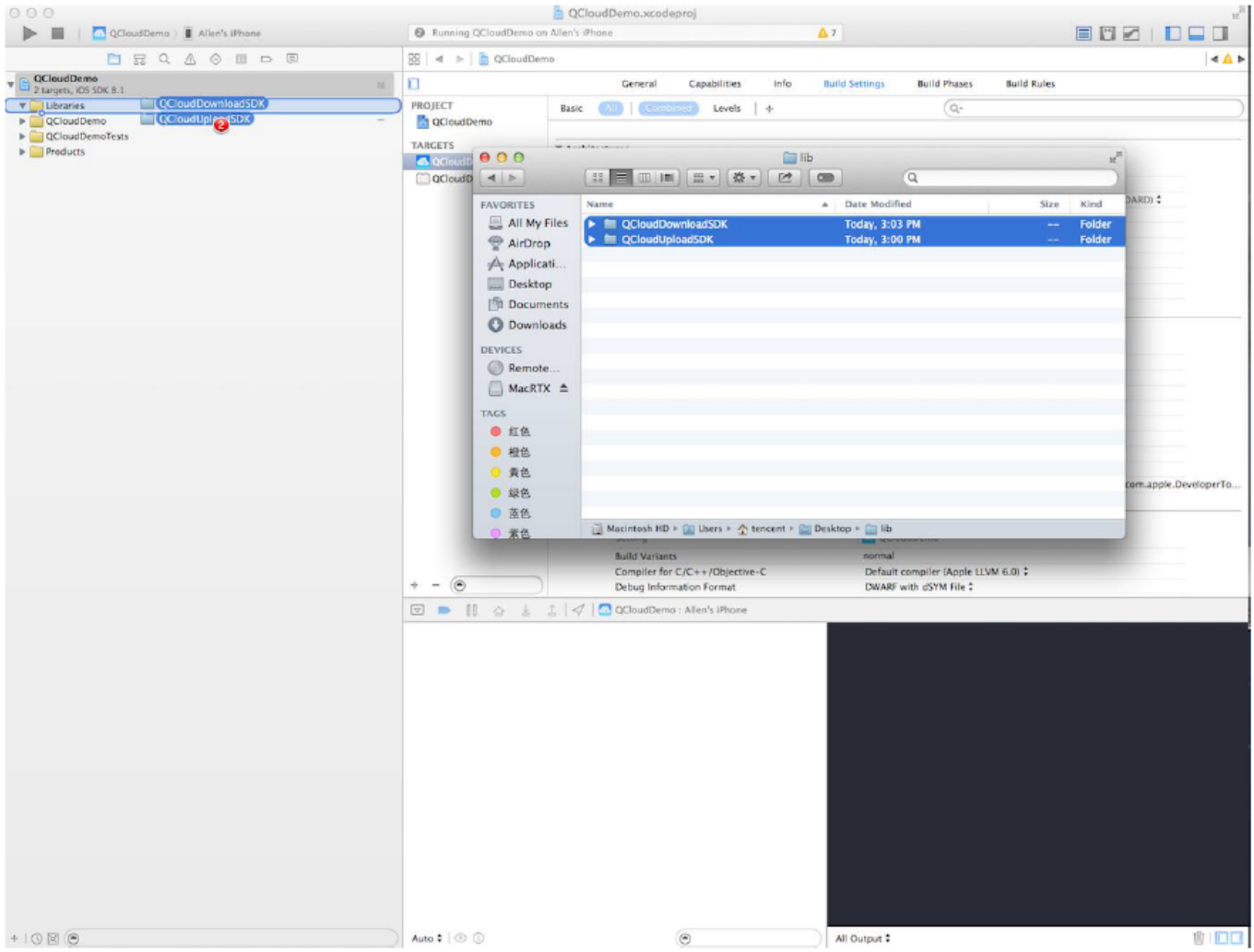
SDK 导入

COS 的 iOS SDK 由上传 SDK 和下载 SDK 两个压缩包组成：

- QCloudUploadSDK.zip
- QCloudDownloadSDK.zip

每个压缩包都包含了一个 .a 静态库和一个包含头文件的文件夹 Headers，如下图所示。上传包提供了支持 bitcode 版本，与不支持 bitcode 版本，可根据业务需要进行选择。

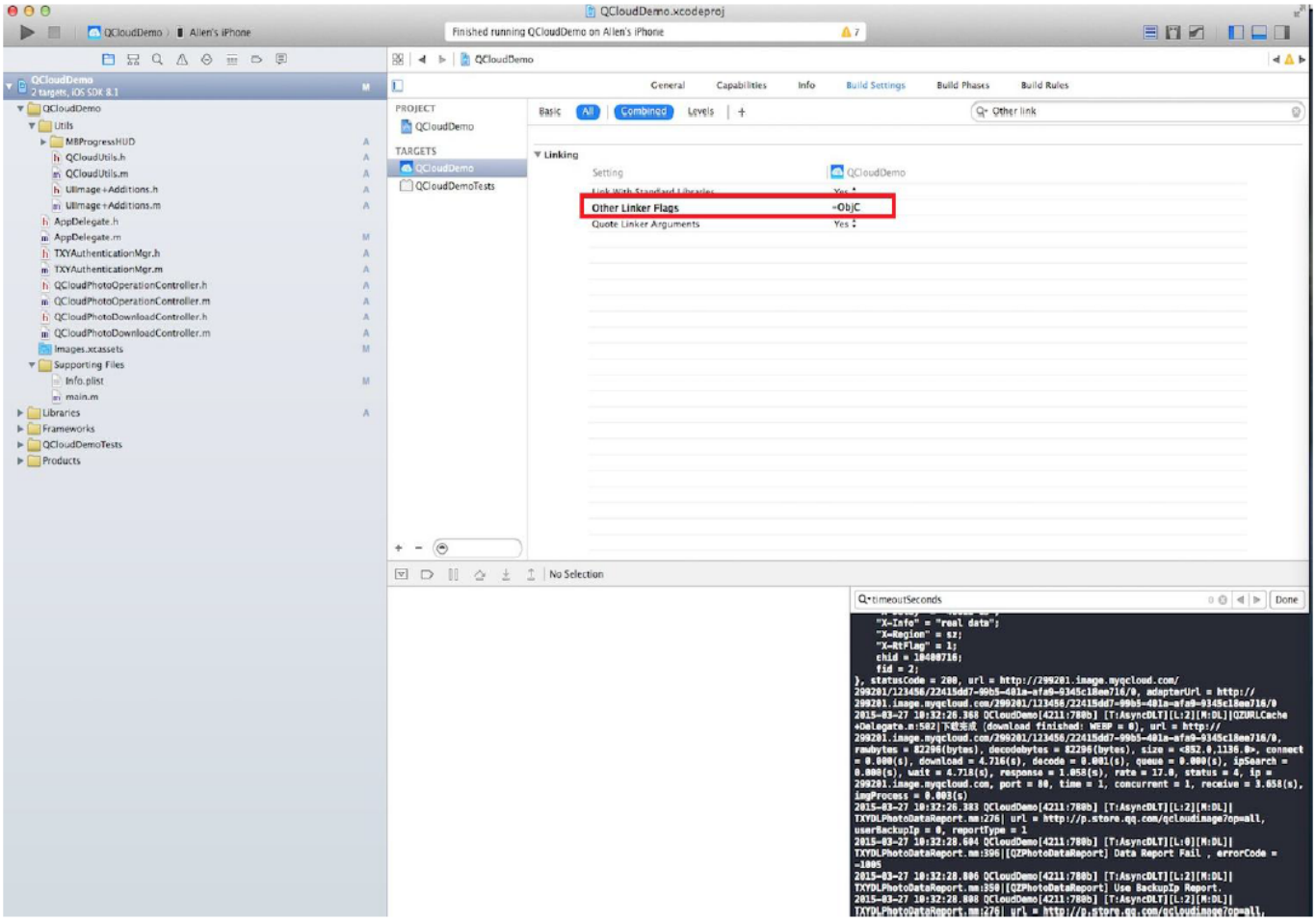
将解压后的 QCloudUploadSDK 和 QCloudDownloadSDK 拖入工程目录，Xcode 会自动将其加入链接库列表中。



注意：上传/下载SDK包可根据业务需求选择性导入。

工程配置

在 Build Settings 中设置 Other Linker Flags，加入参数 -ObjC。



依赖库添加

在 build Phases -> Link Binary With Libraries 中加入下列依赖库：

- SystemConfiguration.framework
- CoreTelephony.framework

若需要使用上传SDK ，还需加入以下依赖库：

- libstdc++.6.dylib

若需要使用下载 SDK ，还需加入以下依赖库：

- MobileCoreServices.framework
- libxml2.dylib
- libz.dylib

签名获取

签名类型：

类型	含义
多次有效	有效时间内多次始终有效
单次有效	与资源URL绑定，一次有效

签名获取：

移动端 SDK 中用到的签名，推荐使用 服务器端SDK，并由移动端向业务服务器请求。SIGN 的具体生成和使用请参照 访问权限 。

目录操作

初始化

引入上传 SDK 的头文件 TXYUploadManager.h，使用目录操作时，需要先实例化 TXYUploadManager 对象。

方法原型

```
- (instancetype)initWithCloudType:(TXYCloudType)cloudType
  persistenceId:(NSString *)persistenceId
  appId:(NSString*)appId;
```

参数说明

参数名称	类型	是否必填	说明
cloudType	TXYCloudType	是	业务类型，COS服务设置为：TXYCloudTypeForFile
persistenceId	NSString *	是	TXYUploadManager 实例对应的持久化 id，此

参数名称	类型	是否必填	说明
			id 必须全局唯一，persistenceId 为 nil 时，上传任务不持久化
appId	NSString *	是	项目ID，即APP ID。

示例

```
TXYUploadManager *uploadManager = [[TXYUploadManager alloc]
  initWithCloudType(TXYCloudTypeForFile
  persistenceId:@"qcloudobject"
  appId:@"10000002")];
```

目录创建

通过此接口在指定的 bucket 下创建目录，具体步骤如下：

1. 实例化 TXYCreateDir 对象；
2. 调用 TXYUploadManager 的 sendCommand 方法，将 TXYCreateDir 对象传入。
3. 通过TXYCreateDirCommandRsp的对象返回结果

方法原型

```
- (instancetype) initWithPath:(NSString*)path
  bucket:(NSString*)bucket
  sign:(NSString*)sign
  needOverWrite:(BOOL)overwrite
  customAttribute:(NSString*)attrs;
```

参数说明

参数名称	类型	是否必填	说明
path	NSString *	是	目录路径（相对于bucket的路径）

参数名称	类型	是否必填	说明
bucket	NSString *	是	目录所属 bucket 名称
sign	NSString *	是	签名
overwrite	BOOL	是	是否覆盖相同名字的目录或文件
attrs	NSString *	否	用户自定义属性

返回结果说明

通过TXYCreateDirCommandRsp的对象返回结果

属性名称	类型	说明
overwrite	BOOL	文件是否被覆盖
ctime	NSInteger	文件的创建时间
accessUrl	NSString *	文件的访问url

示例

```
TXYCreateDir *createDirCommand = [[TXYCreateDir alloc] initWithPath(path:DIRPATH
  bucket:BUCKETNAME
  sing:SIGN
  overwrite:YES
  attrs:ATTRS)];

[uploadManager sendCommand:createDirCommand
  complete:^(TXYCreateDirCommandRsp *resp) {
  if (resp.retCode >= 0 ) {
  // 创建成功
  }
  else {
  // 创建失败
  }
  }];
```

目录属性更新

通过调用此接口更新目录的自定义属性，具体步骤如下：

1. 实例化 TXYUpdate 对象；
2. 调用 TXYUploadManager 的 sendCommand 方法，将 TXYUpdate 对象传入；
3. 通过TXYUpdateCommandRsp 对象返回结果

方法原型

```
- (instancetype) initWithPath:(NSString*)path
  bucket:(NSString*)bucket
  sign:(NSString*)sign
  objectType:(TXYObjectType)objectType
  authorityType:(TXYAuthorityType)fileAuthorityType
  customAttribute:(NSString*)attrs;
```

参数说明

参数名称	类型	是否必填	说明
path	NSString *	是	目录路径（相对于bucket的路径）
bucket	NSString *	是	目录所属 bucket 名称
sign	NSString *	是	签名
objectType	TXYObjectType	是	业务类型，目录属性更新时设置为：TXYObjectDir
fileAuthorityType	TXYAuthorityType	是	文件权限类型，如设置文件权限和bucket相同：eInvalidAuth
attrs	NSString *	否	用户自定义属性

返回结果说明

通过TXYUpdateCommandRsp 对象返回结果

属性名称	类型	说明
retCode	int	任务描述代码，为retCode >= 0时标示成功，为负数表示为失败
descMsg	NSString *	任务描述信息

示例

```
TXYUpdate *updateDirCommand = [[TXYUpdate alloc] initWithPath:DIRPATH
  bucket:BUCKETNAME
  sign:SIGN
  objectType:TXYObjectDir
  authorityType:eInvalidAuth
  customAttribute:ATTRS];

[uploadManager sendCommand: updateDirCommand
  compelete:^(TXYUpdateCommandRsp *resp) {
  if(resp.retCode >= 0) {
  // 更新成功
  }
  else {
  // 更新失败
  }
  }];
```

目录属性查询

通过调用此接口来查询目录的详细属性，具体步骤如下：

1.实例化 TXYStat 对象；
2.调用 TXYUploadManager 的 sendCommand 方法，将 TXYUpdate 对象传入；
3.通过TXYStatCommandRsp的对象返回结果信息；

方法原型

```
-(instancetype) initWithPath:(NSString*)path
  bucket:(NSString*)bucket
  sign:(NSString*)sign
  objectType:(TXYObjectType)objectType;
```

参数说明

参数名称	类型	是否必填	说明
path	NSString *	是	目录路径（相对于bucket的路径）
bucket	NSString *	是	目录所属 bucket 名称
sign	NSString *	是	签名
objectType	TXYObjectType	是	文件属性查询是，设置为：TXYObjectDir

返回结果说明

通过TXYStatCommandRsp的对象返回结果信息

属性名称	类型	说明
fileDirInfo	TXYFileDirInfo *	单个文件目录信息
retCode	int	任务描述代码，为retCode >= 0时标示成功，为负数表示为失败
descMsg	NSString *	任务描述信息

示例

```
TXYStat *statCommand = [[TXYStat alloc] initWithPath:DIRPATH
  bucket:BUCKETNAME
  sign:SIGN
  objectType:TXYObjectDir];

[uploadManager sendCommand: statCommand
  compelete:^(TXYStatCommandRsp *resp){
  if(resp.retCode >= 0){
```

```
// 查询成功
}
else{
// 查询失败
}
}];
```

目录删除

调用此接口，进行指定 bucket
下目录的删除，如果目录中存在有效文件或目录，将不能删除。具体步骤如下：

1. 实例化 TXYDelete 对象；
2. 调用 TXYUploadManager 的 SendCommand 命令，传入 TXYDelete 对象；
3. 通过TXYTaskRsp的对象返回结果信息

方法原型

```
- (instancetype) initWithPath:(NSString*)path
  bucket:(NSString*)bucket
  sign:(NSString*)sign
  objectType:(TXYObjectType)objectType;
```

参数说明

参数名称	类型	是否必填	说明
path	NSString *	是	目录路径（相对于bucket的路径）
bucket	NSString *	是	目录所属 bucket 名称
sign	NSString *	是	签名
objectType	TXYObjectType	是	业务类型，目录删除时设置为：TXYObjectDir

返回结果说明

通过TXYTaskRsp的对象返回结果信息

属性名称	类型	说明
retCode	int	任务描述代码，为retCode >= 0时标示成功，为负数表示为失败
descMsg	NSString *	任务描述信息

示例

```
TXYDelete *deleteCommand = [[TXYDelete alloc] initWithPath:DIRPATH
  bucket:BUCKETNAME
  sign:SIGN
  objectType:TXYObjectDir];

[uploadManager sendCommand: deleteCommand
  compelete:^(TXYTaskRsp *resp) {
  if(resp.retCode >= 0) {
  // 删除成功
  }
  else {
  // 删除失败
  }
  }];
```

列举目录中的文件和目录

调用此接口可以列出 bucket 中，指定目录下的文件、目录信息，具体步骤如下：

1. 实例化 TXYListDir 对象；
2. 调用 TXYUploadManager 对象的 sendCommand 方法，将 TXYListDir 对象传入;
3. 通过TXYListDirCommandRsp的对象返回结果信息

方法原型

```
- (instancetype) initWithPath:(NSString*)path
  bucket:(NSString*)bucket
  sign:(NSString*)sign
  number:(NSUInteger)num
  listPattern:(TXYListPattern)pattern
  order:(BOOL)order
  pageContext:(NSString*)context;
```

参数说明

参数名称	类型	是否必填	说明
path	NSString *	是	目录路径（相对于bucket的路径）
bucket	NSString *	是	目录所属 bucket 名称
sign	NSString *	是	签名
num	NSUInteger	是	一次拉取数目设定
pattern	TXYListPattern	是	拉取设定：TXYListBoth - 目录和文件都拉取，TXYListDirOnly - 只拉取目录，TXYListFileOnly - 只拉取文件
order	BOOL	是	排序设定，0 - 正序 1-反序
context	NSString*	否	分页浏览的上下文，第一次拉取时可以为空，后续拉取分页内容时必须填充

返回结果说明

通过TXYListDirCommandRsp的对象返回结果信息

属性名称	类型	说明
dirCount	NSUInteger	目录个数
fileCount	NSUInteger	文件个数

属性名称	类型	说明
fileDirInfoList	NSArray *	文件目录属性列表
pageContext	NSString *	分页浏览目录的上下文，后台返回
hasMore	BOOL	有无下一页数据

示例

```
// 第一次拉取目录时，context 为空
TXYListDir *listDirCommand = [[TXYListDir alloc] initWithPath:DIRPATH
  bucket:BUCKETNAME
  sign:SIGN
  num:10
  pattern:TXYListBoth
  order:YES
  context:nil];

[uploadManager sendCommand: listDirCommand
  compelete:^(TXYListDirCommandRsp *resp) {
  if(resp.retCode >= 0) {
  // 拉取成功
  }
  else {
  // 拉取失败
  }
  }];
```

列举目录下制定前缀文件&目录

通过此接口查询目录下，指定前缀的文件和目录，具体步骤如下：

1. 实例化 TXYSearch 对象；
2. 调用 TXYUploadManager 对象的 sendCommand 方法，将 TXYSearch 对象传入；
3. 通过TXYSearchCommandRsp的对象返回结果信息

方法原型

```
- (instancetype) initWithPath:(NSString*)path
  bucket:(NSString*)bucket
  sign:(NSString*)sign
  number:(NSUInteger)num
  pageContext:(NSString*)context;
```

参数说明

参数名称	类型	是否必填	说明
path	NSString *	是	目录路径，前缀查询
bucket	NSString *	是	目录所属 bucket 名称
sign	NSString *	是	签名
num	NSUInteger	是	一次拉取数目设定
context	NSString*	否	分页浏览的上下文，第一次拉取时可以为空，后续拉取分页内容时必须填充

返回结果说明

通过TXYSearchCommandRsp的对象返回结果信息

属性名称	类型	说明
dirCount	NSUInteger	目录个数
fileCount	NSUInteger	文件个数
fileDirInfoList	NSArray *	文件目录属性列表
pageContext	NSString *	分页浏览目录的上下文，后台返回
hasMore	BOOL	有无下一页数据

示例

```
// 第一次拉取目录时，context 为空
TXYSearch *listDirCommand = [[TXYSearch alloc] initWithPath:DIRPATH
  bucket:BUCKETNAME
```

```
 sign:SIGN
 num:10
 context:nil];

[uploadManager sendCommand: listDirCommand
 compelete:^(TXYSearchCommandRsp *resp) {
 if(resp.retCode >= 0) {
 // 拉取成功
 }
 else {
 // 拉取失败
 }
 }];
```

文件操作

初始化

与目录操作相同，在进行文件操作之前，需引入上传 SDK 的头文件 TXYUploadManager.h，实例化 TXYUploadManager 对象。

方法原型

```
- (instancetype)initWithCloudType:(TXYCloudType)cloudType
 persistenceId:(NSString *)persistenceId
 appId:(NSString*)appId;
```

参数说明

参数名称	类型	是否必填	说明
cloudType	TXYCloudType	是	业务类型，COS服务设置为：TXYCloudTypeForFile
persistenceId	NSString *	是	TXYUploadManager 实例

参数名称	类型	是否必填	说明
			对应的持久化id，此id必须全局唯一，persistenceId为nil时，上传任务不持久化
appId	NSString *	是	项目ID，即APP ID。

示例

```
TXYUploadManager *uploadManager = [[TXYUploadManager alloc]
  initWithCloudType:TXYCloudTypeForFile
  persistenceId:@"qcloudobject"
  appId:@"10000002"];
```

文件上传

调用此接口者可进行本地文件上传操作，具体步骤如下：

1. 实例化 TXYFileUploadTask对象；
2. 调用 TXYUploadManager 对象的 upload 方法，将 TXYFileUploadTask 对象传入；
3. 通过TXYFileUploadTaskRsp的对象返回结果信息

方法原型

```
- (instancetype)initWithPath:(NSString *)filePath
  sign:(NSString*)sign
  bucket:(NSString *)bucket
  fileName:(NSString *)fileName
  customAttribute:(NSString *)attrs
  uploadDirectory:(NSString*)directory
  insertOnly:(BOOL)inser;
```

参数说明

参数名称	类型	是否必填	说明

参数名称	类型	是否必填	说明
filePath	NSString *	是	文件路径
sign	NSString *	是	签名
bucket	NSString *	是	目标 Bucket 名称
fileName	NSString *	是	目标 文件上传cos后显示的名称
attrs	NSString *	否	文件自定义属性
directory	NSString *	是	文件上传目录，相对路径,举例“/path”
insertOnly	BOOL	是	上传文件的动作是插入覆盖，举例“YES” 文件不会覆盖之前的上传的文件

返回结果说明

通过TXYFileUploadTaskRsp的对象返回结果信息

属性名称	类型	说明
retCode	int	任务描述代码，为retCode >= 0时标示成功，为负数表示为失败
descMsg	NSString *	任务描述信息
fileURL	NSString *	成功后，后台返回文件的 CDN url
fileId	NSString *	成功后，后台返回文件的key（cos业务没有fileid）
sourceURL	NSString *	成功后，后台返回文件的 源站 url

示例

```
TXYFileUploadTask *fileTask = [[TXYFileUploadTask alloc] initWithPath:FILEPATH
  sign:SIGN
  bucket:BUCKETNAME
  fileName:fileName
  customAttribute:nil
  uploadDirectory:(NSString*)directory
  insertOnly:YES];
```

```
[uploadManager upload:fileTask
  complete:^(TXYTaskRsp *resp, NSDictionary *context) {
  fileResp = (TXYFileUploadTaskRsp *)resp;
  if(fileResp.retCode >= 0) {
  //上传成功
  }
  else {
  //上传失败
  }
  }
  process:^(int64_t totalSize, int64_t sendSize, NSDictionary *context) {
  // 上传进度
  }
  stateChange:^(TXYUploadTaskStat state, NSDictionary *context) {
  // 上传状态变化
  }];
```

文件属性更新

调用此接口更新文件的自定义属性，具体步骤如下：

1. 实例化 TXYUpdate 对象；
2. 调用 TXYUploadManager 的 SendCommand 命令，传入 TXYUpdate 对象；
3. 通过TXYUpdateCommandRsp的对象返回结果信息

方法原型

```
- (instancetype) initWithPath:(NSString*)path
  bucket:(NSString*)bucket
  sign:(NSString*)sign
  objectType:(TXYObjectType)objectType
  authorityType:(TXYAuthorityType)fileAuthorityType
  customAttribute:(NSString*)attrs;
```

参数说明

参数名称	类型	是否必填	说明
path	NSString *	是	目录路径（相对于bucket的路径）
bucket	NSString *	是	目录所属 bucket 名称
sign	NSString *	是	签名
objectType	TXYObjectType	是	业务类型，目录属性更新时设置为：TXYObjectDir
fileAuthorityType	TXYAuthorityType	是	文件权限类型
attrs	NSString *	否	用户自定义属性

返回结果说明

通过TXYUpdateCommandRsp的对象返回结果信息

属性名称	类型	说明
retCode	int	任务描述代码，为retCode >= 0时标示成功，为负数表示为失败
descMsg	NSString *	任务描述信息

示例

```
TXYUpdate *updateFileCommand = [[TXYUpdate alloc] initWithPath:FILEPATH
  bucket:BUCKETNAME
  sign:SIGN
  objectType:TXYObjectFile
  authorityType:nil
  customAttribute: ATTRS];

[uploadManager sendCommand: updateFileCommand
  compelete:^(TXYUpdateCommandRsp *resp){
  if(resp.retCode >= 0){
  // 更新成功
  }
  else{
```

```
// 更新失败
}
}];
```

文件属性查询

调用此接口可查询文件的属性信息，具体步骤如下：

1. 实例化 TXYStat 对象；
2. 调用 TXYUploadManager 的 SendCommand 命令，传入 TXYStat 对象；
3. 通过TXYStatCommandRsp类返回结果信息

方法原型

```
- (instancetype) initWithPath:(NSString*)path
  bucket:(NSString*)bucket
  sign:(NSString*)sign
  objectType:(TXYObjectType)objectType;
```

参数说明

参数名称	类型	是否必填	说明
path	NSString *	是	文件路径
bucket	NSString *	是	文件所属 bucket 名称
sign	NSString *	是	签名
objectType	TXYObjectType	是	文件属性查询是，设置为：TXYObjectFile

返回结果说明

通过TXYStatCommandRsp类返回结果信息

属性名称	类型	说明

属性名称	类型	说明
retCode	int	任务描述代码，为retCode >= 0时标示成功，为负数表示为失败
descMsg	NSString *	任务描述信息
fileInfo	TXYFileInfo *	成功时，文件基本信息

示例

```
TXYStat *statCommand = [[TXYStat alloc]initWithPath:FILEPATH
  bucket:BUCKETNAME
  sign:SIGN
  objectType:TXYObjectFile];

[uploadManager sendCommand: statCommand
  compelete:^(TXYStatCommandRsp *resp){
  if(resp.retCode >= 0){
  // 查询成功
  }
  else{
  // 查询失败
  }
  }];
```

文件删除

调用此接口进行文件的删除操作，具体步骤如下：

1. 实例化 TXYDelete 对象；
2. 调用 TXYUploadManager 的 SendCommand 命令，传入 TXYDelete 对象。
3. 通过TXYTaskRsp的对象返回结果信息

方法原型

```
- (instancetype) initWithPath:(NSString*)path
```

```
bucket:(NSString*)bucket
sign:(NSString*)sign
objectType:(TXYObjectType)objectType;
```

参数说明

参数名称	类型	是否必填	说明
path	NSString *	是	要删除的文件的路径
bucket	NSString *	是	文件所属 Bucket 名称
sign	NSString *	是	签名
objectType	TXYObjectType	是	业务类型，文件删除时设置为：TXYObjectFile

返回结果说明

通过TXYTaskRsp的对象返回结果信息

属性名称	类型	说明
retCode	int	任务描述代码，为retCode >= 0时标示成功，为负数表示为失败
descMsg	NSString *	任务描述信息

示例

```
TXYDelete *deleteCommand = [[TXYDelete alloc]initWithPath:FILEPATH
  bucket:BUCKETNAME
  sign:SIGN
  objectType:TXYObjectFile];

[uploadManager sendCommand: deleteCommand
  compelete:^(TXYTaskRsp *resp){
  if(resp.retCode >= 0){
  // 删除成功
  }
  else{
```

```
// 删除失败
}
}];
```

对象下载

初始化

在下载文件前，需要先实例化下载管理类 TXYDownloader，并对下载器的一些属性进行配置。

方法原型

```
- (instancetype)initWithPersistenceId:(NSString *)persistenceId
  type:(TXYDownloadType)type;
```

参数说明

参数名称	类型	是否必填	说明
persistenceId	NSString *	是	persistenceId 值不同表示缓存目录不同，为nil内部会自动创建一个缓存目录统一管理
type	TXYDownloadType	是	下载业务类型，COS服务设置为：TXYDownloadTypeFile

示例

```
TXYDownloader *downloadManager = [[TXYDownloader alloc]
  initWithPersistenceId:@"qcloudobject"
  type:TXYDownloadTypeFile];
```

下载并发数设置：可以指定下载器最大并发数。

方法原型：

```
- (void)setMaxConcurrent:(int)count;
```

示例：

```
[downloadManager setMaxConcurrent:10];
```

长连接/断点续传设置：可以设定是否开启长连接和端点续传功能。

方法原型：

```
// enable - YES 表示支持断点续传， No 表示不支持断点续传
- (void)enableHTTPRange:(BOOL)enable;
// enable - YES 表示支持 HTTP 长连接， No 表示不支持
- (void)enableKeepAlive:(BOOL)enable;
```

示例：

```
[downloadManager enableHTTPRange:YES];
[downloadManager enableKeepAlive:YES];
```

下载器使用

对已知 URL 的资源进行下载，异步模式进行。

方法原型

```
- (void)download:(NSString *)url
  target:(id)target
  succBlock:(void (^)(NSString *url, NSData *data, NSDictionary *info))succBlock
```

```
failBlock:(void (^)(NSString *url, NSError *error))failBlock
progressBlock:(void (^)(NSString *url, NSNumber *value))progressBlock
param:(NSDictionary *)param;
```

参数说明

参数名称	说明
url	资源地址
target	通知的对象
succBlock	成功通知
failBlock	失败通知
progressBlock	进度通知，当前下载百分比
param	可以指定 TXYDownloaderParam 族一系列参数,也可以用于透传使用者的参数

返回结果说明

参数名称	说明
url	资源地址
data	文件数据
info	资源相关信息
error	失败信息
value	进度通知，当前下载进度

示例

```
[downloadManager download:URL
 target:self
 succBlock:^(NSString *url, NSData *data, NSDictionary *info) {
 // 下载成功
 }
 failBlock:^(NSString *url, NSError *error) {
 // 下载失败
 }
 progressBlock:^(NSString *url, NSNumber *value) {
```

```
// 下载进度
}
param:nil];
```

取消下载

取消对指定URL资源的下载任务，或取消所有下载请求。

方法原型

```
// 取消 target 对应的下载请求，url 为资源地址
- (void)cancel:(NSString *)url target:(id)target;
// 取消所有的下载请求
- (void)cancelAll;
```

示例

```
[downloadManager cancel:URL target:self];
[downloadManager cancelAll];
```

Android SDK

开发准备

SDK 获取

对象存储服务的 Android SDK 的下载地址：Android SDK。

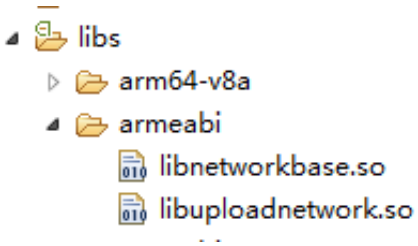
更多示例可参考Demo：Android SDK Demo。

开发准备

1. SDK 支持 Android 2.2 及以上版本的手机系统；
2. 手机必须要有网络（GPRS、3G或 WIFI 网络等）；
3. 手机可以没有存储空间，但会使部分功能无法正常工作；
4. 从控制台获取APP ID、SecretID、SecretKey，详情参考权限控制。

SDK 配置

解压 SDK 包，将其中的 libs 目录合并到本地工程libs目录：



注意：如果工程中存在 armeabi-v7a/armeabi-v8a目录，需将上述.so 文件拷贝一份到此目录下，否则由于 android 系统的问题在安装 apk 后会找不到 so 。

注意：若项目需要兼容 x86 ，则将上述 so 复制一份放入 x86 目录即可。

配置工程导入下列 jar 包：

- upload.jar

- download.jar
- wup-1.0.0-SNAPSHOT.jar

注意：可根据业务功能所需，对 upload/download 包进行选择性导入。

SDK 需要网络访问相关的一些权限，需要在 AndroidManifest.xml 中增加如下权限声明：

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

生成签名

签名类型：

类型	含义
多次有效	有效时间内多次始终有效
单次有效	与资源URL绑定，一次有效

签名获取：

SDK 中用到的 SIGN，推荐使用 服务器端SDK，并由移动端向业务服务器请求。SIGN 的具体生成和使用请参照 访问权限 。

目录操作

初始化

进行目录操作之前需要先实例化 UploadManager 对象。

方法原型

```
public UploadManager(Context context, String appId, FileType fileType,
  String persistenceId)
```

参数说明

参数名称	类型	是否必填	说明
context	Context	是	
appId	String	是	腾讯云APP ID
fileType	FileType	是	业务类型，COS服务指明为：FileType.File
persistenceId	String	否	持久化 ID，每个 UploadManager 需设置一个唯一的 ID 用于持久化保存未完成任务列表，以便应用退出重进后能够继续进行上传；传入为 Null，则不会进行持久化保存

示例

```
import com.tencent.upload.UploadManager;
import com.tencent.upload.Const.FileType;

UploadManager fileUploadMgr = null;
fileUploadMgr = new UploadManager(this, "10000002", FileType.File, "qcloudobject");
```

目录创建

调用此接口可在指定的 bucket 下创建目录，具体步骤如下：

1. 实例化 DirCreateTask 对象；
2. 调用 UploadManager 的 sendCommand 方法，传入 DirCreateTask 对象。
3. 目录创建成功后，DirCreateTask 对象回调监听器的 onSuccess()方法，返回 DirCreateTask.CmdTaskRsp 对象。

方法原型

public DirCreateTask (FileType fileType, String bucket, String path,
String bizAttr, IListener listener)

参数说明

参数名称	类型	是否必填	说明
fileType	FileType	是	业务类型，COS 服务设置为：FileType.File
bucket	String	是	目录所属bucket 名称
path	String	是	需要创建目录的路径
bizAttr	String	否	目录绑定的属性信息，由业务维护
listener	IListener	是	结果回调

返回结果说明

通过DirCreateTask.CmdTaskRsp对象的成员变量返回请求结果。

成员变量名称	类型	变量说明
path	String	创建的目录路径
ctime	String	创建时间
accessUrl	String	目录的URL
ret	int	结果码
msg	String	详细结果信息

示例

import com.tencent.upload.task.impl.DirCreateTask;

```
DirCreateTask filetask = null;
filetask = new DirCreateTask(FileType.File, BUCKETNAME, DIRPATH, ATTRS, new
DirCreateTask.IListener() {
  @Override
  public void onSuccess(final DirCreateTask.CmdTaskRsp result) {
  // 创建成功
  }
  @Override
  public void onFailure(final int ret, final String msg) {
  // 创建失败
  }
});

filetask.setAuth(SIGN);
fileUploadMgr.sendCommand(filetask);
```

目录属性更新

通过此接口更新业务的自定义属性字段，具体步骤如下：

1. 实例化 ObjectUpdateTask 对象；
2. 调用 UploadManager 的 sendCommand 方法，传入 ObjectUpdateTask 对象。
3. 目录更新成功后，ObjectUpdateTask 对象回调监听器的 onSuccess()方法，返回 ObjectUpdateTask.CmdTaskRsp 对象。

方法原型

```
public ObjectUpdateTask(FileType fileType, String bucket, String path, int type, String attr, int eauth,
HashMap<String,String> custom_headers, IListener listener)
```

参数说明

参数名称	类型	是否必填	说明
fileType	FileType	是	业务类型，COS 服务统一写为 FileType.File
bucket	String	是	目录所属bucket 名称
path	String	是	需要更新的目录路径
attr	String	否	新的目录绑定的属性信息
type	String	是	对象类型，目录更新设置为：Dentry.DIR
eauth	int	否	文件的读写权限：与bucket拥有相同的权限（0），私有读写权限（1），私有写公有读权限（2）
cust_headers	HashMap	否	文件自定义的header
listener	IListener	是	结果回调

返回结果说明

通过ObjectUpdateTask.CmdTaskRsp对象的成员变量返回请求结果。

成员变量名称	类型	变量说明
ret	int	结果码
msg	String	详细结果信息

示例

```
import com.tencent.upload.task.impl.ObjectUpdateTask;

ObjectUpdateTask filetask = null;
Filetask = new ObjectUpdateTask(FileType.File, BUCKETNAME, FILEPATH, Dentry.FILE, ATTR,
EAUTH,CUSTOM_HEANDERS,new ObjectUpdateTask.IListener() {
  @Override
  public void onSuccess(final CmdTaskRsp result) {
  // TODO Auto-generated method stub
  // 更新成功

  }
```

```
  @Override
  public void onFailure(final int ret, final String msg){
  // TODO Auto-generated method stub
  // 更新失败
  }
  });
filetask.setAuth(SIGN);
fileUploadMgr.sendCommand(filetask);
```

目录属性查询

通过此接口来查询目录的详细属性，具体步骤如下：

1. 实例化 ObjectStatTask 对象；
2. 调用 UploadManager 的 sendCommand 方法，传入 ObjectStatTask 对象。
3. 目录属性查询成功后，ObjectStatTask 对象回调监听器的 onSuccess()方法，返回 ObjectStatTask.CmdTaskRsp 对象。

方法原型

```
public ObjectStatTask(FileType fileType, String bucket, String path,
  int type, IListener listener);
```

参数说明

参数名称	类型	是否必填	说明
fileType	FileType	是	业务类型，COS 服务设置为：FileType.File
bucket	String	是	Bucket 名称
path	String	是	需要查询的目录路径
type	String	是	对象类型，目录属性查询时设置为：Dentry.DIR

参数名称	类型	是否必填	说明
listener	IListener	是	结果回调

返回结果说明

通过ObjectStatTask.CmdTaskRsp对象的成员变量返回请求结果。

成员变量名称	类型	变量说明
inode	Dentry	目录的详细属性
ret	int	结果码
msg	String	详细结果信息

示例

```
import com.tencent.upload.task.impl.ObjectStatTask;

ObjectStatTask filetask = null;
filetask = new ObjectStatTask(FileType.File, BUCKETNAME, DIRPATH, Dentry.DIR,
new ObjectStatTask.IListener() {
  @Override
  public void onSuccess(final ObjectStatTask.CmdTaskRsp result) {
  // 查询成功
  }

  @Override
  public void onFailure(final int ret, final String msg) {
  // 查询失败
  }
});

filetask.setAuth(SIGN);
fileUploadMgr.sendCommand(filetask);
```

目录删除

调用此接口，进行指定 bucket
下目录的删除，如果目录中存在有效文件或目录，将不能删除。具体步骤如下：

1. 实例化 ObjectDeleteTask 对象；
2. 调用 UploadManager 的 sendCommand 方法，传入 ObjectDeleteTask 对象。
3. 目录删除成功后，ObjectDeleteTask 对象回调监听器的 onSuccess()方法，返回 ObjectDeleteTask.CmdTaskRsp 对象。

方法原型

```
public ObjectDeleteTask(FileType fileType, String bucket, String path,
  int type, IListener listener)
```

参数说明

参数名称	类型	是否必填	说明
fileType	FileType	是	业务类型，COS 服务设置为： FileType.File
bucket	String	是	目录所属 bucket 名称
path	String	是	要删除的目录路径
type	int	是	对象类型，删除目录时设置为：Dentry.DIR
listener	IListener	是	结果回调

返回结果说明

通过ObjectDeleteTask.CmdTaskRsp对象的成员变量返回请求结果。

成员变量名称	类型	变量说明
ret	int	结果码
msg	String	详细结果信息

示例

```
import com.tencent.upload.task.impl.ObjectDeleteTask;
```

```
ObjectDeleteTask filetask= null;
filetask = new ObjectDeleteTask(FileType.File, BUCKETNAME, DIRPATH, Dentry.DIR, new
ObjectDeleteTask.IListener() {
  @Override
  public void onSuccess(ObjectDeleteTask.CmdTaskRsp result) {
  // 删除成功
  }

  @Override
  public void onFailure(final int ret, final String msg) {
  // 删除失败
  }
});

filetask.setAuth(SIGN);
fileUploadMgr.sendCommand(filetask);
```

列举目录中的文件和目录

通过此接口查询目录下的文件、目录信息，具体步骤如下：

1. 实例化 DirListTask 对象；
2. 调用 UploadManager 的 sendCommand 方法，传入 DirListTask 对象。
3. 查询信息成功后，DirListTask 对象回调监听器的 onSuccess()方法，返回 DirListTask.CmdTaskRsp 对象。

方法原型

```
public DirListTask (FileType fileType, String bucket, String path, int num,
  int pattern, boolean order, String content, IListener listener)
```

参数说明

参数名称	类型	是否必填	说明
fileType	FileType	是	业务类型，COS 服务设置为： FileType.File
bucket	String	是	目录所属 bucket 名称
path	String	是	请求的目录路径
num	int	是	拉取记录数目
pattern	int	是	拉取方式，Dentry.ListBoth - 目录和文件都拉取，Dentry.ListDirOnly - 只拉取目录，Dentry.ListFileOnly - 只拉取文件
order	boolean	是	true - 反序，false - 正序
content	String	否	首次拉取必须清空，拉取下一页是需要传入上一次拉取时返回的content
listener	IListener	是	结果回调

返回结果说明

通过DirListTask.CmdTaskRsp对象的成员变量返回请求结果。

成员变量名称	类型	变量说明
dirCount	long	当前层子目录个数
fileCount	long	当前层子文件个数
hasMore	boolean	是否有更多
content	String	透传字段
inodes	ArrayList	目录下文件的详细属性
ret	int	结果码
msg	String	详细结果信息

示例

```
import com.tencent.upload.task.impl.DirListTask;

// 第一次拉取目录传入空，拉取path下一页时填入上一次返回的result.content
String content = "";
```

```
DirListTask filetask = null;
filetask = new DirListTask(FileType.File, BUCKETNAME, DIRPATH, 10, Dentry.ListBoth, false, content,
new DirListTask.IListener() {
  @Override
  public void onSuccess(final DirCreateTask.CmdTaskRsp result) {
  ArrayList<Dentry> dirList = result.inodes;
  String strEntryList = "";
  for (Dentry entry : dirList) {
  strEntryList += entry.path + "\n";
  }
  if (result.hasMore) {
  // 存在下一页保存当前页的状态信息
  String content = result.content;
  }
  // 拉取成功
  }

  @Override
  public void onFailure(final int ret, final String msg) {
  // 拉取失败
  }
});

filetask.setAuth(SIGN);
fileUploadMgr.sendCommand(filetask);
```

列举目录下指定前缀文件&目录

通过此接口查询目录下的指定前缀的文件和目录，具体步骤如下：

1. 实例化 DirListTask 对象；
2. 调用 DirListTask 的 setPrefixSearch 方法设置前缀查询开关；
3. 调用 fileUploadMgr 的 sendCommand 方法，传入 DirListTask 对象。

4. 查询成功后，DirListTask 对象回调监听器的 onSuccess()方法，返回 DirListTask.CmdTaskRsp 对象。

方法原型

```
public DirListTask (FileType fileType, String bucket, String path, int num,
  int pattern, boolean order, String content, IListener listener)
//前缀查询开关
public void setPrefixSearch(boolean prefixSearch);
```

参数说明

参数名称	类型	是否必填	说明
fileType	FileType	是	业务类型，COS 服务统一写为 FileType.File
bucket	String	是	目录所属 bucket 名称
path	String	是	搜索的目录路径
num	int	是	拉取记录数目设定
pattern	int	是	拉取方式，Dentry.ListBoth - 目录和文件都拉取，Dentry.ListDirOnly - 只拉取目录，Dentry.ListFileOnly - 只拉取文件
order	boolean	是	true - 反序，false - 正序
content	String	否	首次拉取必须清空，拉取下一页是需要传入上一次拉取时返回的 content
listener	IListener	是	结果回调
prefixSearch	boolean	是	true - 开启前缀搜索开关，false - 关闭前缀搜索开关

返回结果说明

通过DirListTask.CmdTaskRsp对象的成员变量返回请求结果。

成员变量名称	类型	变量说明
dirCount	long	指定前缀的目录个数
fileCount	long	指定前缀的文件个数
hasMore	boolean	是否有更多
content	String	透传字段
inodes	ArrayList	指定前缀的文件详细属性
ret	int	结果码
msg	String	详细结果信息

示例

```
import com.tencent.upload.task.impl.DirListTask;

// 第一次拉取目录传入空，拉取path下一页时填入result.content
String content = "";
DirListTask filetask = null;
filetask = new DirListTask(FileType.File, BUCKETNAME, DIRPATH, 10, Dentry.ListBoth, false, content,
new DirListTask.IListener() {
  @Override
  public void onSuccess(final DirCreateTask.CmdTaskRsp result) {
  ArrayList<Dentry> dirList = result.inodes;
  String strEntryList = "";
  for (Dentry entry : dirList) {
  strEntryList += entry.path + "\n";
  }
  if (result.hasMore) {
  // 存在下一页保存当前页的状态信息
  String content = result.content;
  }
  // 拉取成功
  }

  @Override
  public void onFailure(final int ret, final String msg) {
  // 拉取失败
```

```
  }
});

filetask.setPrefixSearch(true);
filetask.setAuth(SIGN);
fileUploadMgr.sendCommand(filetask);
```

文件操作

初始化

进行文件操作之前需要先实例化 UploadManager 对象。

方法原型

```
public UploadManager(Context context, String appId, FileType fileType,
  String persistenceId)
```

参数说明

参数名称	类型	是否必填	说明
context	Context	是	
appId	String	是	腾讯云APP ID
fileType	FileType	是	业务类型，COS服务指明为：FileType.File
persistenceId	String	否	持久化 ID，每个 UploadManager 需设置一个唯一的 ID 用于持久化保存未完成任务列表，以便应用退出重进后能够继续进行上传；传入为 Null，则不会进行持久化保存

示例

```
import com.tencent.upload.UploadManager;
import com.tencent.upload.Const.FileType;

UploadManager fileUploadMgr = null;
fileUploadMgr = new UploadManager(this, "10000002", FileType.File, "qcloudobject");
```

文件上传

调用者可通过此接口进行本地文件上传操作，具体步骤如下：

1. 实例化 FileUploadTask 对象；
2. 调用 UploadManager 的 upload 方法，将 FileUploadTask 对象传入。
3. 文件上传后，FileUploadTask 对象根据上传的状态和结果调用监听器的回调接口。

方法原型

```
public FileUploadTask(String bucket, String srcFilePath, String destFilePath, String bizAttr,boolean
insertOnly,IUploadTaskListener listener)

public interface IUploadTaskListener
{
  void onUploadSucceed(FileInfo result);//上传成功

  void onUploadFailed(int errorCode, String errorMsg);//上传失败

  void onUploadProgress(long totalSize, long sendSize);//上传进度

  void onUploadStateChange(TaskState state);//上传状态变化
}
```

参数说明

参数名称	类型	是否必填	说明
bucket	String	是	目标 bucket 名称
srcFilePath	String	是	本地绝对路径
destFilePath	String	是	远程相对路径
bizAttr	String	否	文件属性，业务维护
insertOnly	boolean	否	上传是插入（true)还是覆盖（false）
listener	IUploadTaskListener	是	上传回调监听器

回调接口说明

方法名称	说明
onUploadSucceed(FileInfo result);	上传成功
onUploadFailed(int errorCode, String errorMsg);	上传失败
onUploadProgress(long totalSize, long recvDataSize);	上传进度
onUploadStateChange(TaskState state);	上传任务状态变化

回调接口中的类说明

FileInfo

成员变量名称	类型	变量说明
fileType	FileType	文件类型
url	String	文件URL
fileId	String	文件ID
extendInfo	Map	文件额外信息

TaskState

成员变量名称	类型	变量说明
code	int	状态码
desc	String	详细说明

示例

```
import com.tencent.upload.task.impl.FileUploadTask;
import com.tencent.upload.task.IUploadTaskListener;

/* 若已存在同名的文件： INSERTONLY==false 表示能够上传成功，且会覆盖同名的文件；
INSERTONLY==true 则不能成功上传*/

FileUploadTask task = new FileUploadTask(BUCKETNAME, appidFILEPATH, ARGETDIR, ATTRS,
INSERTONLY, new IUploadTaskListener(){

 @Override
 public void onUploadFailed(int arg0, String arg1) {
 // TODO Auto-generated method stub
 // 上传成功
 }

 @Override
 public void onUploadProgress(long arg0, long arg1) {
 // TODO Auto-generated method stub
 // 上传进度
 }

 @Override
 public void onUploadStateChange(TaskState arg0) {
 // TODO Auto-generated method stub
 // 上传状态变化
 }

 @Override
 public void onUploadSucceed(FileInfo arg0) {
 // TODO Auto-generated method stub
 // 上传失败
 }});
filetask.setAuth(SIGN);
```

fileUploadMgr.upload(task);

文件属性更新

用于目录业务自定义属性的更新，调用者可以通过此接口更新业务的自定义属性字段，具体步骤如下：

1. 实例化 ObjectUpdateTask 对象；
2. 调用 UploadManager 的 sendCommand 方法，将 ObjectUpdateTask 对象传入。
3. 文件属性更新成功后，ObjectUpdateTask 对象回调监听器的 onSuccess()方法，返回 ObjectUpdateTask.CmdTaskRsp 对象。

方法原型

public ObjectUpdateTask(FileType fileType, String bucket, String path, int type, String attr, int eauth, HashMap<String,String> custom_headers, IListener listener)

参数说明

参数名称	类型	是否必填	说明
fileType	FileType	是	业务类型，COS服务设置为：FileType.File
bucket	String	是	文件所属 bucket名称
path	String	是	需要更新的文件路径
attr	String	否	新的文件绑定属性
type	String	是	对象类型，文件属性更新时设置为：Dentry.FILE
eauth	int	否	文件的读写权限：文件的读写权限：与bucket拥有相同的权限（0），私有读写权限（1），私有写公有读权限（2）
cust_headers	HashMap	否	文件自定义的header
listener	IListener	是	结果回调

返回结果说明

通过ObjectUpdateTask.CmdTaskRsp对象的成员变量返回请求结果。

成员变量名称	类型	变量说明
ret	int	结果码
msg	String	详细结果信息

示例

```
import com.tencent.upload.task.impl.ObjectUpdateTask;

ObjectUpdateTask filetask = null;
Filetask = new ObjectUpdateTask(FileType.File, BUCKETNAME, FILEPATH, Dentry.FILE, ATTR,
EAUTH,CUSTOM_HEANDERS,new ObjectUpdateTask.IListener() {
  @Override
  public void onSuccess(final CmdTaskRsp result) {
  // TODO Auto-generated method stub
  // 更新成功



  }

  @Override
  public void onFailure(final int ret, final String msg){
  // TODO Auto-generated method stub
  // 更新失败
  }
  });
filetask.setAuth(SIGN);
fileUploadMgr.sendCommand(filetask);
```

文件属性查询

用于文件的查询，调用者可以通过此接口查询文件的各项属性信息，具体步骤如下：

1. 实例化 ObjectStatTask 对象；
2. 调用 UploadManager 的 sendCommand 方法，将 ObjectStatTask 对象传入。
3. 文件属性查询成功后，ObjectStatTask 对象回调监听器的 onSuccess()方法，返回 ObjectStatTask.CmdTaskRsp 对象。

方法原型

```
public ObjectStatTask(FileType fileType, String bucket, String path,
  int type, IListener listener);
```

参数说明

参数名称	类型	是否必填	说明
fileType	FileType	是	业务类型，COS 服务设置为：FileType.File
bucket	String	是	文件所属 bucket名称
path	String	是	需要查询的文件路径
type	String	是	对象类型，文件属性查询时设置为：Dentry.FILE
listener	IListener	是	结果回调

返回结果说明

通过ObjectStatTask.CmdTaskRsp对象的成员变量返回请求结果。

成员变量名称	类型	变量说明
inode	Dentry	文件的详细属性
ret	int	结果码
msg	String	详细结果信息

示例

```
import com.tencent.upload.task.impl.ObjectStatTask;
```

```
ObjectStatTask filetask = null;
filetask = new ObjectStatTask(FileType.File, BUCKETNAME, FILEPATH, Dentry.FILE, new
ObjectStatTask.IListener() {
  @Override
  public void onSuccess(final ObjectStatTask.CmdTaskRsp result) {
  // 查询成功
  }

  @Override
  public void onFailure(final int ret, final String msg) {
  // 查询失败
  }
});

filetask.setAuth(SIGN);
fileUploadMgr.sendCommand(filetask);
```

文件删除

调用此接口可进行文件的删除操作，具体步骤如下：

1. 实例化 ObjectDeleteTask 对象；
2. 调用 UploadManager 的 sendCommand 方法，将 ObjectDeleteTask 对象传入。
3. 文件删除成功后，ObjectDeleteTask 对象回调监听器的 onSuccess()方法，返回 ObjectDeleteTask.CmdTaskRsp 对象。

方法原型

```
public ObjectDeleteTask(FileType fileType, String bucket, String path,
  int type, IListener listener)
```

参数说明

参数名称	类型	是否必填	说明
fileType	FileType	是	业务类型，COS服务设置为：FileType.File
bucket	String	是	文件所属 bucket名称
path	String	是	删除文件的路径
type	int	是	对象类型，删除文件时设置为：Dentry.FILE
listener	IListener	是	结果回调

返回结果说明

通过ObjectDeleteTask.CmdTaskRsp对象的成员变量返回请求结果。

成员变量名称	类型	变量说明
ret	int	结果码
msg	String	详细结果信息

示例

```
import com.tencent.upload.task.impl.ObjectDeleteTask;

ObjectDeleteTask filetask= null;
filetask = new ObjectDeleteTask(FileType.File, BUCKETNAME, FILEPATH, Dentry.FILE, new
ObjectDeleteTask.IListener() {
  @Override
  public void onSuccess(ObjectDeleteTask.CmdTaskRsp result) {
  // 删除成功
  }

  @Override
  public void onFailure(final int ret, final String msg) {
  // 删除失败
  }
});

filetask.setAuth(SIGN);
```

fileUploadMgr.sendCommand(filetask);

对象下载

初始化

在使用对象下载器时，需要先实例化一个Downloader对象，并对下载器的一些属性进行配置。

方法原型

public Downloader(Context context, String appid, String persistenceId);

参数说明

参数名称	类型	是否必填	说明
context	Context	是	
appid	String	是	腾讯云注册的APP ID
persistenceId	String	否	资源的持久化ID

示例

Downloader downloader = new Downloader(this, "10000002","qcloudobject");

下载并发数设置：可以指定下载器最大并发数。

方法原型：

public void setMaxConcurrent(int count);

示例：

downloader.setMaxConcurrent(10);

长连接/断点续传设置：下载器提供开关，可以设定是否开启长连接和断点续传功能。

方法原型

```
// enable True-支持断点续传，False-不支持
public void enableHTTPRange(boolean enable);
// enable Ture-支持HTTP长连接，False-不支持
public void enableKeepAlive(boolean keepalive);
```

示例

```
downloader.enableHTTPRange(true);
downloader.enableKeepAlive(true);
```

对象下载器

对已知URL的资源进行下载，异步模式进行。

方法原型

```
public boolean download(String url, DownloadListener listener);

public interface DownloadListener{
  public void onDownloadSucceed(String url, DownloadResult result);
  public void onDownloadFailed(String url, DownloadResult result);
  public void onDownloadCanceled(String url);
  public void onDownloadProgress(String url, long totalSize, float progress);
}
```

参数说明

参数名称	类型	是否必填	说明
url	String	是	请求资源的URL地址
listener	DownloadListener	是	回调结果

监听器接口说明

方法名称	说明
onDownloadSucceed(String url, DownloadResult result);	下载成功
onDownloadFailed(String url, DownloadResult result);	下载失败
onDownloadCanceled(String url);	取消下载
onDownloadProgress(String url, long totalSize, float progress);	下载进度

监听器接口中参数说明

参数名称	类型	说明
url	String	请求资源的URL地址
result	DownloadResult	下载状态

示例

```
DownloadListener listener = new DownloadListener() {
  @Override
  public void onDownloadSucceed(String url, DownloadResult result) {
  // 下载成功
  }

  @Override
  public void onDownloadProgress(String url, long totalSize, float progress) {
  // 下载进度
  }

  @Override
  public void onDownloadFailed(String url, DownloadResult result) {
  // 下载失败
```

```
  }

  @Override
  public void onDownloadCanceled(String url) {
  // 下载被取消
  }
};

downloader.download(URL, listener);
```

取消下载任务

取消对指定URL资源的下载任务，或取消所有 downloader 开启的下载任务。

方法原型

```
//取消指定URL资源下载任务
public void cancel(String url, DownloadListener listener);
//取消所有下载任务
public void cancelAll();
```

监听器接口中参数说明

参数名称	类型	说明
url	String	请求资源的URL地址
result	DownloadResult	下载状态

示例

```
downloader.cancel(URL, listener);
downloader.cancelAll();
```